# Ökad tillgänglighet för statistiska data

en studie av några gränssnitt mot statistiska databaser

Mattias Lindberg

R&D Report
Statistics Sweden
Research - Methods - Development
1993:9

INLEDNING

TILL

**R & D report : research, methods, development / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1988-2004. – Nr. 1988:1-2004:2.**
**Häri ingår Abstracts : sammanfattningar av metodrapporter från SCB med egen numrering.**

**Föregångare:**

Metodinformation : preliminär rapport från Statistiska centralbyrån. – Stockholm : Statistiska centralbyrån. – 1984-1986. – Nr 1984:1-1986:8.

U/ADB / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1986-1987. – Nr E24-E26

R & D report : research, methods, development, U/STM / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1987. – Nr 29-41.

**Efterföljare:**

Research and development : methodology reports from Statistics Sweden. – Stockholm : Statistiska centralbyrån. – 2006-. – Nr 2006:1-.

R & D Report 1993:9. Ökad tillgänglighet för statistiska data : en studie av några gränssnitt mot statistiska databaser / Mattias Lindberg.
Digitaliserad av Statistiska centralbyrån (SCB) 2016.

urn:nbn:se:scb-1993-X101OP9309

# Ökad tillgänglighet för statistiska data

en studie av några gränssnitt mot statistiska databaser

Mattias Lindberg

R&D Report
Statistics Sweden
Research - Methods - Development
1993:9

Från trycket Oktober 1993
Producent Statistiska centralbyrån, utvecklingsavdelningen
Ansvarig utgivare Lars Lyberg
Förfrågningar Erik Malmborg, tel 08-783 40 27


ISSN 0283-8680

Printed October 1993
Producer Statistics Sweden,
Department of Research and Development
S-115 83 Stockholm
Publisher Lars Lyberg
Inquiries Erik Malmborg, telephone +46 08 783 40 27


ISSN 0283-8680

## Referat

Denna rapport innehåller en undersökning av olika sätt att visualisera statistiska metadata på. Dessa visualiseringar ska användas för att konstruera frågor, som ska bygga upp en statistisk tabell. Undersökningen är uppdelad i två delar, en litteraturstudie och en studie av befintlig programvara. I litteraturstudien förklarar och kommenterar jag följande fyra lösningsförslag: Entity-Relationship diagram, träddiagram, naturliga språk och kommandospråk. De tre program som undersöks är Supercross, GQL och PC-AXIS.

I början av rapporten finns det en introduktion till vad som är karaktäristiskt för statistiska databaser samt till notationer och begrepp som används på SCB (Statistiska Centralbyrån). I slutet så presenteras en prototyp, baserad på Entity-Relationship diagram, som jag har utvecklat under den tid jag har skrivit rapporten.

## Abstract

This report contains an examination of different ways to visualise statistical metadata. This visualisations are to be used during query formulation, to construct a statistical table. The examination is divided into two parts, a study of related literature and a study of three commercially available programs. In the literature study I explain and comment the following four approaches: Entity-Relationship diagrams, tree diagrams, natural languages and command languages. The three programs examined are Supercross, GQL and PC-AXIS.

In the beginning of the report I have included an introduction to the characteristics of statistical databases and notations used at SCB (Statistics Sweden). In the end I present a prototype, based on the Entity-Relationship approach, which I have developed during the time I wrote this report.

## Innehållsförteckning

## Förord

Denna rapport är mitt examensarbete i datalogi vid NADA KTH. Examensarbetet har gjorts som en del av min fyraåriga utbildning på Matematisk-Naturvetenskaplig linje, med inriktning mot datalogi, vid Stockholms Universitet. Arbete är utfört på Statistiska Centralbyråns (SCB) Utvecklingsavdelning i Stockholm under sommaren 1993. Handledare på NADA har varit Serafim Dahl, på SCB har Erik Malmborg varit handledare.

Jag skulle vilja tacka Erik Malmborg för de synpunkter och uppslag som han har kommit med under arbetets gång.

Stockholm i september 1993

Mattias Lindberg

# 1. Bakgrund

Statistiska Centralbyrån (SCB) startade hösten 1992 ett projekt som syftar till att göra SCBs datamaterial mer åtkomligt utanför SCB. Detta projekt kallas Tillgänglighetsprojektet. Målet med Tillgänglighetsprojektet är att göra det möjligt för personer med en egen dator att använda data från SCBs olika databaser.

# 2. Problemställning

När en forskare eller statistiker vill konstruera en tabell utifrån en mängd data som ligger lagrade i en databas, så behövs det någon sorts gränssnitt mot databasen. Detta gränssnitt bör hjälpa en användare att hitta bland den information som finns i databasen, detta kan ske genom att på något sätt visualisera, grafiskt eller textuellt, den metadata som finns lagrad i databasen. Hur denna visualisering kan genomföras ska jag diskutera i denna rapport.

## 2.1. Allmänt

När man har stora mängder data samlade i ett antal olika databaser kan det vara svårt att få en överblick av vilken information som egentligen finns tillgängligt. För att underlätta användandet av dessa databaser så bygger man program som kapslar in dem, och ger dem ett mer användarvänligt gränssnitt. Att sådana inkapslingar verkligen behövs förstår man om man ställer t.ex. TAB68 och Supercross mot varandra (se respektive avsnitt för en beskrivning av dessa system).

Den information som finns i en statistisk databas är av tre olika typer, som alla har olika funktion:

- **Mikrodata** är de insamlade data som ligger till grund för allt arbete, dessa kan t.ex. vara information från en folkräkning.
- **Makrodata** är bearbetade mikrodata som kan läggas in för att t.ex. snabba upp vanliga sökningar genom att gruppera eller summera mikrodata. Exempel på makrodata kan vara en lista över Sveriges befolkning indelade efter kommun. Ofta kallas makrodata även för aggregerade data.
- **Metadata** är data om data, d.v.s. en beskrivning av den struktur som mikro- respektive makrodata ligger lagrade i. Här återfinns bland annat information om vilka olika datamängder som finns i databasen, samt vilka samband som finns mellan dessa.

En vanlig tillämpning mot databaser, speciellt statistiska databaser, är konstruktion av tabeller, d.v.s. att välja ut vissa data ur en stor databas och visa dessa på ett ordnat sätt. Problemet med denna typ av tillämpningar är ofta hur man ska lotsa fram användaren till rätt data, med detta menas hur databasens metadata ska visualiseras på ett sätt som passar användaren. I allmänhet så kan man säga att tabellkonstruktion är en process som kan delas upp i tre steg:

- Samordning av mikrodata
- Aggregering till makrodata
- Presentation av de data man har valt

I det första steget, samordning av mikrodata, görs en samkörning av olika register, detta sker för att skapa ett basregister som sedan används för att påvisa samband mellan olika typer av områden. Samkörningen består av:

- Val av de register som ska ingå i samkörningen, samt vilka variabler som är intressanta i varje register.
- Matchning av posterna i de olika registren, t.ex. genom personnummer.

I steg två, aggregering till makrodata, bestämmer man vilka data som ska synas i tabellen, ofta så har man här flera möjliga datamängder att arbeta mot. Deluppgifterna blir därför:

- Aggregering av materialet till en matris, man har då skapat makrodata.
- Val av tabell eller relation som man vill undersöka
- Plocka ut några variabler ur tabellen/relationen

I presentationsdelen, som är det sista men kanske viktigaste steget, så bestäms hur datamaterialet ska presenteras på skärmen. Att detta är viktigt beror på att om man inte lyckas visa sina resultat på ett bra sätt så får man ingen nytta av dem. Detta steg innefattar:

- Specificera vilken del av variabelns värdemängd som man vill ha med
- Placera dessa värdemängder i förspalt eller överspalt
- Välja om man ska göra någon ytterligare aggregering och/eller gruppera data på något vis
- Välja rubriker, font m.m.

Dagens datoranvändare är vana att manipulera olika objekt, t.ex. ikoner, knappar och menyer, detta gör att ett användargränssnitt bör utnyttja dessa typer av manipulations möjligheter. Om man ser på vårt problem utifrån detta så inser man att det behövs någon form av struktur för att beskriva vilka tabeller som finns i databasen samt hur dessa hänger samman. Man bör också på något sätt bli informerad om en tabells variabler när man markerar denna, och på samma sätt bör man informeras om variabelns värdemängd när denna väljs. Vilken struktur som ska användas samt hur man kan specificera var olika värden ska hamna behandlas i kapitel 5 (Undersökning av möjliga lösningar).

## 2.2. Avgränsning

Den prototyp som jag har utvecklat har inte någon direkt koppling till SCBs riktiga databaser, utan generering av data sköts via en testdatabas. Prototypen tillåter en användare att bygga upp en tabell och presenterar sedan en SQL-fråga som hämtar in det material som behövs för tabellen. Prototypen tillåter inte förändring, borttagning eller nyskapande av register eller register mellan tabeller. Dessa avgränsningar görs då syftet med arbetet främst är att studera olika aspekter på tabellframställning, och alltså inte inriktar sig på uppbyggnad av databasen.

Prototypen har utvecklats i Smalltalk/V för Windows version 2.0. Detta system har en rad färdiga byggstenar som har underlättat konstruktionen av användargränssnittet. Den del som fattades var en s.k. spread-sheet funktion som behövs för att visa de färdiga tabellerna, då det fanns en sådan funktion tillgänglig från en annan Smalltalk så skrevs den om för att passa till Smalltalk/V för Windows. Metadata för den databas som modelleras i prototypen hämtas in till prototypen med hjälp av Q+E Database Library (QELIB), detta verktyg kan kopplas till en mängd olika program och databaser. Vi använder dBase-filer samt SQL för databashanteringen.

För att få ett konkret exempel att arbeta med så anknyter min prototyp till DAISY (DAtaförsörjnings och InformationsSYstem för arbetsmarknads-forskare). Detta sker genom att några av de register som ingår i DAISY kommer att ligga till grund för den modell som prototypen använder. Mer om DAISY finns att läsa i ett senare avsnitt.

## 3. Syfte

Syftet med arbetet är att:

- Genom litteraturstudier och undersökning av befintlig programvara studera olika möjligheter att definiera användargränssnitt mot statistiska databaser.
- Specificera ett eget gränssnitt.
- Implementera en prototyp av det egna gränssnittet.

# 4. Introduktion till statistiska databaser på SCB

I detta kapitel ges en förklaring av olika begrepp som förekommer i rapporten, samt en bakgrund till DAISY. De begrepp som förklaras är statistiska databaser, OPREM och αβγ-analys.

För en mer fullständig förklaring av statistiska databaser är [Shoshani 80] lämplig läsning. Det mesta av det som behandlas i avsnitten om OPREM och αβγ-analys finns beskrivet i [Sundgren 84], αβγ-analysen grundar sig även på [Langefors 75].

## 4.1. Allmänna egenskaper hos statistiska databaser

Vad är det som skiljer en statistisk databas från andra typer av databaser? Nedan följer ett försök att förklara vad som är karakteristiskt för just statistiska databaser. I allmänhet så skiljer man på statistiska databaser, som innehåller makrodata, och 'scientific databases', som innehåller mikrodata. I denna rapport kommer jag däremot ej att göra denna distinktion, detta beror på att i Tillgänglighetsprojektet och DAISY finns avsikten att tillåta användaren att få arbeta direkt mot de mikrodata som finns på SCB.

I en statistisk databas har man två typer av data, mätdata och parameterdata (eng. summary- och category attributes). Dessa skiljer sig åt genom att mätdata oftast är numeriska värden av något slag, medan parameterdata ofta bara kan anta ett litet antal värden, t.ex. så kan kön anta värdena man eller kvinna. Till varje mätdata så finns det en kombination av parametervärden, dessa beskriver vad som har mätts. Eftersom parameterdata har ett väl definierat värdeförråd så kan de kodas effektivt och på så sätt spara lagringsutrymme. Parameterdata grupperas även ofta för att få en bättre överblick av mätdata, t.ex. så kan man dela in ålder i grupper om fem år istället för att presentera varje åldersgrupp för sig själv. Det är vanligt att man i en statistisk databas inte har mätdata för alla möjliga kombinationer av parameterdata, man säger att de är glesa. Orsaken till detta illustreras bäst med ett exempel. Antag att vi studerar olika typer av industrier i Sveriges kommuner, då kommer fiskeindustrin inte att vara representerad i inlandskommunerna, medan gruvindustrin kommer att förekomma främst i norrlandskommunerna.

En viktig egenskap hos statistiska databaser är att de är stabila, d.v.s. de mätdata som en gång har lagts in i databasen ändras inte. Detta beror på att statistiska data ofta samlas in för framtida behov.

Ett problem är att det kan vara svårt att hålla namnen på alla olika tabeller, giltiga värdemängder, förkortningar m.m. i huvudet, detta gäller de flesta typer av databaser. Ett problem som speciellt förekommer i statistiska databaser är att man ändrar på parameterdatas värdemängder, t.ex. kan kommuner slås samman eller delas upp, detta gör att man hela tiden måste uppdatera sin kunskap om databasen.

I en statistisk databas så arbetar man inte alltid mot de ursprungligt insamlade värdena, istället så används föraggregerade data. Föraggregerade data är data som har bearbetats på något vis, t.ex. kan man summera över de värden som en variabel kan anta. Detta medför att information går förlorad, men man vinner både hastighet och lagringsutrymme.

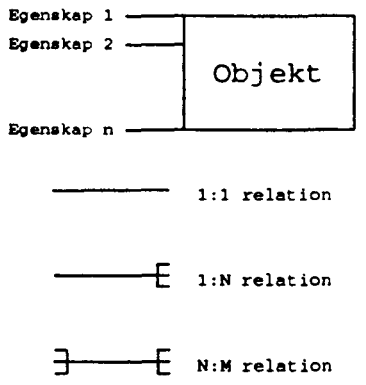
## 4.2. OPREM

Den utvecklingsmetod för databasapplikationer som används på SCB kallas "SCBs systemeringsmodell" eller OPREM, vilket är en förkortning för Object-Property-Relationship-Event-Message. OPREM består av två huvudfaser, den infologiska fasen och den datalogiska fasen. I den infologiska fasen så koncentrerar man sig på sammanhanget som databasen ska användas i. De saker som man specificerar i denna fas är de som är saker som är relaterade till användaren, t.ex. vilken information som ska lagras i databasen och hur långa svarstider man kan acceptera. Under den datalogiska fasen så är man inriktad på att konstruera ett system som uppfyller de specifikationer som den infologiska fasen har givit.

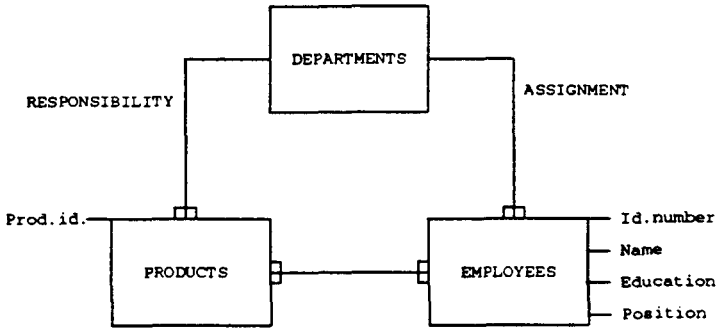
Trots denna uppdelning så är det inte nödvändigt att den infologiska delen är helt slutförd innan den datalogiska delen påbörjas. Ofta så kan de överlappa lite, men man måste vara försiktig så att man inte lägger ner för mycket arbeta på en viss sak i den datalogiska modellen som senare kommer att ändras i den infologiska modellen och därmed även i den datalogiska modellen. Det är även positivt om det förekommer kommunikation mellan de två grupperna, man kan då förklara eventuella begränsningar, tekniska eller ekonomiska, som kan finnas och modifiera modellerna så att problemen kringgås.

### 4.2.1. Infologiska delen

Den infologiska modellen ska specificera hur man vill att systemet ska se ut för en användare, för att kunna göra detta så behöver man en notation som beskriver modellen. Den notation som används i OPREM är av Entity-Relationship-typ, och bygger på begreppen objekt, egenskaper och relationer. Ett objekt är något som användarna är intresserad av att få information om, det kan vara ett fysiskt objekt, en organisation, m.m. . Ett objekt har egenskaper, d.v.s. information, som karaktäriserar objektet. Det kan finnas samband och beroenden mellan olika objekt, dessa kallas relationer. Ett begrepp som hör samman med relationer är kardinalitet, detta beskriver hur många kopplingar det kan finnas till ett visst objekt. Exempel på en relation kan vara att en person arbetar på ett visst företag, varje person arbetar på endast ett företag medan ett företag kan ha flera anställda. Den notation som används för att visuellt beskriva dessa begrepp visas i Figur 4.1, och i Figur 4.2 finns ett exempel på en objektgraf.




**Figur 4.1.** Grafiskt notation för objekt, egenskap och relation i OPREM




**Figur 4.2.** Exempel på objektdiagram

Den del av OPREM som kallas för Event-delen analyserar hur databasen påverkas av olika händelser (events). Detta görs genom att man gör en lista på de händelser som kan inträffa i systemet, sedan så undersöker man vilka objekt och relationer som ändras vid varje händelse. Med ändras så menar man främst att det skapas en ny instans eller att man tar bort en instans. Resultatet av undersökningen presenteras i något som kallas för konsekvens matris, och som innehåller en rad för varje händelse och en kolumn för varje objekt eller relation, se Tabell 4.1.

| Event/Object, Relation | EMPL | PROD | DEPT | EMPL/DEPT | PROD/DEPT | EMPL/PROD |
|---|---|---|---|---|---|---|
| a new person is employed | C | | | C | | C* |
| an employee leaves | D | | | D | | D* |
| new product | | C | | | C | C* |
| discontinuation of product | | D | | | D | D* |
| reorganization | | | C*, D* | C*, D* | C*, D* | |
| employee moves | | | | C, D | | |
| responsibility transferred | | | | | C, D | |
| replanning of work | | | | | | C*, D* |

**Tabell 4.1.** Tabell sid 70 i [Sundgren 84]. C= Creation, D=Destruction

Den sista delen i OPREM, Message-delen, består av att precisera de informations behov som de tidigare delarna av OPREM har beskrivit. Detta måste göras för att undvika tvetydigheter, som kan leda till att det system som konstrueras inte uppfyller de krav som användarna ställde. Den metod som OPREM använder för att formellt beskriva informations behov kallas αβγ-analys (se **Beskrivning av αβγ-analys** nedan), och man använder elementära αβγ-frågor för att undersöka hur ofta olika objekt eller relationer refereras. Dessa frekvenser används sedan för att avgöra om man ska lägga upp ett index på detta objekt för att snabba upp sökningen.

### 4.2.2. Datalogiska delen

I den datalogiska fasen så ska man realisera den specifikation som har utarbetats under den infologiska delen, d.v.s. man ska skapa ett databas-hanteringssystem som uppfyller de krav som är ställda. För att beskriva den databas som man bygger upp så använder man relationsdatabasmodellen och man arbetar med databaser i 3NF.

När man ska omvandla den infologiska modellen till datalogiska så kan i stort sett överföra de objekt, relationer och egenskaper som man tidigare har fått fram till relationsdatabasens struktur. Vissa omvandlingar görs när man normaliserar modellen till 3NF, men informationsmängden är den samma. Ett problem man har är vad man ska göra om man har redundant information, t.ex. om man har bruttoinkomst, skatt och nettoinkomst så kan två av dessa användas för att beräkna den tredje. Den redundanta informationen kan inte bara tas bort, eftersom man kan komma att behöva den, utan man får göra en avvägning om man kan behålla informationen eller om man ska ordna med en procedur som beräknar informationen ur den som finns. Man kan även behöva tillföra information för att snabba upp sökningar, t.ex. kan man lägga upp index på de variabler (attribut) som kommer att efterfrågas ofta.

## 4.3. Beskrivning av αβγ-analys

αβγ-analys är ett sätt att beskriva innehållet i frågor som ställs mot databaser, frågorna är av två typer, αβ-frågor och αβγ-frågor, där αβγ-frågor är typiska för statistiska databaser.

I [Langefors 75] definieras αβ-frågor som frågor med följande mönster:

> För alla objekt som har egenskapen $P^{\alpha}$, hämta värdena för attributen $A^{\beta}_1,...,A^{\beta}_m$, vid tidpunkterna $t^{\beta}_1,...,t^{\beta}_m$, respektive.

När man gör beskrivningar med hjälp av αβ-frågor så brukar man utelämna tidsdelen om man talar om 'nu'. $P^{\alpha}$ är ofta ett uttryck som består av flera delar, $P^{\alpha}_1,...,P^{\alpha}_k$, som alla måste vara uppfyllda för att $P^{\alpha}$ ska vara uppfyllt.

Exempel på αβ-fråga:

"Lista namn och telefonnummer på de anställda på NADA som bor på Terminalsalsvägen 17 och äger mer än tre datorer".

Här har vi alltså:

$P^{\alpha}$ = personer anställda på NADA som bor på Terminalsalsvägen 17 och äger mer än tre datorer.
$A^{\beta}_1$ = namn
$A^{\beta}_2$ = telefonnummer
Och $P^{\alpha}$ kan delas upp i:
$P^{\alpha}_1$ = att vara en person
$P^{\alpha}_2$ = att ha en anställning på NADA
$P^{\alpha}_3$ = adress == Terminalsalsvägen 17
$P^{\alpha}_4$ = #datorer > 3

Det som tillkommer i αβγ-frågor är ett γ-attribut ($V^{\gamma}$) som anger hur man ska partitionera resultatet. Själva frågan blir n stycken αβ-frågor där α-delen modifieras till att vara $P^{\alpha}_i = P^{\alpha}_o \wedge (V^{\gamma} = a_i)$ med i = 1...n. Här är $P^{\alpha}_o$ α-delen enligt αβ-frågeanalysen, och n är storleken på värdemängden för den variabel som $V^{\gamma}$ specificerar. Om man har flera γ-attribut, säg k stycken, så blir $n = n_1 \times n_2 ... \times n_k$, och α-delen blir (k + 1) stycken konjunktioner.

Enkelt exempel med αβγ-notation:
"Lista namnen på de anställda indelade efter avdelning".
Här har vi alltså:
$P^{\alpha}_0$ = anställda personer (kan delas upp i $P^{\alpha}_1$ och $P^{\alpha}_2$ som ovan)
$A^{\beta}_1$ = namn
$V^{\gamma}$ = avdelnings namn
Säg att det finns tre avdelningar, nämligen produktion, försäljning och underhåll. Det tre αβ-frågorna får då samma β-del men α-delarna blir:
$P^{\alpha}_1 = P^{\alpha}_0 \wedge (V^{\gamma} = \text{'produktion'})$
$P^{\alpha}_2 = P^{\alpha}_0 \wedge (V^{\gamma} = \text{'försäljning'})$
$P^{\alpha}_3 = P^{\alpha}_0 \wedge (V^{\gamma} = \text{'underhåll'})$

En allmän αβγ-fråga kan skrivas i SQL-notation på följande vis:

β : SELECT <variables>
α : FROM <object type>
    WITH <property>
γ : GROUP BY <variables>

Ett sätt att använda αβγ-analys är att bryta ner dessa i elementära αβ-frågor, d.v.s. frågor som använder endast ett objekt eller en relation, dessa används i sin tur för att uppskatta vad i databasen som kommer att användas oftast. Denna uppskattning kan ligga till grund för hur man bygger upp t.ex. index i databasen.

## 4.4. DAISY

DAISY är ett system avsett att användas inom arbetsmarknadsforskning för att underlätta sökandet efter olika statistiska uppgifter. Systemet består dels av en metadatabas och dels ett utsökningssystem. Metadatabasen innehåller bl.a. information om de olika ingående registren, vilka register som går att

kombinera och kvalitén på data. Syftet med DAISY är att man ska ge forskare möjlighet att arbeta direkt mot mikrodata, och alltså inte vara bunden av föraggregerade data, på detta vis kommer man bättre att kunna anpassa och ta fram statistik som en enskild forskare efterfrågar. Forskaren möjligheter att göra en egen analys av statistiskt materialet kommer även att öka väsentligt. Grunden i DAISY kommer att vara ett flertal register som behandlar individer och arbetsställen, dessa kommer att anpassas så att samkörningar underlättas. Ett problem som ej behandlas i denna rapport är hur individdata ska skyddas, för att lösa detta behöver man utnyttja en kombination av tekniska och juridiska åtgärder.

I dagens läge så finns det bara en första version av metadatabasen, medan utsökningssystemet ännu inte är klart. Mitt arbete kan att ses som en förstudie till ett sådant.

# 5. Undersökning av möjliga lösningar

För att få en uppfattning om hur man kan konstruera tabeller från statistiska databaser så har jag dels läst artiklar, kompendier och böcker som anknyter till ämnet, och dels undersökt tre program. De artiklar som har ingått i litteraturstudien kommer främst från konferensserier om databaser.

## 5.1. Litteraturstudie

I detta avsnitt redogör jag för olika sätt att visualisera den information som finns lagrad i databaser och hur man kan välja ut vilken information som ska presenteras i en tabell. Det finns även ett avsnitt om vilka möjligheter till bearbetning av färdiga tabeller som bör finnas.

En allmän uppdelning av användargränsnitt mot databaser kan göras enligt följande:

- **Grafiska:** En modell av databasen presenteras grafisk för användaren. En fråga specificeras genom att manipulera modellen med t.ex. en mus. Exempel på en grafisk modell kan vara den utvidgade Entity-Relationship modellen som finns i OPREM.
- **Menybaserade:** Användaren får upp menyer som används för att bygga upp en fråga steg för steg. En fördelen med dessa är att användaren inte behöver komma ihåg kommandon och dess syntax.
- **Formulär-baserade:** Användaren får ett formulär presenterat på skärmen, detta formulär representerar den information som finns lagrad i databasen. Genom att fylla i vissa fält kan man göra en sökning som hittar de poster som matchar de ifyllda fälten.
- **Kommandospråk:** Specialiserade språk där man ofta har den bästa kontrollen av databasen. Nackdelen med dessa är att de ofta är svåra att lära sig och att användaren måste veta vad databasen innehåller för att kunna ställa sina frågor.
- **Naturliga språk:** Man skriver sin fråga i ett vanligt språk, t.ex. engelska. Denna fråga tolkas av datorn, som skickar den vidare till databasen. Ett problem med naturliga språk är att det kan vara svårt att få den exakthet som krävs, detta löses ofta genom att datorn frågar användaren om eventuella oklarheter.

| | Grafiska | Menybaserade | Formulärbaserade | Kommandospråk | Naturliga språk |
|---|---|---|---|---|---|
| inlärnings tid | kort | kort | kort | lång | kort |
| hast. vid anv. | medium | medium | hög | hög | medium |
| fel frekv. | låg | låg | låg | hög | hög |
| utvidgbarhet | liten | medium | medium | bra | bra |
| skrivförmåga | ingen | ingen | hög | hög | hög |

**Tabell 5.1.** En jämförelse mellan de olika gränssnitten, tabellen är hämtad ur [Foley 90] och visas här i en omarbetad form.

När det gäller just gränssnitt till statistiska databaser så hittar vi följande krav på dessa i [Elmasri 89]:

> "Adequate support to remember names of hundreds of fields and their abbreviations is an absolute necessity for statistical users. Any interface must include some sort of browsing. Users' exploratory investigation and incremental query building must be supported. Users should be able to view record internal structure with some semantically meaningful groupings. *Meta-data browsing* support is essential for users to determine which database or what portion of a database they should be looking at."

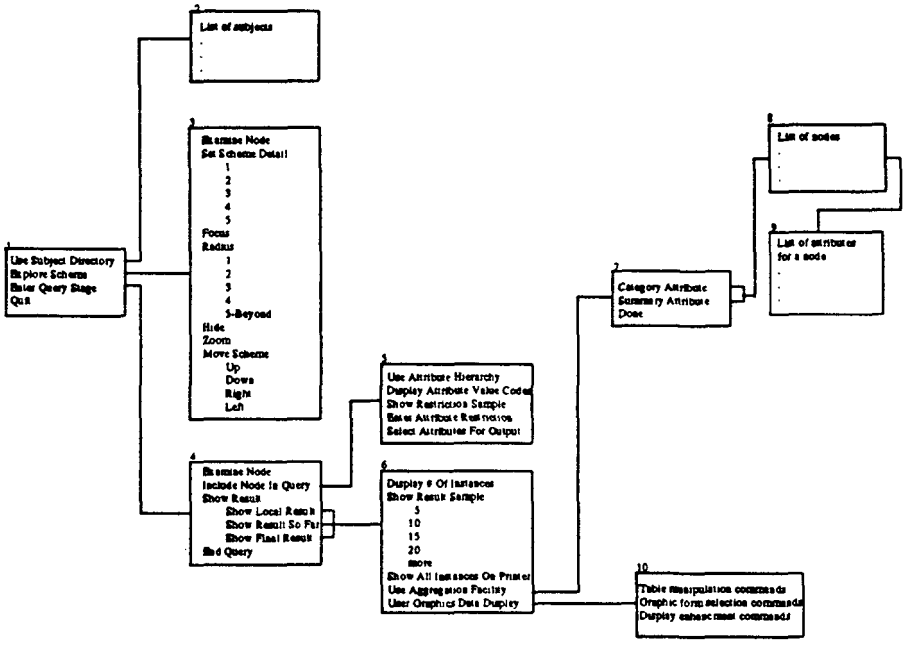
### 5.1.1. Entity-Relationship modell

Entity-Relationship (E-R) modellen kan användas för att illustrera vilka objekt som finns i databasen, och hur dessa hänger samman. Detta är ett sätt att ge en beskrivning av databasens struktur, och borde därför gå att använda som användargränssnitt. Denna ansats finns beskriven i [Malmborg 88] och [Wong 83].

Tanken i [Malmborg 88] är att man ska ha ett användargränssnitt som är mode-löst, WYSIWYG och samtidigt ha kvar skrivbordsmetaforen. För att visualisera databasen så använder man en metadatabas för att rita upp en modell av den typ som finns i OPREM, denna modell är alltid synlig i sin helhet. För att få WYSIWYG under konstruktionen av tabellen så har man ett fönster, eller del av fönster, där man kontinuerligt uppdaterar tabellens utseende under det att den byggs upp. När man väljer ett objekt i modellen, genom att markera det med musen, så visas en meny med olika alternativ, t.ex. visa en beskrivning av objektet, visa en förklaring av var i databasen som objektet lagras och visa de variabler som är associerade med objektet. Om man väljer att visa variablerna så kommer dessa upp i en ny meny, och om man i denna väljer en variabel så visas en ny meny med dess värdemängd. Ur denna värdemängd så väljer man vilka värden man vill ha med i tabellen. För att sedan flytta över dessa till tabellen så kan man antingen använda copy-and-paste eller drag-and-drop. När man flyttar över värdena till tabellen så väljer man om de ska stå i förspalt eller överspalt, och beroende på detta så uppdateras utseendet på dessa, d.v.s. man utökar antalet nivåer och fördelar värdena som de ska vara. När man har valt de delar som man vill ha i tabellen så kan man välja om man vill göra några beräkningar på elementen i tabellen. Även detta väljs från en meny.

Det system som beskrivs i [Wong 83] kallas GUIDE (Graphical User Interface for Database Exploration), systemet visar en Entity-Relationship modell över databasen och använder sig av menyer för att ge kommandon. Två begrepp som används i GUIDE är radie och detaljnivå. Med radie så menar man det största avståndet från aktuellt objekt som visas i modellen, mätt i antal länkar. Detaljnivån används för att lättare kunna urskilja viktiga och vanligt använda delar av modellen, detta görs genom att varje del i modellen får en 'rank' och bara de delar med 'rank' <= detaljnivå visas.

Det första en användare gör efter att ha startat GUIDE är att välja vilket ämnesområde som frågan ska behandla. Det valda ämnesområdet presenteras som en Entity-Relationship modell, i vilken man kan vandra omkring och undersöka vad de olika delarna innehåller. Det finns även möjlighet att manipulera modellen så att man får en bättre överblick över de delar som är relevanta, de operationer som är aktuella är t.ex. att dölja en del av grafen som ej används, ändra detaljnivån och ändra radien (dessa alternativ återfinns i meny nummer 3, figur 5.1).



**Figur 5.1.** Menyhierarkin i GUIDE

För att bygga upp en fråga så markerar man en nod i modellen och väljer *Include Node in Query* från menyn. Då visas en undermeny (nr. 5 i figur 5.1) där man bl.a. kan se vilka attribut ett objektet har, hur de är kodade samt välja att aggregera över eller lägga restriktioner på dessa. En fråga i GUIDE visas som en sammanhängande väg genom modellen, det är också möjligt att ha flera mindre delfrågor definierade samtidigt, dessa kodas då i olika färger. Orsaken till att man har infört delfrågor i GUIDE är att användaren ska kunna koncentrera sig på en liten del av modellen och att man inte ska behöva bygga en stor och komplicerad fråga med en gång, dessa delfrågor kan sedan knytas samman till en fråga med ett enkelt kommando. För att visa resultatet av en fråga så väljer man något av alternativen från meny nr. 4 (figur 5.1), skillnaden mellan de två sista alternativen är att när man visat det slutliga resultatet så antar systemet att man vill komponera en ny fråga efteråt. När man har valt någon av dessa så kommer det upp en meny (nr. 6, figur 5.1) där man väljer hur man vill visa resultatet. För att redigera hur resultatet ska presenteras så väljer man alternativet *Use Graphics Data Display*. Den meny som då kommer upp ger användaren möjlighet att manipulera tabellen på olika vis, t.ex. gruppering av kolumner, att välja i vilken form som värdena ska visas, t.ex. piechart,

barchart, och det finns även möjlighet att skugga områden och lägga till rubriker.

I den beskrivning av systemet som gjorts ovan så har jag avsiktligt valt undvika frågan om hur menyerna presenteras, det beror på att det varierar, antingen så läggs de ute i högre delen av skärmen eller så läggs de i det område som används av modellen. Det är alltså inte så att man får upp alla menynivåer på skärmen samtidigt, utan de kan närmast liknas vid skärmbildsmenyer, eller att de alltid kommer upp på samma ställe.

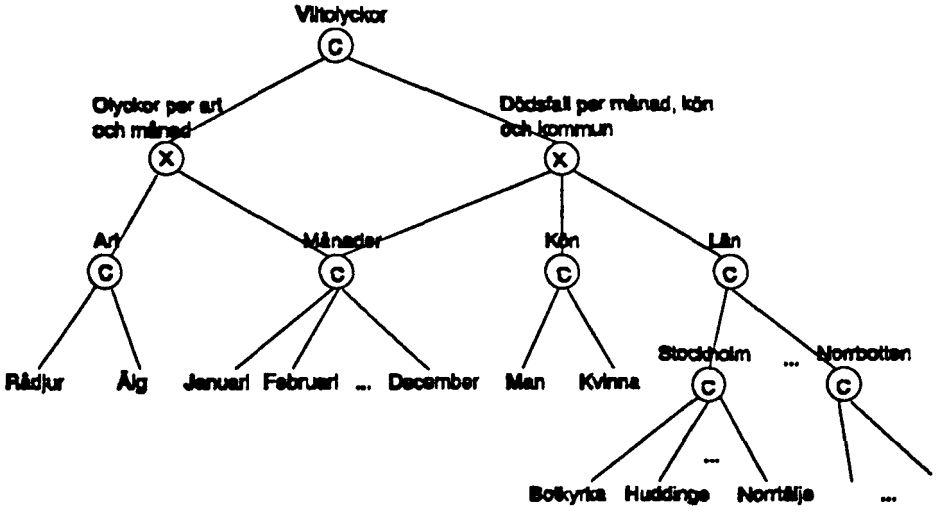
Fördelarna med GUIDE är möjligheten att bygga upp en fråga med hjälp av delfrågor, att man kan titta på dessa delresultat för att bestämma om det var detta man ville ha, att det finns hjälp att komma ihåg attributnamn och dess kodningar, att det finns goda möjligheter att själv styra vilka delar av modellen som är synliga och att det finns en mängd olika sätt att presentera sitt resultat på. En nackdel som finns är att processen att konstruera en tabell är uppdelad i tre faser: val av ämnesområde, manipulation av modellen och frågekonstruktion. Det är naturligt att ha val av ämnesområde som en separat fas, men det finns anledning att tro en användare kommer att vilja manipulera modellen även under frågekonstruktionen, t.ex. om man upptäcker att man behöver en större radie, och man blir då tvungen att gå ur frågedelen, gå in i manipulationsdelen, göra ändringar och gå tillbaka till frågedelen. En annan nackdel med GUIDE är att menyerna inte alltid kommer upp där man förväntar sig att de ska komma.

### 5.1.2. Grafer och träd

En databas kan ses som en hierarkisk samling data, som kan beskrivas i en trädstruktur. De element som finns i trädet kallas noder, och start- respektive slutnoderna kallas rot respektive löv. Vi betraktar själva databasen som rot i trädet, och under den så kommer en indelning i olika ämnesområden, som kan sträcka sig nedåt i flera nivåer. Efter ämnesområdena kommer det ett antal objekt, dessa har i sin tur olika attribut, som kan anta olika värden. En statistisk databas innehåller, förutom de ovan nämnda sakerna, även grupperingar av värden samt föraggregerade tabeller. Att använda denna typ av uppdelning av en databas för att hjälpa en användare att konstruera tabeller görs i bl.a. [Chan 81] och [Rafanelli 90].

I [Chan 81] och görs en indelning av noderna i två olika kategorier, cluster nodes (C-nod) och cross product nodes (X-nod). Skillnaden mellan dessa är att en C-nod kan anta ett av de värden som befinner sig under den i trädhierarkin, medan varje värde en X-nod antar är en beskrivning av en situation som karaktäriseras av ett element ur vart och ett av dess barn. Av de ovan nämnda noderna så tillhör de föraggregerade tabellerna X-noderna, medan filerna, variablerna och grupperingarna tillhör C-noderna. Löven i trädet, alltså de värden som kan antas, betecknas ej som noder i [Chan 81], utan de är helt enkelt löv. I stora databaser så är det vanligt att variabler förekommer på flera olika ställen, och om man inte tar hänsyn till detta så kommer det att finnas samma struktur på flera olika ställen i trädet och denna struktur kommer då att lagras flera gånger, vilket kräver extra lagringsutrymme. Detta problem kommer man runt genom att tillåta att flera

noder delar på en nod på lägre nivå. Det är även värt att notera att det finns en semantisk skillnad mellan X-noder och C-noder. Denna skillnad består av att om man väljer en del av en X-nod så aggregerar man över de delar som ej är valda, medan en vald del i en C-nod används för att begränsa sig till objekt som har denna egenskap.




**Figur 5.2.** Fiktivt exempel på en graf över tillgänglig statistik för viltolyckor

För att illustrera det som har sagts ovan så förklaras Figur 5.2 i detta sammanhang. I Figur 5.2 finns det två stycken X-noder (tabeller), den ena innehåller antalet viltolyckor per art och månad, den andra antalet dödsfall per månad, kön och kommun. Den delade noden *Månader* är desamma i de båda tabellerna, d.v.s. de anger samma tidsindelning. Om de ej delat på denna nod så hade det varit tvunget att ha uppdelningen i månader på två ställen i trädet, vilket hade lett till att den hade blivit större och mindre överskådlig. Noden kallad *Län* är en nod som innehåller en gruppering av kommunerna i län, även detta ger en bättre överblick eftersom man lätt hittar vilket län man är intresserad av och kan sedan leta vidare på kommunnivå.

Det finns även ett exempelsystem beskrivet i [Chan 81], detta system kallas SUBJECT. Detta baserar sig på de ovan beskrivna nodtyperna och grafstrukturen, men det gör en ytterligare indelning av noderna i ämnesnoder, filnoder, datanoder och slutnoder. Systemet är avsett att användas för att konstruera frågor till statistiska databaser. Det sätt på vilket man här visualiserar sin modell är genom menyer som ändras beroende på var i grafen man befinner sig. Från början så befinner man sig längst upp i grafen och får då upp de alternativa riktningar som man kan röra sig i. När man väljer ett alternativ så ändras skärmen till att visa vad som finns under den valda noden. För att kunna välja noder, göra snabbare förflyttningar i grafen, m.m. så finns det sju stycken kommandon, som samtliga kan förkortas till en bokstav, meningen är att avancerade användare inte ska behöva gå den långa vägen varje gång, utan det skall finnas genvägar som snabbar upp arbetet.

Ett alternativt sätt att använda en trädstruktur för databaser finns beskrivet i [Rafanelli 90]. Där delas noderna in i sex olika typer och det definieras även regler för hur dessa noder kan knytas samman samt hur man ska manipulera ett träd för att bygga upp en fråga till databasen, tillsammans så definierar dessa tre delar en GRASS* graf. Jag kommer ej att beskriva de olika delarna i GRASS* här, utan istället så inriktar jag mig på de delar som ingår i det system för frågekonstruktion som beskrivs i artikeln, och i vilket GRASS* används för att visualisera vilken information som databasen innehåller.

Systemet består av ett antal olika fönster som alla visar någon aspekt av databasen eller frågan. Det första fönstret är *GRASS* fönstret*, i detta visas de olika nivåerna i databasen med hjälp av en GRASS* graf. Det är i detta fönster som det mesta av arbetet sker med konstruktionen av frågan, frågan specificeras genom att manipulera grafen på olika sätt och visa olika delar av den. Det finns även ett *informations fönster*, här visas information om den nod som är vald i GRASS* fönstret. I *instans fönstret* visas de olika värden som en viss nod kan anta. För att användaren hela tiden ska kunna få en bra bild av hur resultatet kommer att se ut så finns det två fönster som visar detta, dessa är *utskriftsformat fönstret* och *utskrifts fönstret*. Dessa två fönster är i grunden lika, de visar både ett tabellskelett med förspalt och överspalt. Skillnaden mellan dem är att det förra bara visar i vilken ordning som olika variabler visas i förspalt och överspalt, medan det senare även visar fördelningen av värden. Den sista delen är *STAQUEL fönstret*, i detta visas den fråga som har byggts upp, omskriven till språket STAQUEL.

Det finns två olika sätt att manipulera GRASS* grafen på, antingen så använder man kommandon från de menyer som finns eller så väljer man ett kommando från en kommandopalett och applicerar sedan detta. De typer av manipulation som är möjlig är t.ex. att bilda nya noder genom att slå samman befintliga noder och att välja att inkludera en nod i frågan.

### 5.1.3. Naturliga Språk

Naturliga språk har fördelen att användaren ej behöver lära sig en ny syntax för att konstruera frågor mot databasen. Detta gör att naturliga språk är mest lämpat för användare som känner till ett specifikt problemområde väl, men som ej har någon djupare kunskap om datorer och frågespråk. En erfaren användare uppfattar naturliga språk som oprecisa och "pratiga" vid en jämförelse med vanliga kommandospråk.

I [Schneiderman 87] beskrivs två system där naturliga språk har använts för att ställa frågor till databaser. Det första systemet kallas INTELLECT, och där får användaren upp en prompt där man ska formulera sin fråga. När datorn har tolkat användarens fråga så skriver den ut sin tolkning tillsammans med svaret på frågan. Den tolkning som datorn skriver ut är formulerad på ett vis som mer liknar kommandospråk är naturliga språk. Tanken med att visa denna tolkning är dels att man ska få möjlighet att kontrollera om datorn har tolkat rätt, och dels att influera användaren att skriva sina frågor på datorns mer strukturerade och precisa form.

Det andra systemet som beskrivs i [Schneiderman 87] är NaturalLink, detta system använder en kombination av naturliga språk och menyer för att konstruera frågor mot databasen. Skärmen är indelad i flera mindre delar (menyer), och i varje del finns en uppräkning av giltiga ord och fraser som kan användas i frågor. En fråga specificeras genom att man successivt väljer olika ord och fraser från menyerna. Fördelarna med systemet är att man direkt får en fråga som är semantiskt och syntaktiskt korrekt, samt att även användare utan vana av tangentbord snabbt kan sätta ihop en fråga.

Ytterligare ett system som baseras på naturliga språk är LIDS86 (Land Information Dictionary System developed in 1986), som beskrivs i [Sato 88]. LIDS86 är egentligen en utvidgning av ett tidigare system som enbart bestod av en metadata browser, med vars hjälp användaren fick leta sig fram till den information som eftersöktes. Problemet med detta tidiga system var att en användare var tvungen att förstå datamodellen innan det blev möjligt att få fram någon information ur systemet. Dessa problem var en av orsakerna till att systemet vidareutvecklades, och att man lade till ett frågesystem baserat på naturliga språk. När LIDS86 får en fråga ställd till sig så börjar det med att göra en grammatisk analys av frågan, den bryter då ner frågan i dess beståndsdelar och ser efter om den överensstämmer med grammatiken i systemet. Efter den grammatiska kontrollen undersöker LIDS86 om det finns några tvetydigheter i frågan, och om det finns det så får användaren precisera vilken information som efterfrågas. När alla oklarheter är utredda använder LIDS86 sig av det tidigare systemets metadatabas för att undersöka om den eftersökta informationen finns tillgänglig, om den ej finns det så söker LIDS86 efter annan information som nästan uppfyller de krav som eftersöktes i frågan. Den information som LIDS86 fanns i sökningen i metadatabasen presenteras för användaren och användaren får besluta om det är denna information som efterfrågades. När användaren till slut har accepterat en viss information hämtar LIDS86 fram den.

Ett exempel på en session med LIDS86 hämtad ur [Sato 88]:

LIDS: Welcome to LIDS86. What do you want ?
USER: I need number of employees by sex in Tokyo at 1975.
LIDS: Do you mean Tokyo-prefecture by "Tokyo" ?
USER: No.
LIDS: You mean Tokyo-city, don't you ?
USER: Yes.
LIDS: <data about employees>
We have data such as number of employees by TIME-POINTS-10 by CITY AND PREFECTURE by MAJOR-INDUSTRIAL-CLASSIFICATION-A. But it is not categorized by sex. Is it OK ?
USER: No.
LIDS: <data about employees>
We have data such as number of employees by TIME-POINTS-10 by CITY AND PREFECTURE by MAJOR-OCCUPATION-CLASSIFICATION. But it is not categorized by sex. Is it OK ?
USER: What is MAJOR-INDUSTRIAL-CLASSIFICATION-A ?
LIDS: MAJOR-INDUSTRIAL-CLASSIFICATION-A consists of the following categories.

* WHOLE INDUSTRY
* AGRICULTURE
* FORESTRY
...

USER: The first data, please.

LIDS: <data about employees>
We have data such as number of employees by TIME-POINTS-10 by CITY AND PREFECTURE by MAJOR-INDUSTRIAL-CLASSIFICATION-A. But it is not categorized by sex. Is it OK ?

USER: Yes.

### 5.1.4. Kommandospråk

Oftast så är det ej lämpligt att använda de språk som följer med en databas för att konstruera tabeller. Detta beror på att dessa är inriktade på allmän administration av databasen, och de möjligheter som finns att styra hur resultatet på en fråga ska visas är otillräckliga. Det som saknas är främst möjligheten att generera korstabeller, man brukar istället få svaren i form av en lista med alla poster som matchar. Exempelvis så kan man i SQL använda GROUP BY för att gruppera resultatet på olika vis, men svaret blir alltid en lista och ej en tabell. Skälet till att det saknas denna funktionalitet kan vara att man använder vanliga databasfrågor för att undersöka om det finns några poster som passar in på frågan och om så är fallet vill man visa några av dessa posters egenskaper, medan en statistisk fråga främst är intresserad av antalet poster som passar in.

TAB68 är ett traditionellt tabellspråk, det fungerar som ett slags programspråk där man beskriver vilka data man vill ha, i vilka rader dessa ska placeras och hur de ska behandlas. Följande kodavsnitt, som vi ska analysera nedan, är hämtat ur [TAB68], men radnumreringen är ditlagd i efterhand:

```
1.      FILE      VAR=(SEX,1,N2),(INCOME,2,N5),(AGE,7,N3);
2.      GROUP     AGEGROUP(AGE)=1(<26),2(<66),3(>65);
3.      TABLE     COL(1,2)=SEX(1,2);
4.      COL(3,4)=SEX(1,2),SUM(INCOME);
5.      ROW(1:3)=AGEGROUP(1:3);
```

På rad ett definierar man vilka variabler man vill ha med i tabellen, och var dessa ska hämtas ifrån. Rad två delar in variabeln AGE i tre grupper, de under 26 år, de under 66 år, och de över 65 år. Själva tabellen byggs upp på raderna tre, fyra och fem. Rad tre säger att vi ska ha antalet personer av respektive kön i kolumn ett och två. Sedan specificerar rad fyra att kolumn tre och fyra ska innehålla den totala lönesumma för personerna i kolumn ett och två. Slutligen säger rad fem att de tre åldersgrupperna ska utgöra indelningen på raderna.

Det nu analyserade exemplet, trots att det i någon mening nästan var minimalt, var inte helt lättförståligt, särskilt inte om man saknar kunskaper i programmering. Fördelen med att använda tabellspråk är att en van användare snabbt kan göra komplicerade konstruktioner, man har även en mer direkt kontroll över utformningen av tabellen. Den stora nackdelen är

dock densamma som gör tabellspråken bra, nämligen att de är komplexa och svårlärda system byggda för experter. En vanlig användare som bara använder databaser ibland har en mycket hög tröskel att komma över innan man kan producera något, bara att förstå hur man specificerar vilken databas man vill använda är ett problem.

### 5.1.5. Hur data bör kunna bearbetas

I de föregående avsnitten har jag studerat olika lösningar på hur man kan konstruera en tabell utgående från statistiska data, men en brist i detta har varit att det inte har tagits upp så mycket om hur man bör kunna behandla sina data. Den bristen ska detta avsnitt försöka råda bor på. Avsnittet behandlar sätt som man skulle kunna vilja bearbeta sitt datamaterial på, och det grundar sig främst på [Sundgren 85].

En första indelning av dessa funktioner kan göras i relationsalgebraiska operationer och statistiska operationer. Med relationsalgebraiska operationer menas de vanligaste operationer som finns definierade i språk som behandlar relationsdatabaser, dessa är: union, snitt, differens, kartesisk produkt, selektion, projektion, join och division. Statistiska operationer är ofta aggregeringar av olika slag, aggregeringar innebär att man utför en operation på en kolumn. Vanliga typer av aggregeringar är räkning av antalet förekomster, summering, minimum, maximum, standardavvikelse och varians.

Efter att ha genomfört en aggregering så vill man även ofta kunna föra in olika delsummor, gruppera värdena på olika vis eller lägga till nya kolumner. Möjlighet att lägga till nya kolumner kan man använda till att härleda nya data ur sådana som redan finns i tabeller, t.ex. skapa en kolumn för ålder som härleds ur dagens datum och födelsedatum, detta används för att underlätta för den som ska studera tabellen.

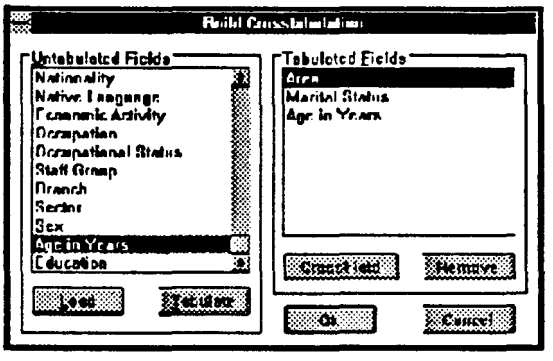
## 5.2. Studie av befintlig programvara

För att få en uppfattning om hur problemet med tabellgenerering tidigare har lösts så har jag undersökt på tre befintliga program, dessa är Supercross, PC-AXIS och GQL. Jag har valt att studera dessa genom att först beskriva hur de olika programmen fungerar och hur man använder dem, sedan har jag gjort en jämförande undersökning av vad som finns i programmen och hur de har löst olika problem.

### 5.2.1. Supercross

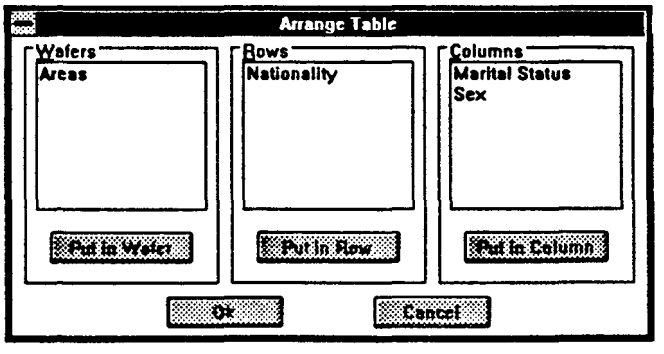
Studiet av Supercross gjordes i en demonstrationsversion. Den databas som fanns tillgänglig var en folkräkningsdatabas över hela Ungerns befolkning, den innehöll data om cirka 11.5 miljoner personer. De data som låg i databasen var mikrodata, som var indexerade på geografiska regioner. Detta ledde till att man kunde få snabba svar om man begränsade sin fråga till ett litet geografiskt område, medan en körning som innefattade hela Ungern tog längre tid.

När man inleder en ny fråga i Supercross så kommer det upp ett dialogfönster i vilket man väljer variabler som ska ingå i tabellen (variabeldialog). De variabler som finns tillgängliga visas i en lista, och man väljer att inkludera en variabel i tabellen genom att markera den och trycka på *Tabulate*, detta resulterar i att ytterligare ett dialogfönster kommer upp på skärmen (värdedialog). I värdedialogen visas den valda variabelns värdemängd och man väljer här om man vill ha med alla värden eller om man vill begränsa sig till några få, det finns även möjligheter att gruppera värden. Vissa variabler hade värdena grupperade i flera nivåer, t.ex. Area som hade 4 nivåer. När man är klar med värdedialogen så trycker man på *OK*, då uppdateras variabeldialogen genom att den variabel man har arbetat med läggs till i listan med *Tabulated Fields*. Ett exempel på hur en variabeldialog kan se ut finns i Figur 5.3, i detta exempel så har variablerna Area, Marital Status samt Age In Years valts ut för att ingå i tabellen.




**Figur 5.3.** Val av variabler till en tabell i Supercross. Area, Marital Status och Age In Years har valts ut för att ingå i tabellen.

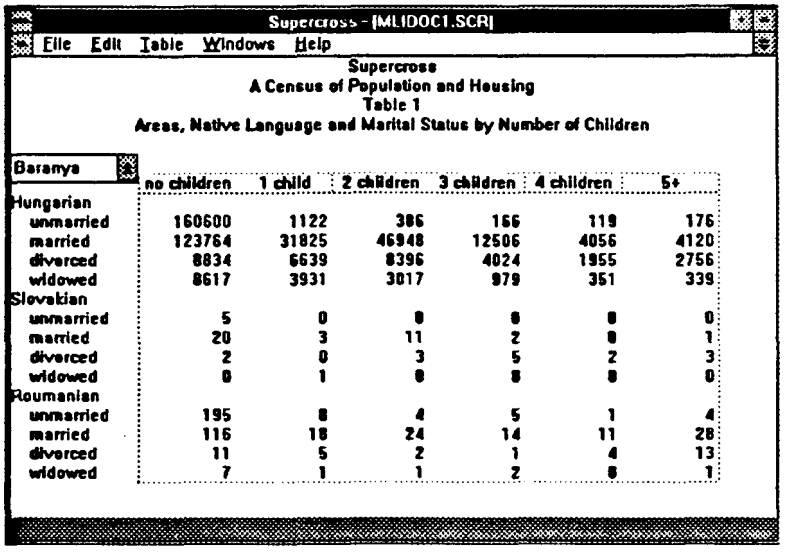
När man har specificerat vad som ska ingå i tabellen så trycker man *OK* i variabeldialogen, så får man fram ett preliminärt utseende på tabellen där Supercross har bestämt om variablerna ska placeras i förspalt eller överspalt. Om man inte tycker att detta utseende är bra så kan man lätt ändra detta genom att välja *Arrange Table* från menyn, man får då upp ett dialogfönster där man kan flytta variablernas placering mellan förspalt, överspalt och wafer, se Figur 5.4. Wafer är en ytterligare dimension i tabellen genom att den bygger upp flera mindre tabeller, en för varje värde som tillhör wafer-variabeln. Att flytta en variabel går till så att man först markerar den, och sedan klickar på knappen för wafer, förspalt eller överspalt.




**Figur 5.4.** Dialogfönster för placering av variabler i wafer, förspalt och överspalt.

Det finns även möjlighet att lägga till, ta bort eller ändra värdemängden för en variabel. Detta sker genom att man får upp samma dialogruta som i inledningen, d.v.s. Figur 5.3.

När man är nöjd med utseendet hos tabellen så väljer man *Cross Tabulate* från menyn för att generera tabellen. Under det att programmet arbetar så visas en markering om hur långt det har hunnit, det finns även möjlighet att avbryta genereringen och spara de resultat som har hunnit bli klara. Figur 5.5 visar ett exempel på hur en tabell kan se ut. I Supercross så kan man även spara tabeller, antingen före eller efter genereringen av värden. Förutom Supercross egna format så kan man även spara en tabell i flera andra format, t.ex. kommaseparerade filer och LOTUS 1-2-3.




**Figur 5.5.** Exempeltabell i Supercross

De möjligheter man har i Supercross att bearbeta en färdig tabell på, med avseende på statistiska funktioner, är ganska begränsad. Den enda funktionen som finns är summering av kolumn eller rad. En möjlighet här är att spara tabellen och importera den i ett annat program, som är bättre på att bearbeta tabeller statistiskt.

Det finns ingen separat rapportfunktion i Supercross utan om man skriver ut så får man det som syns på skärmen. Ett problem när man skriver ut från Supercross är hur man ska behandla wafers, programmet struntar i dessa wafers och skriver bara ut den aktuella tabellen, men det vore även bra om man skulle kunna skriva ut alla deltabellerna på en gång. En liten nackdel med utskriften är också att den använder default-fonter, detta gör att tabellen blir ganska tråkig på papper.

Sammanfattningsvis så är Supercross ett program där man successivt utformar sin tabell genom val i olika listor. Det som är fördelen med Supercross är att det är väldigt enkelt att konstruera själva tabellen, medan nackdelen ligger i att det ej går att göra så mycket med den färdiga tabellen. Det är naturligtvis en nackdel att det tar lång tid att generera data till

tabellen, men det är faktiskt en följd av att man arbetar direkt mot mikrodata. I detta program kan man även se hur tabellen kommer att se ut innan man genererar data till den, detta gör att en ovan användare inte behöver göra om en långdragen tabellgenerering om man gör ett mindre fel i utformningen.

### 5.2.2. PC-AXIS

PC-AXIS är ett program som har utvecklats av SCB för att hjälpa användare att plocka fram olika tabeller. Programmet har ett sorts fönstersystem, men dessa är textuellt baserade och kör direkt i DOS. På SCBs stordatorer används AXIS-systemet för att lagra och presentera statistik, och som namnet antyder så är PC-AXIS ett gränssnitt för PC mot AXIS-systemet. PC-AXIS kan arbeta antingen mot data som ligger lagrade lokalt på PCn eller direkt mot SCBs större datorer. När jag undersökte PC-AXIS så arbetade jag mot en databas som låg på en CD-ROM skiva. Informationen i PC-AXIS ligger i form av makrodata, d.v.s. de är redan aggregerade i någon form. Detta avspeglar sig i att om man gör samma sorts testkörning som i Supercross, fast nu med Sveriges 8.5 miljoner invånare, så behöver man ej vänta alls, utan resultatet kommer upp direkt.

När man ska använda PC-AXIS för att ta fram en viss information så börjar man med att välja ämnesområde, exempel på ämnesområden kan vara befolkning, rättsväsen, arbetsmarknad m.m., dessa presenteras i en lista i vilken man markerar sitt val. När man har valt ett ämnesområde så får man upp en lista på de register som finns tillgängliga inom det området, efter att även här ha markerat ett val så kommer det upp en dialogruta som presenterar de variablerna som finns i det aktuella registret samt deras värdemängder. Här markerar man vilka av variablernas värden som man vill ha med i sin egen tabell, man kan antingen markera enstaka eller välja alla genom att markera själva variabeln. När man är klar och trycker OK så genereras en tabell som innehåller de data man efterfrågade, därefter byter skärmen helt utseende och man kommer till en bearbetningsbild (Figur 5.6).

I bearbetningsbilden kan man utföra beräkningar och redigeringar av den tabell man precis har skapat, på skärmen visas hur tabellen ser ut och de olika kommandona finns tillgängliga via menyer. Själva utseendet på tabellen, d.v.s. placeringen av variablerna i förspalt och överspalt, är från början bestämd av programmet. De funktioner som finns för att bearbeta tabellen är de vanliga aritmetiska operationerna (+, -, * och /), summeringar och omvandling av tabellvärdena till procent eller promille. De aritmetiska operationerna finns i två versioner, dels som funktioner mot vanliga tal som matas in från tangentbordet, och dels som funktioner mot andra tabeller. Alla de ovan angivna funktionerna kan utföras antingen så att både original värdena och de beräknade värdena visas i tabellen eller så att bara de beräknade värdena visas. I PC-AXIS finns även funktioner för att slå samman två tabeller med samma innehåll till en tabell. De möjligheter som finns tillgängliga för en användare att förändra en tabells utseende är: byta en variabels placering, byta den ordning i vilken värden visas och editera tabellrubrikerna. En nackdel med den funktion som finns för att flytta variabler mellan förspalt och överspalt är att man alltid måste specificera

placeringen för samtliga variabler, alltså även de vars placering man ej vill förändra, samma problem finns även vid förflyttningar av värden.

```
 Calculate  Arithmetic  Arithmetic_(table)  Edit  Show  Save  Quit
+BE007-----------------------------Processing-------------------------BE007.PX-+
¦Foreign citizens by region, citizenship, sex, time and age.                   ¦
¦                                                                              ¦
¦                                  1990-12-31                                  ¦
¦                                       45-49      50-54      55-59      60-64 ¦
¦                                                                              ¦
¦Sweden     Albanien         Men            1          0          0          0 ¦
¦                            Women          0          0          0          0 ¦
¦           Andorra          Men            0          0          0          0 ¦
¦                            Women          0          0          0          0 ¦
¦           Belgien          Men           10          9          7          2 ¦
¦                            Women         10         10          8          3 ¦
¦           Bulgarien        Men           32         12          8          4 ¦
¦                            Women         28         11         10         17 ¦
¦           Danmark          Men         1340        897        897        946 ¦
¦                            Women        893        686        574        592 ¦
¦           Finland          Men         5737       4276       3065       1964 ¦
¦                            Women       5304       3706       2654       1922 ¦
¦           Frankrike        Men          140         65         34         35 ¦
¦                            Women         80         53         41         23 ¦
¦           Grekland         Men          244        261        169         92 ¦
¦                            Women        119        116        110         87 ¦
¦F1=Help F3=Exit F5=Undo F7=Footnote F10=Menu       [ • ]  [ • ]  [ - ] [  ]   ¦
+------------------------------------------------------------------------------+
```

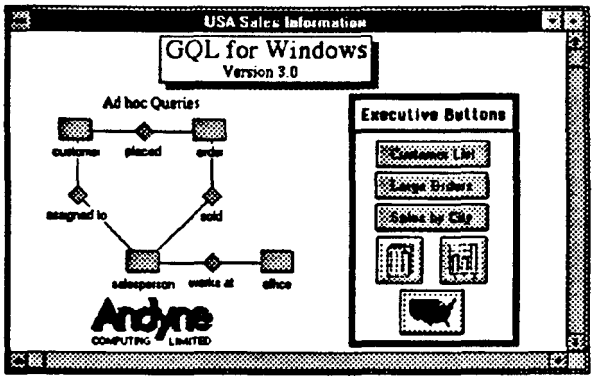
**Figur 5.6.** Bearbetningsbilden ur PC-AXIS

En funktion som PC-AXIS är ensam om bland de tre undersökta programmen är möjligheten att visualisera en tabell genom diagram. De diagramtyper som man kan välja mellan är linjediagram, stapeldiagram och cirkeldiagram, det fanns möjlighet att visa dessa diagram antingen på skärmen eller på skrivaren. PC-AXIS har även goda möjligheter att spara i olika format, det går t.ex. att direkt spara i format som kan läsas av EXCEL, LOTUS och dBASE. Andra påpekanden man kan göra är att det finns en undo-funktion som backar ett steg, men det finns ej någon redo-funktion för att gå tillbaka igen. Det finns även ett hjälpsystem online som täcker det viktigaste. Utskriftsmöjligheterna för tabeller tillhandahåller ej några formateringsmöjligheter, utan tabeller skrivs helt enkelt ut som den ser ut på skärmen. Detta är ej någon större nackdel då PC-AXIS har goda exportmöjligheter till andra program, som kan ordna snyggare rapporter.

Allmänt så kan man alltså beskriva PC-AXIS som ett system där val görs m.h.a. listor och där det finns goda möjligheter att manipulera tabellerna efter att man har genererat dem.

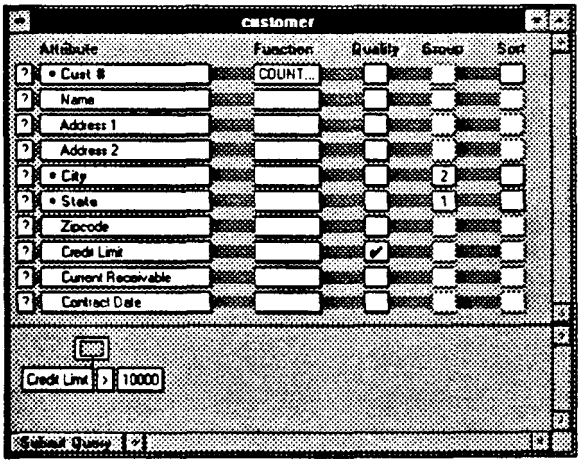
### 5.2.3. Graphical Query Language (GQL)

Jag har undersökt en demonstrationsversion av GQL/User V3.0 för Windows, där databasen bestod av fiktiva affärsdata. GQL är ett program där man konstruerar tabeller och SQL-frågor genom att manipulera grafiska strukturer. GQL är ej ett statistiskt program, men det inkluderas i jämförelsen eftersom det har ett intressant gränssnitt.




**Figur 5.7.** GQLs öppningsbild

När man startar GQL så får man först välja vilken applikation man vill använda, i demoversionen fanns det bara en tillgänglig applikation, när man har gjort detta kommer det upp ett fönster som beskriver applikationen m.h.a. en grafisk modell. Det finns tre olika typer av objekt i en modell: dataobjekt, relationer och kommandoknappar. I den applikation som fanns tillgänglig så bestod modellen av fyra olika dataobjekt relaterade till varandra med fyra relationer, samt sex stycken knappar för olika ändamål (se Figur 5.7). Interaktionen mellan användaren och modellen sker genom val med musen, dubbelklickning på ett dataobjekt resulterar i ett attributfönster, och enkelklickning på en knapp kör igång en fördefinierad fråga. Relationerna används för att kombinera variabler som finns i olika objekt. När man ska använda ett samband mellan två objekt i en fråga så börjar man med att konstruera en fråga i ett objekt, sedan markerar man relationen mellan objekten, och slutligen så dubbelklickar man på den andra deltagaren i relationen och väljer där vilka attribut från detta objekt som ska läggas till de poster som blir resultatet av frågan i det första objektet. Slutligen så trycker man på *Submit Query* för att visa resultatet.




**Figur 5.8.** Exempel på attributfönster för dataobjektet customer

Det är i attributfönstret som all specificering av en fråga sker. Det går till så att man markerar vilka attribut som ska visas genom att klicka på dem. Om man vill att attributen ska bearbetas på något vis väljer man en funktion i fältet *Function*, vill man lägga villkor på en fråga används *Qualify*, gruppering av data sker med hjälp av *Group* och sortering med hjälp av *Sort*. Figur 5.8 innehåller ett exempel på ett attributfönster, i detta har man valt att visa *Customer #*, *State* och *City* för de kunder som har en *Credit Limit* som är större än $10.000 samt att de ska visas grupperade på först *State* och sedan *City*.

När man har genererat en fråga och fått upp svaret så kan man välja att skapa en rapport. Grundutseendet på denna är en rubrik, datumangivelse i hörnet, samt då själva tabellen. När man har fått upp detta första utkast till rapporten så kan man själv editera texten, ställa in olika fonter samt lägga till delsummor på kolumnerna. Om man sedan sparar den fråga som genererade tabellen och knyter frågan till en knapp, så kommer man varje gång man trycker på knappen, d.v.s. när man ställer frågan, att få som svar en rapport formaterad på det sätt man hade valt. Observera att GQL ej genererar korstabeller, utan bara vanliga listor. En fördel med detta program är att man kan få se sin fråga i SQL format.

Sammanfattar man GQL så är det ett system där man grafiskt markerar vilka objekt som ska ingå i frågan, även val av ingående variabler samt hur dessa ska grupperas, selekteras m.m. sker genom manipulation med musen. Detta urval uppfattar jag som lättförståligt. När man har fått fram ett resultat så har man stora möjligheter att manipulera dess grafiska utformning, men man kan ej ändra själva tabellens utseende på samma sätt som i t.ex. PC-AXIS.

## 5.3. Diskussion

Under den litteraturstudie och programundersökning som har redovisats ovan har jag skapat mig en uppfattning om hur man kan använda sig av olika interaktionsmodeller samt vilka funktioner som bör finnas i ett - program för tabellkonstruktion. Jag tänkte därför här dra fram de fördelar och nackdelar som jag ser med olika lösningar.

### 5.3.1. Jämförelse mellan de undersökta programmen

I detta avsnitt tar jag upp de viktigaste egenskaperna för Supercross, PC-AXIS och GQL i en jämförelse dem emellan. Jämförelsen sammanfattas i form av en tabell i slutet av avsnittet.

En av de saker som skiljer programmen åt är valet av gränssnitt mot användaren, GQL använder sig av en grafisk modell, medan Supercross och PC-AXIS använder en sorts menybaserad modell. Supercross och PC-AXIS arbetar mest med listor i olika utformningar, men då grundprincipen är densamma som för menyer, d.v.s. man väljer ett eller flera alternativ från en uppräkning av möjliga val, så kallar jag helt enkelt dessa listor för menyer. Programmen har även olika stora möjligheter att kombinera uppgifter från olika register. I GQL använder man den grafiska modellen för att markera

om man vill använda en koppling mellan två register, för att på så sätt samköra dessa, denna möjlighet finns varken i Supercross eller PC-AXIS. Att samkörningsmöjligheter saknas i Supercross beror på att programmet är avsett att användas med endast ett register, avsaknaden i PC-AXIS beror på att PC-AXIS arbetar med makrodata, d.v.s. redan aggregerade data, och att det därför inte finns någon möjlighet att para ihop poster i olika register. I PC-AXIS finns det möjlighet att slå ihop tabeller med samma struktur, detta kan ses som en sorts samkörning på makronivå.

Om man jämför möjligheterna att bearbeta data i och omforma utseendet på färdiga tabeller, så framstår PC-AXIS som det bästa alternativet för bearbetningar medan PC-AXIS och Supercross är jämbördiga när det gäller att omforma utseendet på tabellerna. Att PC-AXIS är att föredra för databearbetning beror på att det innehåller flest funktioner, dessa är: summeringar, omvandlingar till procent och promille, aritmetiska operationer samt möjligheter att slå ihop tabeller. Något som saknas i alla tre programmen är möjligheten att direkt göra enkla statistiska bearbetningar, som t.ex. medelvärde och standardavvikelse. Funktionerna för att omforma färdiga tabellen var ungefär desamma i både PC-AXIS och Supercross, de kunde flytta variabler, byta ordning på värden samt gruppera värden, men de löstes på ett smidigare sätt i Supercross. I Supercross kan man byta placering på enstaka variabler genom att flytta dem mellan olika listor, i PC-AXIS är man tvungen att specificera alla variablers placering varje gång man vill byta placeringen på en av dem, och detta ska göras utan att programmet informerar om var de är placerade innan omflyttningen. Detta gör att man kan behöva göra om en omflyttning flera gånger i PC-AXIS innan resultatet blir det man vill ha. Den funktion som finns i GQL för att ändra variablernas placering tillåter bara att man byter ordningen i vilken de visas i förspalten, det finns ingen överspalt att flytta om i. Den omarbetnings-funktion som GQL är ensam om att tillhandahålla är att byta typsnitt och storlek på tecknen, detta gör att GQL kan producera rapporter direkt, medan tabeller från Supercross och PC-AXIS måste exporteras för att kunna göra en snygg rapport. PC-AXIS har en inbyggd diagramfunktion som är ganska lättanvänd och praktisk när man är intresserad av att visa någon speciell aspekt av en tabell.

Ytterligare några viktiga funktioner är möjligheterna att spara och ladda tabeller, att få hjälp och att kunna ångra val. I alla tre programmen kan man spara sina tabeller och även ladda in dem igen. PC-AXIS skiljer sig lite åt när det gäller att ladda in en tabell genom att det ej går att få upp exakt samma tabell direkt, detta beror på att när man spara en tabell i PC-AXIS så läggs den med i listan över tillgängliga tabeller som man kan arbeta med, så när man vill ladda in en tabell så väljer tabellen och väljer sedan att visa alla variabler och värden. Hjälpfunktionen i PC-AXIS är bra, och den känner även av i vilket sammanhang som man bad om hjälp och visar den hjälpbild som är förklarar just denna situation. Supercross hade ej hjälp i demoversionen, men det fanns förberett i menyerna, så det finns antagligen i den riktiga versionen. I PC-AXIS kunde man ångra senaste kommandot i bearbetningsbilden, men det var ej möjligt att ångra ångra-kommandot.

Avsaknaden av ångra-funktioner i Supercross förvånade mig då detta program annars är mycket bra. GQL saknade även det helt ångra-funktioner.

| | Supercross | PC-AXIS | GQL |
|---|---|---|---|
| Typ av data i databasen | mikrodata | makrodata | mikro/makro |
| Typ av interaktion | menyer | menyer | grafisk |
| Preview på tabellen | Ja | Nej (1) | Nej (1) |
| Bearbetning av tabeller | Dåliga | Bra | Dåliga |
| Summeringar | Ja | Ja | Ja |
| % | Nej | Ja | Nej |
| Statistiska bearbetning (4) | Nej | Nej | Nej |
| +, -, * och / | Nej | Ja | Nej |
| Slå ihop tabeller | Nej | Ja | Nej |
| Flytta variabler | Ja | Ja | Ja |
| Flytta värden | Ja | Ja | Nej (2) |
| Gruppera värden | Ja | Ja | Nej |
| Grafisk utformning | Nej | Nej | Ja |
| Spara/Ladda | Ja/Ja | Ja/Ja | Ja/Ja |
| Online hjälp | Ja (3) | Ja | Nej |
| Undo/Redo | Nej/Nej | Ja/Nej | Nej/Nej |
| | | | |
| (1) Behövs ej om makrodata | | | |
| (2) Ej statistiskt program, så funktionen behövs ej | | | |
| (3) Men det fanns ej hjälp i demoversionen | | | |
| (4) T.ex. medelvärde och varians | | | |

Tabell 5.2. Jämförelse mellan programmen

### 5.3.2. Interaktionsmodeller

Den typ av datamodell som förekom oftast i litteraturen var att bygga upp en trädstruktur, med hjälp av olika sorters noder. Om man använder denna modell även för interaktionen med användaren, genom att rita upp trädet på skärmen, så anser jag att det kan uppstå svårigheter för de som inte använder systemet dagligen. Denna slutsats drar jag av att det finns så många olika sorters noder, t.ex. så använder GRASS* sex olika typer av noder, som alla representerar olika strukturer och när man manipulerar dem så har de olika semantisk betydelse. En annan nackdel med graferna är att de blir stora och svåra att överblicka om man skulle modellera stora och komplexa databaser. Att använda sig av menyer för att visualisera trädstrukturen, som det görs i SUBJECT, tror jag leder till långsamma förflyttningar i trädet, speciellt när man ska förflytta sig horisontellt, eftersom man blir tvungen att gå upp i trädet och sedan ner igen. Om man använder menyer så är det i och för sig möjligt att använda snabbkommandon, men för att komma ihåg dessa genvägar så krävs det att man använder programmet ofta.

När det gäller naturliga språk så är mina största invändningar att det är mycket för användaren att skriva innan det nås några resultat och att man hela tiden måste precisera vad som egentligen menas, det är alltså en långsam process om användaren inte är van att använda tangentbordet. För

att slippa tvetydigheter så är det möjligt att presentera de giltiga alternativen i varje frågedel för användaren, detta sätt användes i NaturalLink, den nackdel som jag kan se med detta är att när informationsmängden blir stor så behövs det mycket utrymme för att kunna visa alla valmöjligheter. Jag tycker även att det då kan ifrågasättas om det fortfarande är ett naturligt språk om man visar alla möjliga alternativ, utan att det egentligen är ett sorts menybaserat system. Fördelen med naturliga språk är givetvis att en användare inte behöver veta precis vad som finns lagrat i databasen, utan det är möjligt att börja med en allmän fråga och sedan låta datorn hjälpa till att hitta rätt i den information som finns lagrad. Det är också fördelaktigt att ovana användare lätt kan komma igång utan att behöva lära sig någon ny syntax.

Kommandospråk är de naturliga språkens motsats. De är kortfattade och precisa, men kräver kunskap om språkets syntax samt om vad som finns lagrat i en databas och hur detta kodas. På grund av dessa två nackdelar så är det inte möjligt att använda kommandospråk i system som ska kunna användas av personer som inte har så stor datorerfarenhet.

Metoden att använda menyer för visa vilken information som finns i databasen användes främst i två av de undersökta programmen, nämligen Supercross och PC-AXIS. I båda programmen tycker jag att det gick väldigt bra att konstruera de tabeller man ville ha, listorna presenterade på ett bra sätt vilka alternativ som fanns. Supercross var nog ett snäpp bättre än PC-AXIS, men jag tror att det kan bero på att Supercross kunde utnyttja de möjligheter som Windows ger att ha flera fönster synliga samtidigt, medan PC-AXIS begränsades av sina textfönster.

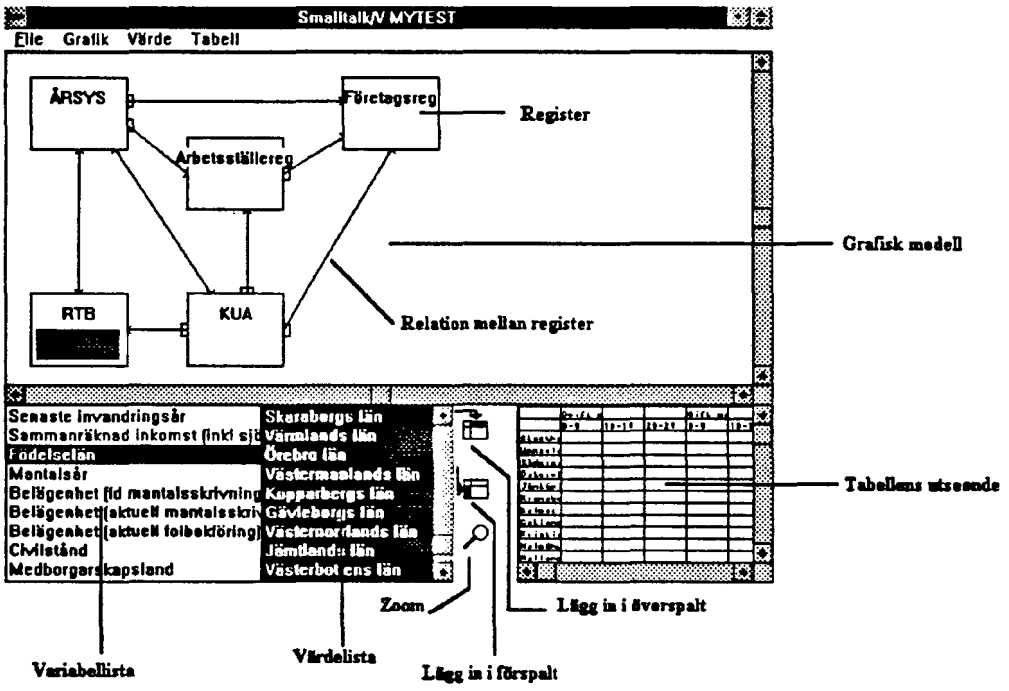
Den sista interaktionstypen som jag har behandlat är att använda en grafisk modell för att visualisera vilken information som finns tillgänglig i en databas. Denna modell förekommer på tre olika ställen: [Malmborg 88], GUIDE och GQL. Alla dessa tre använder den grafiska modellen för att visa vilka informationsmängder som finns och hur de relaterar sig till varandra, men när en användare ska studera variabler och deras värdemängder så presenteras dessa i någon form av meny eller lista. Att på detta vis kombinera grafisk modell med menyer tycker jag ger en lätthanterlig modell som ger användaren en god överblick över vad som finns i databasen.

# 6. Presentation av min prototyp

Det gränssnitt som jag har valt att implementera har de idéer som finns i [Malmborg 88] som grund. Jag har valt att använda denna ansats då jag tycker att den på ett bra sätt visar vilken information som finns tillgänglig, och hur informationen relaterar sig till varandra.

## 6.1. Beskrivning av prototypen

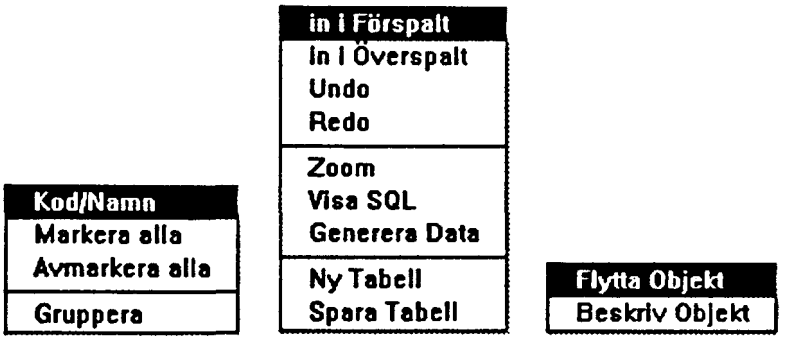
I mitt program arbetar man bara i ett enda fönster, men det finns även ett fönster där man passivt kan inspektera hur tabellen ser ut. I det första fönstret, huvudfönstret, presenteras den information som finns tillgänglig i databasen, de olika registren och deras kopplingar visualiseras genom ett ER-diagram och de variabler och värden som hör till dessa visas i listor. En av huvudideerna i [Malmborg 88] är att man hela tiden ska kunna följa tabellens uppbyggnad, för att åstadkomma detta så finns det ett område, i nedre högra delen av huvudfönstret, som visar tabellen utseende. Huvudfönstret utseende och de olika delarnas placering visas i Figur 6.1.




**Figur 6.1.** Huvudfönstret i mitt tabellkonstruktionsprogram.

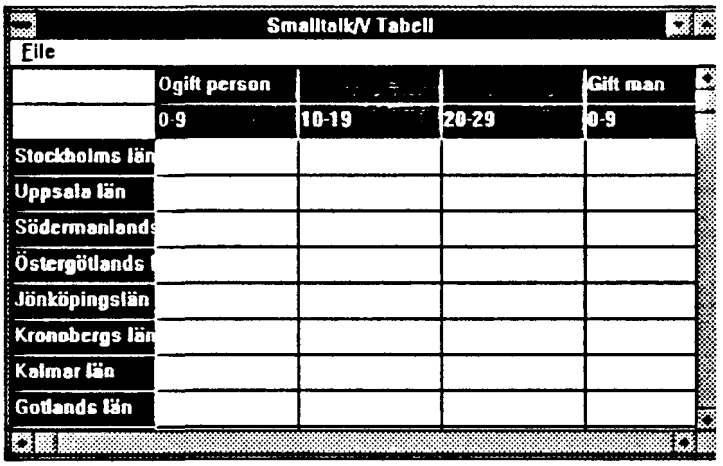
Att konstruera en tabell i mitt program sker i fyra steg för varje mängd av värden som ska ingå i tabellen, dessa är: välja aktuellt register, välja variabel, välja vilka värden från variabeln som ska ingå och placera dessa värden i förspalt eller överspalt. För att välja att arbeta med ett visst register så markerar man det genom att klicka med vänster musknapp i den rektangel som representerar registret, resultatet av detta blir att rektangeln markeras som aktuell och de variabler som ingår i registret visas i variabellistan. När man sedan väljer ett av alternativen i variabellistan så visas de värden som

variabeln kan anta, d.v.s. dess värdemängd, i värdelistan. I värdelistan kan man välja flera alternativ samtidigt och när man har valt alla de värden från variabeln som ska vara med i den tabell man konstruerar, så väljer man *in i Överspalt* eller *in i Förspalt* från *Tabellmenyn*, alternativt så använder man någon av de två snabbknappar för detta som finns till höger om värdelistan.




**Figur 6.2.** Värdemenyn (t.v.), Tabellmenyn (mitten) och Grafikmenyn (t.h.).

Under det att man konstruerar tabellen så kan användaren vilja undersöka mer noggrant hur tabellen ser ut, det syns inte så bra i det lilla området i huvudfönstret, för att göra detta väljer man då *Zoom* från *Tabellmenyn* eller snabbknappen för *Zoom*. Resultatet av detta blir att det visas ett nytt fönster som innehåller den tabell man har konstruerat, men nu mer överblickbar än förut. Om man vill så kan man behålla detta fönster öppet, tabellen kommer då att uppdateras även i detta fönster när man lägger in nya värden i huvudfönstret.



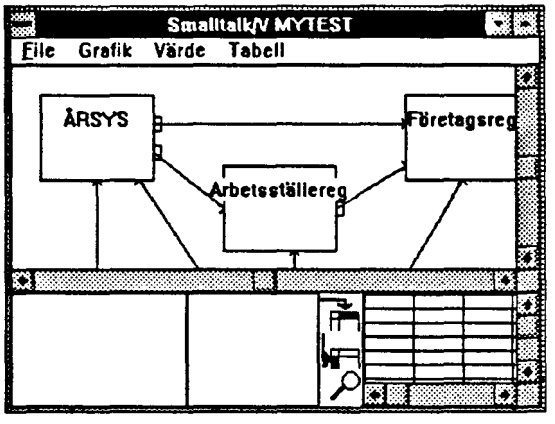

**Figur 6.3.** Resultatet av zoom

En finess som inte fanns i något av de undersökta programmen, men som jag tillhandahåller i mitt program, är multipelt undo och redo. Dessa två funktioner avser tabellens uppbyggnad, det går alltså inte att ångra markeringar i den grafiska strukturen eller i någon av listorna. För att ångra en inläggning av värden i tabellen väljer man *Undo* i *Tabellmenyn*, programmet tar då bort den senaste inläggningen av värden i tabellen och uppdaterar tabellens utseende för att överensstämma med detta. Det är även

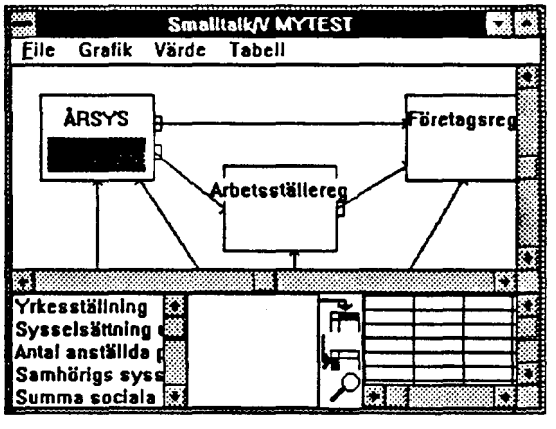
möjligt att välja *Undo* flera gånger i rad, detta innebär att man successivt backar genom konstruktionen för att till slut komma fram till en tom tabell. Det finns även möjligheter att ångra sina ångra kommandon, detta sker med *Redo* från *Tabellmenyn*. Innebörden med *Redo* är att man åter lägger in de senast borttagna värdena i tabellen. Även *Redo* kan utföras flera gånger efter varandra, och även blandat med *Undo*. Effekten av detta blir att det är möjligt att vandra fram och tillbaka genom inläggningarna och se vad som sker i varje steg.

## 6.2. Exempel på session

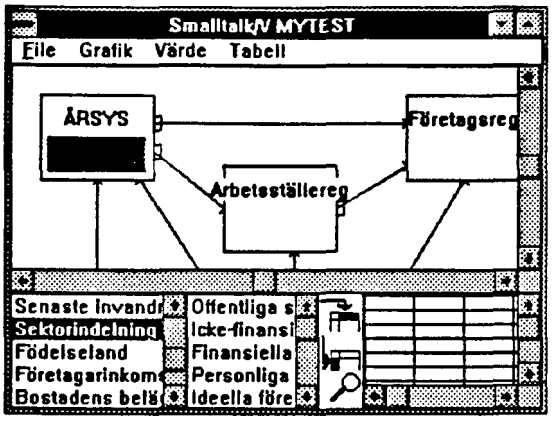
När man startar programmet ser fönstret ut som i Figur 6.4. I Figur 6.5 har registret ÅRSYS valts som det aktuella registret.



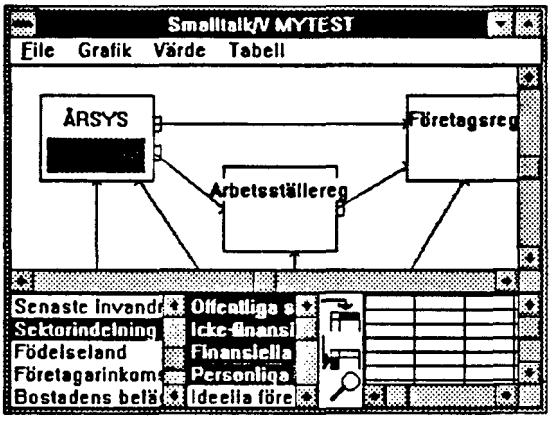

**Figur 6.4.** Som fönstret ser ut när man startat programmet.



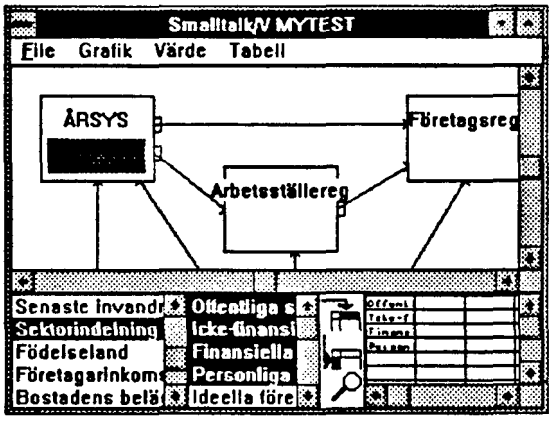

**Figur 6.5.** ÅRSYS valt som aktuellt register.




**Figur 6.6.** Variabeln *Sektorindelning* vald som aktuell variabel.




**Figur 6.7.** Fyra värden har valts att ingå i tabellen.




**Figur 6.8.** Värdena inlagda i tabellens förspalt.

## 6.3. Implementationen av prototypen

Implementationen av prototypen har bestått av två delar, dels att skriva om spread-sheet funktionen som används för att visa tabellerna och dels att bygga själva prototypen. Tidsfördelningen mellan dessa har ungefär 50:50.

Det gick åt förvånansvärt mycket tid till att skriva om spread-sheet funktionen, som är döpt till CellPane, detta berodde dels på att jag var tvungen att sätta mig in i hur Windows hanterar grafiska objekt och dels på att det var svårt att få Smalltalk/V att göra precis det som jag ville ha. De två största problemen med Smalltalk/V var att få CellPane att uppdatera sig automatiskt och att kunna använda scrollbars för att förflytta sig i en CellPane. När man ser tillbaka på omskrivningen så kan jag konstatera att jag inte trodde att det skulle ta så lång tid som det tog.

Själva byggandet av prototypen fortskred ganska snabbt. Den grafiska modellen fanns i stort sett klar sedan tidigare, och krävde bara lite ändringar för att få att fungera som önskat. De listor som finns i prototypen fanns som färdiga byggstenar i Smalltalk/V och krävde därför inte någon tankemöda. Även knappar innehållande bitmappar och all menyhantering fanns färdigt i Smalltalk/V. Spread-sheet funktionen har jag som sagt till stora delar skrivit själv.

Den första versionen av programmet innehöll en modell bestående av tre objekt, som låg som kodat i programmet, och det var möjligt att välja olika variabler och värden och placera dem i tabellen. Till nästa version lade jag till multipelt undo och redo samt möjligheten att se tabellen i större format. Nästa steg var att använda en databas för att lagra modellen, den hämtas in med hjälp av databasanrop när man startar programmet, men den bestod fortfarande av bara tre objekt. Nu lade jag även till möjligheten för användaren att se en SQL-fråga som ska hämta in de data som behövs för tabellen. I den senaste versionen använder jag en datamodell som har fem objekt och 49 variabler, jag har även lagt till möjligheter att spara i PC-AXIS format och att byta mellan att visa kod eller klartext för värdena.

Modellen grundar sig på några av de register som ingår i DAISY. Jag har använt fyra olika register och deras variabler och värdemängder. Ett av dessa register delades upp i två delar, eftersom det fanns en naturlig uppdelning i det, varje del blev ett objekt i modellen. Totalt gav detta fem objekt. Det fanns från början en förhoppning om att det skulle vara möjligt att plocka ut den önskade informationen ur den metadatabas som fanns i DAISY, detta visade sig tyvärr inte vara möjligt. Att detta inte gick berodde på att variablerna och deras värdemängder ligger lagrade i textform i databasen på samma sätt som de visades på skärmen, d.v.s. inklusive rubriker och kommentarer, och för att plocka ut data så skulle man vara tvungen att skriva ett program som tolkade varje rad och letade efter värden och koder. Detta skulle ta för lång tid, och inte vara värt mödan. Det hela slutade med att jag fick koda in det för hand, vilket i och för sig inte tog särskilt lång tid.

## 6.4. Kommentarer till prototypen

I det förslag som presenterades i [Malmborg 88] var det tänkt att användaren skulle markera sambanden mellan två objekt för att på så sätt indikera att de skulle kopplas ihop. I min prototyp så behöver användaren ej göra detta eftersom det ligger inbyggt i semantiken, programmet letar själv reda på vilka samband som finns mellan olika valda objekt. Detta leder till att det ej är möjligt att ha mer än en koppling mellan två register i modellen, för det skulle då inte vara möjligt för programmet att avgöra vilken koppling som menas i varje fall. Om man trots allt har två eller flera kopplingar mellan olika register så kan man separera dem så att de bildar flera fristående objekt i datamodellen.

Den största bristen i prototypen, enligt min åsikt, är att den saknar möjligheten att byta placering och ordning på värdemändger som redan finns i tabellen. Undo/redo-funktionerna ger bara möjlighet att ta bort och lägga tillbaka. För att få prototypen mer användbar skulle en funktion liknande den i Supercross kunna implementeras.

Jag har i min prototyp ej heller lagt in något verktyg för att göra grupperingar av värdena. Det menyval som finns i *värdemenyn* öppnar endast ett tomt gränssnitt, utan funktionalitet.

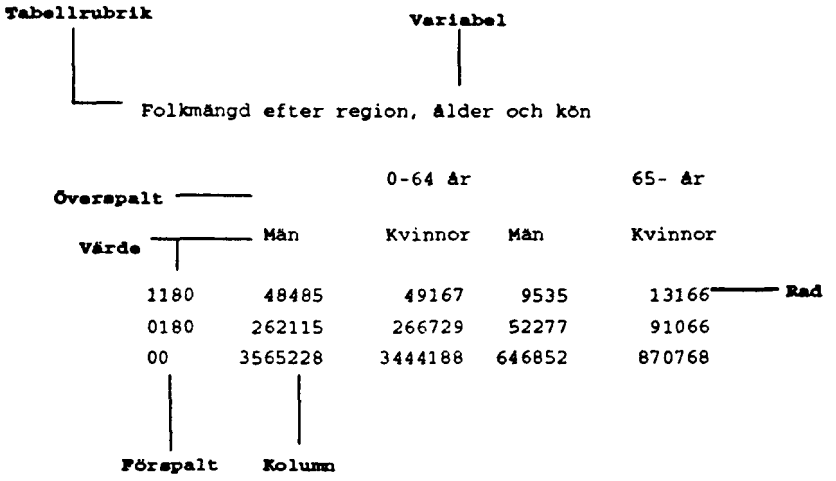
# 7. Slutsatser

Undersökning av programmen Supercross, GQL och PC-AXIS har lett mig att dra slutsatsen att Supercross är det alternativ som är mest lättarbetat. Slutsatsen grundar sig på att det i Supercross var enkelt och intuitivt att välja ut de variabler som skulle ingå i tabellen, samt att det även var lätt att omforma tabellens utseende i ett senare skede. PC-AXIS har valt lösningar som mycket liknar de som finns i Supercross, men de är inte riktigt lika lättanvända, detta kan bero på att Supercross använder sig av de grafiska möjligheter som finns i Windows, medan PC-AXIS bara använder sig av fönster som visas med hjälp av vanliga tecken.

Bland gränssnitten som återfanns i litteraturstudien anser jag att de som var baserade på Entity-Relationship diagram (ER-diagram) bäst presenterar den information som finns tillgänglig, samt att de även på ett naturligt sätt låter användaren välja vilka variabler och värden som ska ingå i tabellen. Metoder för att använda ER-diagram fanns beskrivet dels i [Wong 83], där ett system som kallas GUIDE presenteras, och dels i [Malmborg 88].

I den prototyp som jag konstruerade har jag använt mig av de ideer som finns i [Malmborg 88]. I prototypen kan man konstruera en tabell genom att successivt välja värden i olika variabler och placera dem i förspalt eller överspalt. Det finns även en databas där det finns värden lagrade för några av de register som finns i prototypen, detta gör att man även kan fylla tabellen med värden. För att hämta in data till tabellen så konstrueras en SQL-fråga som hämtar in den information som behövs från databasen.

# Ordlista och förklaringar




| | |
|---|---|
| förspalt | Rubriker till raderna. |
| korstabell | En tabell som har både förspalt och överspalt, t.ex. tabellen ovan. |
| makrodata | Bearbetade data, t.ex. grupperade efter ålder. |
| metadata | Data om data, används för att beskriva hur en t.ex. en databas är uppbyggd. |
| mikrodata | Den enklaste typen av information, de är ej grupperade på något vis. |
| variabel | En egenskap hos ett objekt eller ett fält i en tabell. |
| värdemängd | De värden som en variabel kan anta. |
| överspalt | Rubriker till kolumnerna. |

| | |
|---|---|
| AXIS | Auxiliary System for Interactive Statistics. Används på SCBs stordatorer för att lagra och presentera data. |
| OPREM | SCBs utvecklingsmetod för databasapplikationer. Förkortning för Object-Property-Relationship-Event-Message. |
| GQL | Graphical Query Language |
| WYSIWYG | What-You-See-Is-What-You-Get |

## Litteraturförteckning


[Chan 81] P. Chan och A. Shoshani, SUBJECT: A Directory Driven System For Organizing And Accessing Large Statistical Databases, *Proc. of the Sixth International Conference of Very Large Data Base*, sid. 553-563, 1980

[Elmasri 89] R. Elmasri och S. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 1989

[Foley 90] J.D. Foley et al, *Computer Graphics: Principles and Practice*, Addison-Wesley, 1990

[Langefors 75] B. Langefors och B. Sundgren, *Information Systems Architecture*, Mason/Charter, New York, sid 237-250, 1975

[Malmborg 88] E. Malmborg, Design of the User Interface for an Object-oriented Statistical Database, *Proc. of 4th International Conference on Statistical and Scientific Database Management*, 1988

[Rafanelli 90] M. Rafanelli och F.L. Ricci, A Visual Interface for Browsing and Manipulating Statistical Entities, *Proc. of 5th International Conference on Statistical and Scientific Database Management*, sid. 163-182, 1990

[Sato 86] H. Sato et al, Conceptual Scheme for a Wide-Scope Statistical Database and Its Applications, *Proc. of 3rd International Workshop on Statistical and Scientific Data Base Management*, sid. 165-172, 1986

[Sato 88] H. Sato, A Data Model, Knowledge Base, and Natural Language Processing for Sharing a Large Statistical Database, *Proc. of 4th International Workshop on SSDBM, sid. 207-226, 1988*

[Schneiderman 87] B. Schneiderman, *Designing the User Interface*, Addison-Wesley, 1987

[Shoshani 80] A. Shoshani, Statistical Data Bases: Characteristics, Problems, and Some Solutions, *Proc. of the Eighth International Conference On Very Large Data Bases*, sid. 208-222, 1980

[Sundgren 84] B. Sundgren, *Conceptual Design of Data Bases and Information Systems*, University of Linköping and Statistics Sweden, 1984

[Sundgren 85] B. Sundgern, *Outline of an Algebra of Base Operators for Production of Statistics*, University of Lindköping and Statistics Sweden, 1985

[TAB68] *TAB68 a Programming Language for Table Production*, Statistics Sweden, 1985

[Wong 83] H.K.T. Wong och W-L. Yeh, A Graphical Query System for Complex Statistical Databases, *COMPUTER SCIENCE AND STATISTICS: The Interface*, sid. 35-49, 1983

**R & D Reports** är en för U/ADB och U/STM gemensam publikationsserie, som fr o m 1988-01-01 ersätter de tidigare "gula" och "gröna" serierna.

**R & D Reports Statistics Sweden** are published by the Department of Research & Development within Statistics Sweden. Reports dealing with statistical methods have green (grön) covers. Reports dealing with EDP methods have yellow (gul) covers.

Reports published 1993:

| | |
|---|---|
| 1993:1 (grön) | Kvalitetsfonden 1991/92 - Projekt, handläggning och utvärdering (Roland Friberg) |
| 1993:2 (grön) | Översyn av Undersökningen av lastbilstransporter i Sverige (UVAV) (Bengt Rosén, Mahnaz Zamani) |
| 1993:3 (grön) | Bortfallsbarometern nr 8 (Mats Bergdahl, Pär Brundell, Jan Hörngren, Håkan Lindén, Peter Lundquist, Monica Rennermalm) |
| 1993:4 (gul) | Guidelines on the Design and Implementation of Statistical Meta-information Systems (Bo Sundgren) |
| 1993:5 (grön) | Kvalitetsrapporten 1995 - Utveckling av kvaliteten för SCBs statistikproduktion (Jan Eklöf, Per Nilsson) |
| 1993:6 (grön) | Metoder att få konsistens mellan olika skattningar och samtidigt öka precisionen (Sixten Lundström) |
| 1933:7 (grön) | Variance Estimation in the Swedish Consumer Price Index (Jörgen Dalén, Esbjörn Ohlsson) |
| 1933:8 (grön) | Quantifying Errors in the Swedish Consumer Price Index (Jörgen Dalén) |

Tidigare utgivna **R&D Reports** kan beställas genom Ingvar Andersson, SCB, U/LEDN, 115 81 STOCKHOLM (telefon 08-783 41 47, telefax 08-783 45 99).

**R&D Reports** listed above as well as issues from 1988-92 can - in case they are still in stock - be ordered from Statistics Sweden, att. Ingvar Andersson U/LEDN, S-115 81 STOCKHOLM, SWEDEN (telephone nr 46 08 783 41 47, telefax nr +46 08 783 45 99).