LUXOR
DATORER

& och kommandospr

IMER/Q

Version 3.3

BRUKSANVISNING

Handbok i fråge- och kommandospråket

MIMER/QL

Version 3.3

MIMER

MIMER Information Systems AB
Uppsala

Utgivare: MIMER Information Systems AB, Box 1713,
751 47 UPPSALA, tel 018-15 64 50

Titel: Handbok i fråge- och kommandospråket MIMER/QL

Sättning: Delta Reprosätteri, Stockholm och
Merkantil-Tryckeriet, Uppsala
Tryckning: Merkantil-Tryckeriet i Uppsala AB, 1984

ISBN 91-7878-0012

Innehåll

1 MIMER/QL och MIMER-systemet – läs detta först

Den här handboken beskriver hur man använder MIMER/QL (version 3.3). MIMER/QL är ett interaktivt fråge- och kommandospråk. Den som använder det behöver inte ha några större datakunskaper.

MIMER/QL används i kombination med databashanteraren MIMER/DB för att definiera och bygga upp databaser samt för att söka och manipulera data. "QL" är en förkortning av "query language", som betyder frågespråk. Begreppet frågespråk syftar på möjligheten att med språkets hjälp ställa spontana, oplanerade frågor och få fram svar om innehållet i en databas. Men MIMER/QL kan användas till mer avancerade uppgifter än så, t ex att skapa system helt byggda på s k *menyer* eller att göra beräkningar.

MIMER/QL version 3.3 är en försvenskad version, vilket innebär att kommandoord, parametrar, felmeddelanden, hjälptexter osv är på svenska.

Om handboken

Handboken vänder sig till alla kategorier användare. För den som har ringa ADB-kunskaper finns en litteraturförteckning (bilaga G) med lämpliga introduktionsböcker att läsas vid behov och vidare en kommenterad förteckning över viktiga ord och begrepp som används i handboken (bilaga H) – om läsaren stöter på något okänt och oförklarat databegrepp i texten finns det förhoppningsvis i denna lista.

Handboken är tänkt både som introducerande lärobok i MIMER/QL och som uppslagsbok. Särskilt i den senare funktionen bör det alfabetiska slagordsindex som finns sist i handboken vara till nytta.

I kapitel 2 förklaras översiktligt och lättillgängligt hur databashanteraren i MIMER/DB fungerar. Den som inte redan känner till detta rekommenderas att läsa kapitlet, eftersom det bidrar till förståelsen av MIMER/QL. Den som vill skaffa sig detaljerade kunskaper om MIMER/DB hänvisas till MIMER/DB Reference Manual (på engelska).

Ett genomgående exempel används för att belysa olika kommandon och deras funktioner. Exemplet är baserat på den tänkta verksamheten i ett påhittat företag, MIMER HOTELL AB. Förutsättningarna beskrivs i kapitel 3. I bilaga F återges hela körningen av exemplet. Exemplet följer också med i form av en delvis upplagd databank när MIMER/QL levereras. Tala med din MIMER-ansvariga om hur du skall få tillgång till denna databank för egna övningar.

Att lära sig MIMER/QL

Även användare med inga eller obetydliga ADB-kunskaper bör utan större svårigheter kunna lära sig att praktiskt använda det mesta i MIMER/QL enbart med handbokens hjälp. För den som snabbt vill nå god färdighet rekommenderas de koncentrerade kurser om MIMER/QL som hålls av MIMER Dataskola i Uppsala.

Om du inte redan lärt dig grunddragen i MIMER/QL på annat sätt, rekommenderar vi dig att starta med att bläddra och ögna igenom handboken så att du får en översiktlig uppfattning om dess innehåll och om MIMER/QL. Titta också på innehållsförteckningen.

Därefter kan du läsa boken avsnitt för avsnitt. När du börjar med ett nytt avsnitt i boken bör du först ögna igenom det och sedan läsa det mer grundligt. I anslutning till läsningen bör du vid terminalen pröva de exempel som förekommer. Träning är mycket viktigt om du vill lära dig använda MIMER/QL som det smidiga och effektiva verktyg det är. Konstruera gärna egna exempel efter de mönster som ges i handboken.

När du läser handboken kommer du att finna att viss information upprepas. Orsaken till det är att boken också skall kunna fylla sin funktion som uppslagsbok/handbok utan alltför mycket bläddrande.

MIMER-systemets olika delar

MIMER/QL ingår i databassystemet MIMER, som utvecklas och marknadsförs av MIMER Information Systems AB i Uppsala. Företaget ägs av Uppsala universitet, AB Programator och anställda.

MIMER-systemets olika delar är *integrerade*, vilket bl a innebär att det finns enkla övergångar mellan dem och att data från ett delsystem kan utnyttjas i ett annat. MIMER-systemet är *portabelt*, dvs det kan köras på olika datormärken och under olika operativsystem (f n ett 15tal datorsystem). Systemets version 3 har utöver MIMER/QL följande delsystem:

- **Databashanteraren MIMER/DB**
  MIMER/DB som bygger på den s k *relationsmodellen* är grunden för övriga delsystem. MIMER/DB används bl a för att skapa och underhålla databaser samt för att upprätthålla databassystemets s k *datakatalog* (data dictionary). MIMER/DB är dessutom gränsytan mot andra system. Vad som menas med relationsmodell och datakatalog förklaras i kapitel 2.

- **Applikationsgeneratorn MIMER/PG**
  Med MIMER/PG kan applikationsprogram för valfria ändamål skapas utan programmering i något konventionellt programspråk som exempelvis COBOL. Det går också att kombinera konventionellt skrivna program eller delrutiner med rutiner skapade i MIMER/PG. I MIMER/PG ingår även en rapportgenerator för effektiv utformning av informativa rapporter. MIMER/PG är i första hand avsett som ett arbetsbesparande verktyg för programmerare.

- **Skärm- eller formulärhanteraren MIMER/SH**
  Med MIMER/SH kan man bygga upp bildskärmslayouter, t ex inmatningsformulär och menyer, som sedan kan användas i olika tillämpningar.

- **Informationssökningssystemet MIMER/IR**
  MIMER/IR används för textlagring och sökning i stora databaser, tex bibliotekssystem. Sökning kan ske både i strukturerad och ostrukturerad text. I MIMER/IR finns även en sk *thesaurusfunktion* som medger sökning på begrepp närbesläktade med sökbegreppet (exempel: sökning på "bil" kan ordnas så att även "personvagn", "traktor" etc ger sökträff). MIMER/IR kan efter kort träning användas även av den som inte har ADB-kunskaper.

- **Statistiksystemet MIMER/ST**
  Med MIMER/ST kan en slutanvändare med enkla kommandon bearbeta data statistiskt. Data kan hämtas både från MIMER-tabeller och sekvensiella filer eller matas in direkt på terminal. Användaren kan välja mellan ett 40-tal olika statistiska analyser. MIMER/ST kan efter kort träning användas även av den som inte har ADB-kunskaper.

- **Grafiksystemet MIMER/GR**
  MIMER/GR gör det möjligt att med enkla kommandon omforma data till kurvor, sektordiagram och stapeldiagram, om så önskas i färg. MIMER/GR kan utnyttjas för grafisk presentation av statistiska analyser gjorda med MIMER/ST. MIMER/GR kan efter kort träning användas även av den som inte har ADB-kunskaper.

Utöver de beskrivna delsystemen finns hjälpprogram för vanligt förekommande operationer i databasen.

Figur 1:1 visar sambandet mellan MIMER/QL och MIMER/DB. Som framgår av figuren är MIMER/QL i själva verket ett applikationsprogram, som utnyttjar MIMER/DB som databashanterare. Databasen nås via ett gränssnitt, som består av rutiner för överföringen av data mellan databasen och MIMER/QL.

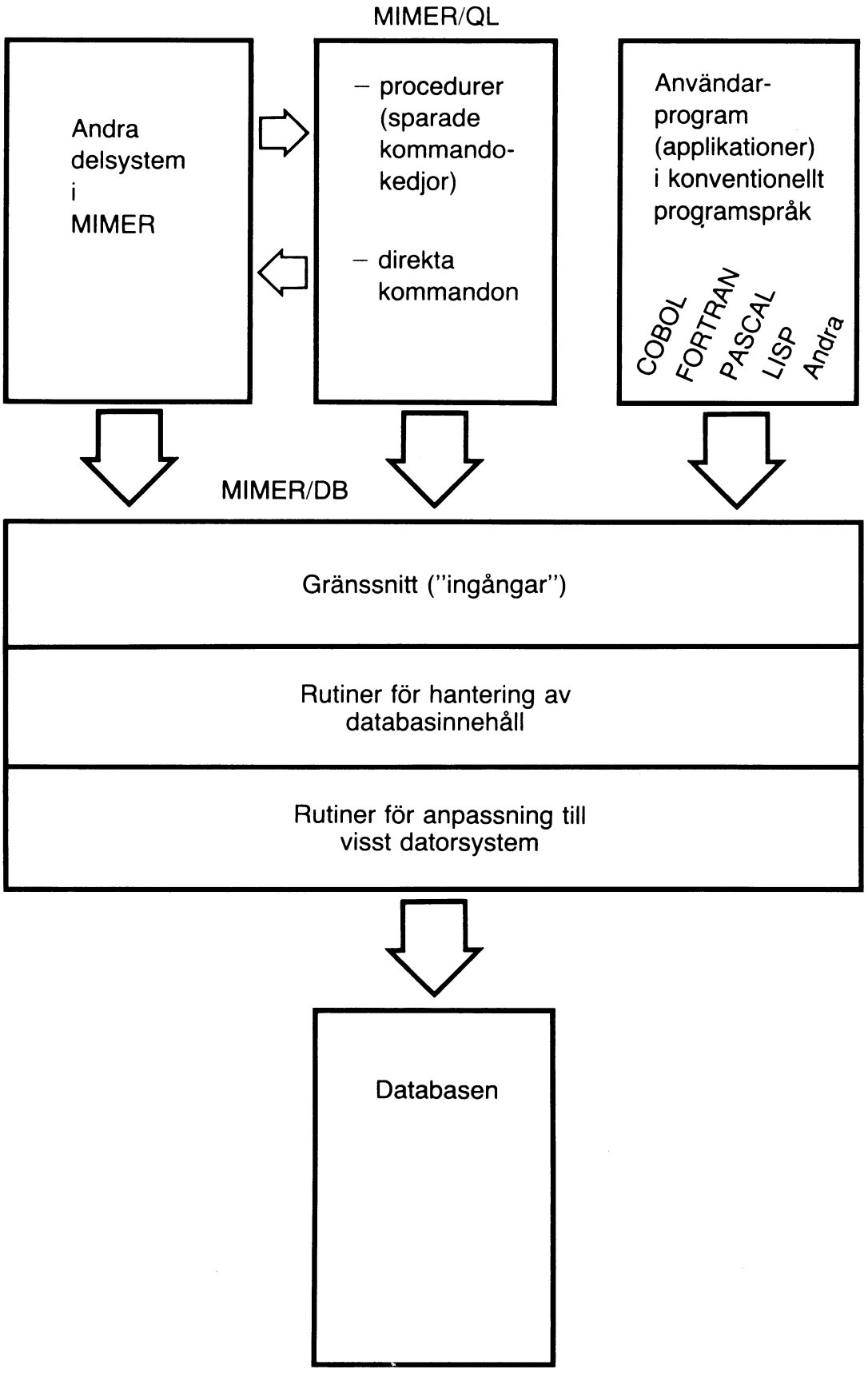


Fig 1:1. Sambandet mellan MIMER/QL och MIMER/DB.

Andra dokument

De övriga delsystemen beskrivs i handböcker som kan rekvireras från MIMER Information Systems AB. Till version 3 finns endast handböckerna till MIMER/QL och MIMER/IR på svenska, övriga handböcker är på engelska. Handböcker på svenska till andra delsystem planeras.

Till flertalet delsystem, däribland MIMER/QL, finns kortfattade lathundar som beskriver kommandonas formella uppbyggnad. De kan användas som minnesstöd vid terminalen för den som en gång lärt sig grunderna i ett delsystem.

Skriften "MIMER concepts and facilities" ger en mer ingående översikt över hela MIMER-systemet och lämpar sig bra som introduktion för den som kommer att använda flera delsystem i MIMER.

Hjälpinformation på skärmen

Användaren av MIMER/QL har direkt på bildskärmen tillgång till beskrivningar av kommandonas form och funktion samt olika parametrar och variabler i kommandona. Läs vidare om detta i anslutning till kommandot HJÄLP i bilaga C.

Använda skrivsätt och beteckningar

Kommandoord återges i löpande text med stora bokstäver, tex ÖPPNA. I kommandoxempel återges kommandoorden med små bokstäver och i fet stil, tex

öppna HOTELLDB(ANSTÄLLD);

I syntaxbeskrivningar, dvs de generella mönster efter vilka kommandon i MIMER/QL skall skrivas, markeras den förkortade versionen av ett kommandoord med stora bokstäver, tex

ÖPPna databanksnamn(tabell1,tabell2);

Observera att det inte spelar någon roll om ett kommando skrivs in med stora eller små bokstäver.

Det som står mellan piltecken i en syntaxbeskrivning, tex <databanksnamn> kan utelämnas. Ett lodrätt streck, |, i en syntaxbeskrivning avgränsar olika alternativ av vilka ett kan väljas.

Det användaren skriver (kommandon, inmatade värden o dyl) anges med rak stil, medan det datorn skriver ut på skärmen anges med kursiv stil (undantag är markörer, tex QL>, som anges med rak stil).

Vagnretur vid inmatning

Glöm inte att all inmatning — av kommandon, data m m — måste åtföljas av tryckning på tangenten för vagnretur (vanligen betecknad RETURN, CARRIAGE RETURN eller CR).

2 MIMER/DB – översikt och grundbegrepp

MIMER/QL utnyttjar databashanteraren MIMER/DB för att läsa och skriva i databasen. I detta kapitel ges därför en översiktlig beskrivning av MIMER/DB. För hela MIMER-systemet viktiga grundbegrepp behandlas också.

2.1 Om datalagring

En traditionell metod att lagra data är att lägga *poster* i en *sekvensiell fil*, tex en fil med personuppgifter. En post är en uppsättning data indelad i olika *fält*. Olika fält i en post kan tex vara en persons förnamn, efternamn, adress och telefonnummer.

I en sekvensiell fil läggs poster efter varandra, i sekvens, så som figur 2:1 visar. Slutet på varje post markeras med ett speciellt tecken. För att leta reda på en viss post är det nödvändigt att läsa filen ända från början tills posten hittats. Varje gång en post eftersöks måste samma läsprocedur upprepas.

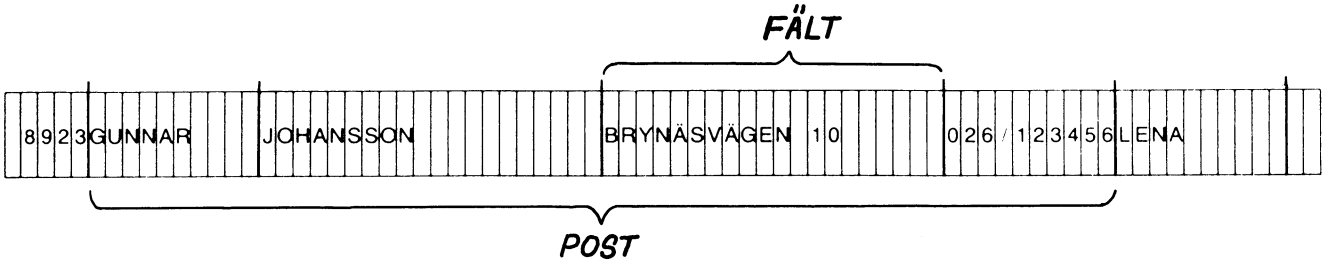


Fig 2:1. I en sekvensiell fil lagras posterna efter varandra.

Att använda *databas* är för många sammanhang en smidigare metod att lagra data. Metoden karakteriseras bl a av följande:

- data upprepas inte onödigtvis på flera ställen
- mängden data kan ökas obegränsat inom ramen för den fysiska storleken på yttre minnen
- nya slag av data kan läggas till utan ändringar av databasen i övrigt
- vilka informationsuttag (kombinationer och bearbetningar av data) som kan göras behöver inte bestämmas i detalj på förhand

En databas manipuleras med en programvara som kallas *databashanterare*. Databaser och databashanterare kan fungera enligt olika principer. MIMER/DB arbetar efter den s k *relationsmodellen*. En viktig del av ett relationsdatabassystem är *datakatalogen* (data dictionary). Den finns dels i en särskild tabell i varje databank (där innehållet i databanken beskrivs, avsnitt 2.3) och dels i *systemdatabanken*, SYSDB (där bl a användare och användarbehörigheter noteras, avsnitt 2.4).

2.2 Tabeller

I relationsdatabaser är data organiserade i tabeller, så som figur 2:2 visar. En *tabell* är i princip lika med en fil. Tabellen kan t ex innehålla personuppgifter.

Tabellen består av *rader* och *kolumner*. Raderna motsvarar poster och kolumnerna fält i ett sekvensiellt filsystem. I en tabell kan t ex finnas kolumner för personers förnamn, efternamn, adress och telefonnummer. En rad i tabellen innehåller då dessa uppgifter för en viss person.

Om man skall kunna hitta en viss rad i en tabell måste det gå att skilja varje rad i tabellen från varje annan rad genom en unik identifikation. Identifikationen utgörs av innehållet i en eller flera kolumner och kallas *primärnyckel*. I vår tabell med personuppgifter måste primärnyckelkolumnerna vara förnamn, efternamn och telefonnummer för att säkra en, praktiskt sett, unik identifikation. Flera personer kan ju ha samma förnamn, efternamn, adress eller telefonnummer, samma kombination av förnamn och efternamn kan förekomma hos flera personer och flera personer med samma för- och efternamn kan bo på samma adress. Däremot är det ytterst ovanligt att det finns två eller flera personer med både samma telefonnummer och samma för/efternamn.

En kolumn är den minsta enheten i MIMER/DB. Varje kolumn tilldelas ett *kolumnnamn*, t ex "förnamn". Kolumnernas inbördes ordning i tabellen har ingen betydelse, med ett viktigt undantag: Den eller de kolumner som utgör primärnyckel måste finnas längst "till vänster" i tabellen. I vår tabell med personuppgifter måste således de tre kolumnerna förnamn, namn och telefonnummer (i inbördes godtycklig ordning) komma först.

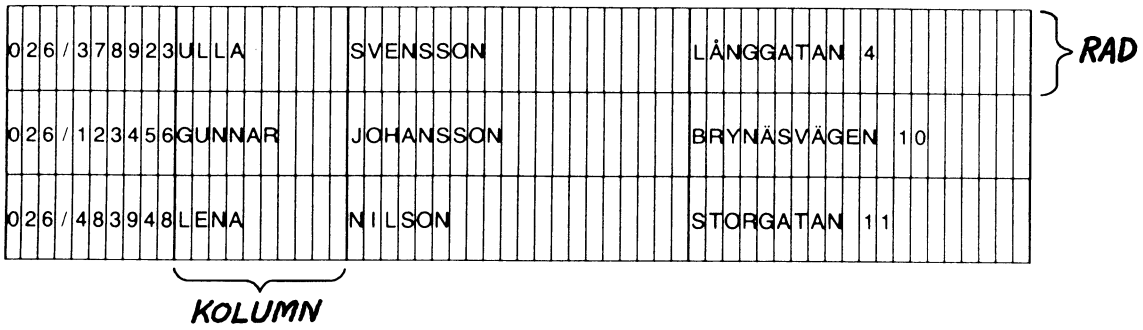


Fig 2:2. MIMER-tabellernas uppbyggnad.

Värdena i en viss kolumn har alltid samma format. Med *värde* menas inte bara ett talvärde utan varje tänkbar kombination av tecken. Med *format* menas värdets typ och storlek (dvs antalet teckenpositioner i kolumnen). Typ betecknas i MIMER/DB med en bokstav, enligt följande:

C för alfanumeriska data, dvs data där både siffror, bokstäver och andra tecken kan ingå

I för heltal

F för flyttal (decimaltal)

En kolumn med formatbeteckningen C10 kan således lagra 10 tecken, såväl siffror som bokstäver eller specialtecken, på varje rad. Man talar om olika *kolumntyper* alltefter vilken typ av data kolumnerna innehåller.

Som framgått kan man för numeriska data således välja vilken som helst av de tre typerna C,I och F. När bör man då använda den ena eller den andra?

Det finns två huvudskäl till att använda typ I eller F vid lagring av numeriska data:

- Man sparar minnesutrymme, dock på bekostnad av extra CPU-tid, vilket medför att körningen går något långsammare.
- Man är säker på att värden sorteras i rätt ordning, eftersom alla värden har en unik representation. Det är inte fallet vid typ C. Den unika representationen är viktig om värdet utgör primärnyckel.

Ett exempel belyser problemet med sortering:

5 och +5, som ju är samma sak, betraktas som två olika värden vid typ C. Om de hanteras som typ C ordnas de vid en sortering enligt sina ASCII- eller EBCDIC-värden (varje tecken representeras av ett unikt talvärde). Låt oss anta att talen 15, -5, 5 och +5 skall sorteras i stigande ordning. Resultaten vid olika kolumntyper blir då följande:

Kolumntyp C	Kolumntyp I
5	–5
+5	5
–5	
15	15

Lägg märke till att värdet 5 och +5 bara har en representation vid typ I.

När man använder kolumntyp F, måste man vara medveten om att värdena har en viss precision och att man kan råka ut för avrundningsfel. Speciellt gäller detta om man använder olika kolumnstorlek, t ex F4 och F3, för samma data. Ett visst tal, som lagras i två kolumner av olika längd, kan alltså komma att betraktas som två olika tal.

Om ett tals absolutvärde, dvs det rena värdet utan tecken, är för stort för kolumnen kan vid typ F eller I s k *spill* eller *overflow* inträffa.

Slutsatsen blir alltså att om man bara skall *flytta* numeriska data till och från databasen, kan man mycket väl lagra dessa data i kolumner av typ C. Om man också har för avsikt att *söka* på numeriska data bör man i stället använda kolumntyp F eller I.

I andra typer av databaser kan ett fält representera flera värden, vilka kan tänkas ligga längre ned i en hierarkisk trädstruktur, som förgrenar sig nedåt nivå efter nivå. Grupper av värden på lägre nivåer ingår i grupper på högre nivåer. Man brukar tala om *repeterande grupper*. I en tabell i en relationsdatabas råder inga sådana underförstådda hierarkiska samband. Ett värde i en kolumn kan inte representera en grupp av värden.

2.3 Databanker

En *databank* består av en eller flera tabeller för lagring av data. Den innehåller dessutom alltid en tabell som inte används för lagring av vanliga data utan för data som beskriver innehållet i databanken (tabellnamn, kolumnstorlekar, kolumntyper etc). Denna tabell är en del av den tidigare nämnda *datakatalogen* (data dictionary).

En databank kan innehålla ett obegränsat antal tabeller. Om databanken innehåller flera tabeller, måste varje tabell ha ett inom databanken unikt namn. Även själva databanken måste ha ett unikt namn.

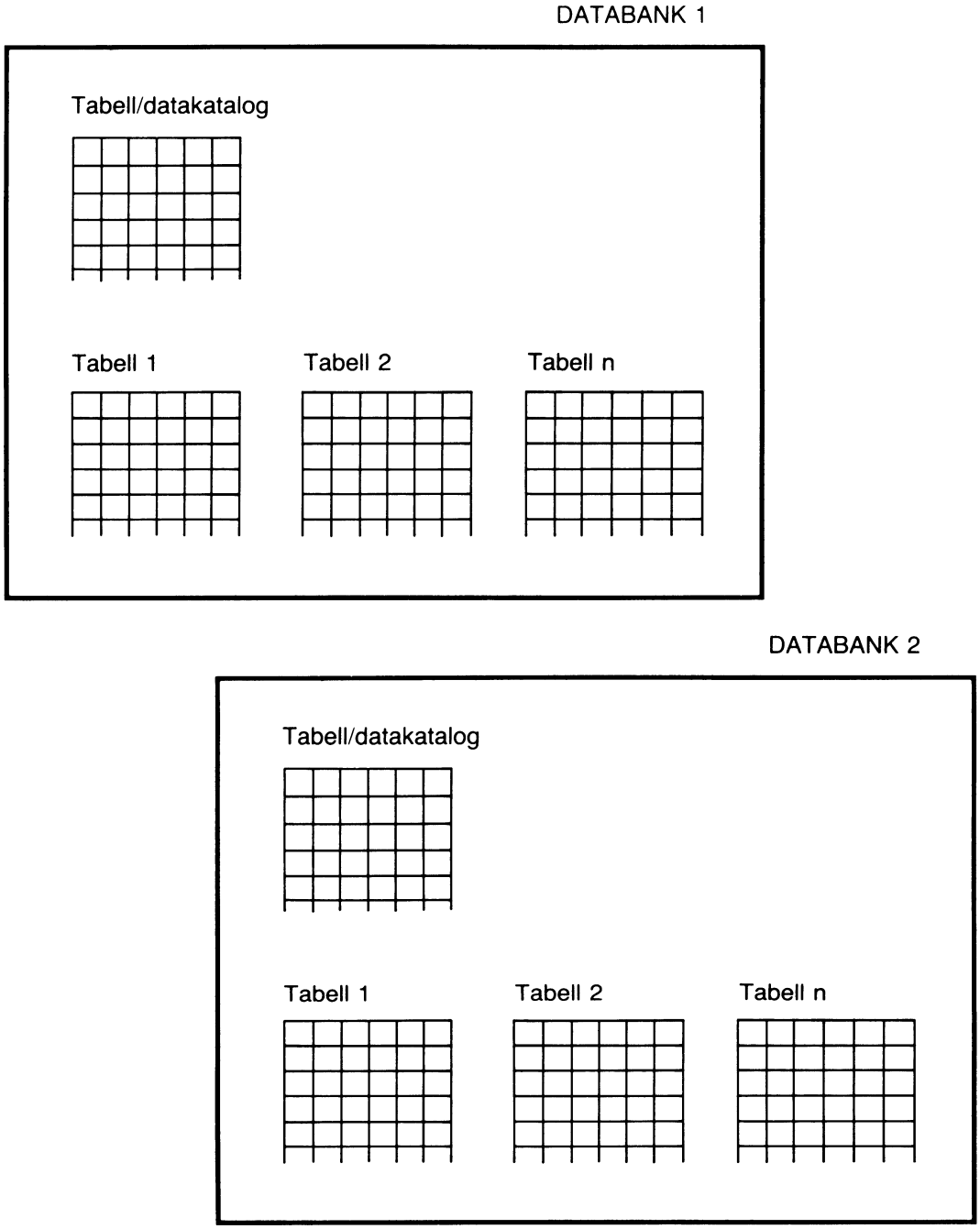


Fig 2:3. Databankernas uppbyggnad.

Datorns operativsystem behandlar databanken som en fil på ett yttre minne, tex ett skivminne.

När man definierar en databank i MIMER/DB, måste man specificera databankens storlek (antal sidor). Sidans storlek är beroende av vilket datorsystem MIMER används på. Systemberoende information kommer upp på skärmen med hjälp av ett speciellt kommando, se bilaga A.

2.4 SYSDB – systemets databank

För styrning av hela databasen finns det en övergripande databank, som kallas SYSDB, *systemdatabanken*. SYSDB är den andra delen av *datakatalogen* (data dictionary) och påminner till sin funktion om databanksnivåns särskilda tabeller för databeskrivning. Den innehåller information om databankerna, om systemets användare och om den koppling som gäller mellan användare och databanker, dvs användarnas behörighet att hantera olika databanker. I SYSDB knyts också varje databanks namn ihop med motsvarande fysiska fil på det yttre minnet.

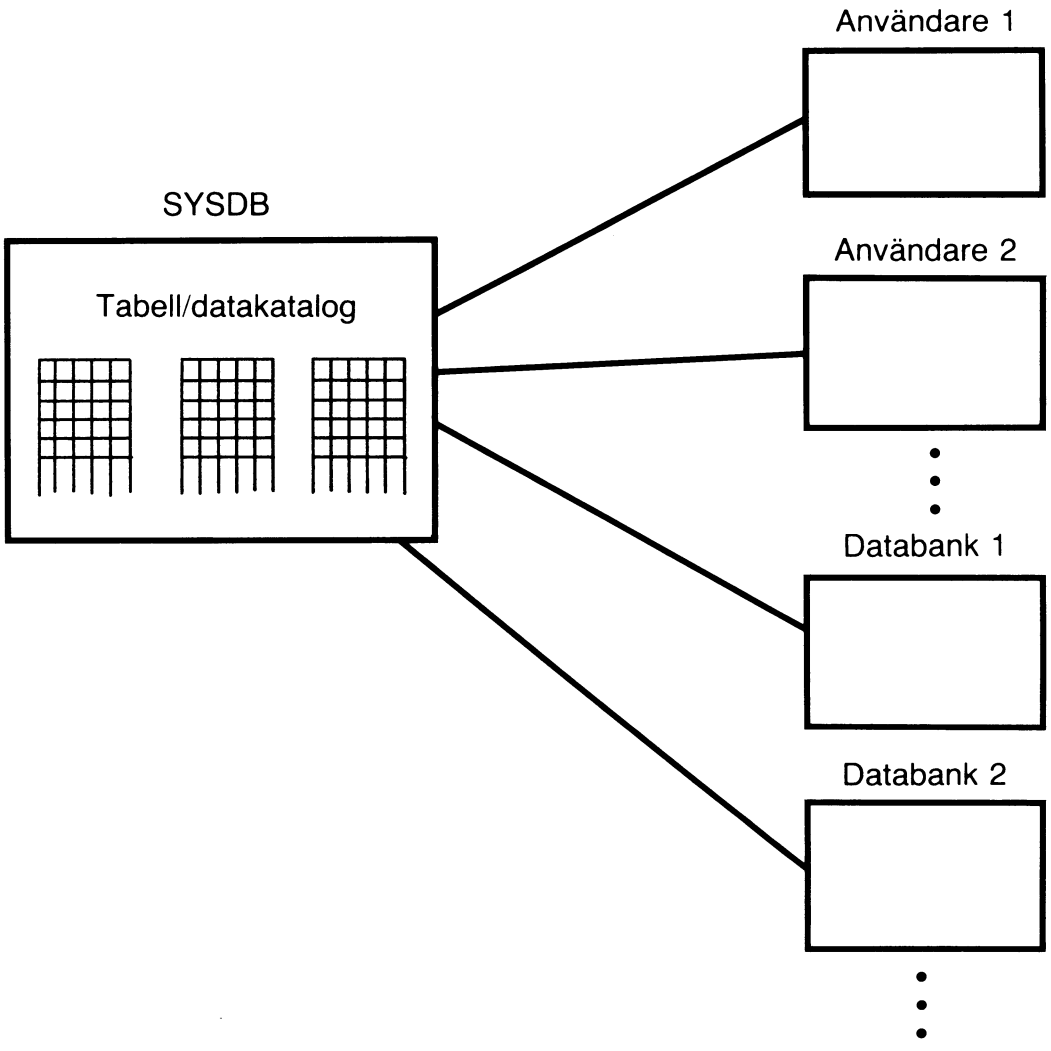


Fig 2:4. Förhållandet mellan SYSDB, användare och databanker.

2.4.1 Uppgifter om databankerna

I SYSDB finns följande uppgifter om varje databank:

- Databankens namn
- Generell användarbehörighet
- Filnamn

Databankens namn får bestå av högst åtta alfanumeriska tecken. Med hjälp av namnet skall man kunna identifiera databanken, vilket innebär att varje databank måste ha ett unikt namn inom den aktuella databasen.

För varje databank anges en *generell användarbehörighet.* Den gäller för alla användare, som inte tilldelats någon personlig behörighet (avsnitt 2.4.2). Den generella behörigheten anger om en användare utan personlig behörighet får göra något med databanken och i så fall vad. Följande generella behörigheter finns (inom parentes anges den engelska term som respektive behörighetsbeteckning kommer av):

B Ingen generell behörighet, databanken innehåller säkerhetskopior av data (back-up).

P Ingen generell behörighet, databanken disponeras endast av en viss användare (private).

R Användarna kan bara *läsa* tabellerna i denna databank (read-only).

S Användarna kan *läsa* och *uppdatera* (dvs ändra, lägga till och ta bort rader) i tabeller (shared).

X Användarna kan utföra *alla förekommande operationer* i denna databank – förutom att läsa och skriva i tabellerna, kan de ändra tabelldefinitioner, vilket innefattar att lägga till nya tabeller och ta bort gamla (exclusive).

2.4.2 Uppgifter om användarna

Endast de användare, som definieras i SYSDB, har tillgång till de databanker, som definieras i SYSDB.

I SYSDB finns följande uppgifter om varje användare:

- Namn
- Ställning
- Lösenord

Användarens *namn* får bestå av högst åtta alfanumeriska tecken.

Användarens *ställning* kan vara endera av två möjliga:

S Vanlig användare (standardanvändare)

X Databasadministratör, förkortat DBA (användare med exklusiva rättigheter)

Skillnaderna mellan en vanlig användare och en databasadministratör förklaras i avsnitt 2.6. Databasadministratörens uppgifter och befogenheter tas upp på flera håll i handboken och sammanfattas översiktligt i kapitel 17.

Lösenordet kan bestå av upp till åtta tecken. Det lagras i chiffrerad form.

2.4.3 Personlig användarbehörighet

Varje användare kan dessutom tilldelas en *personlig behörighet* till en databank. Om personlig behörighet definierats för en användare är det den som gäller, inte den för databanken angivna generella behörigheten (se avsnitt 2.4.2).

En användare kan tilldelas någon av följande personliga behörigheter till en databank (beteckningens motsvarande engelska term inom parentes):

P Användaren har ingen behörighet till databanken (private).

R Användaren kan bara *läsa* tabellerna i databanken (read-only).

S Användaren kan kan både *läsa* och *uppdatera* (ändra och ta bort rader) i tabellerna i databanken (shared).

X Användaren kan utföra alla förekommande operationer i databanken – förutom att läsa, ändra och ta bort rader, kan användaren ändra tabellers utseende samt tilldela personlig behörighet till andra användare (exclusive).

Observera att det finns en väsentlig skillnad mellan personlig användarbehörighet **X** och generell användarbehörighet **X**. I förstnämnda fall kan användaren tilldela andra användare behörighet till databanken, något som inte är möjligt i sistnämnda.

2.5 Transaktionshantering

Med *databasoperationer* menas läsning, tillägg, ändring och radering av tabellrader i databasen. När flera användare från sina terminaler samtidigt vill utföra sådana operationer i en gemensam databas skulle konflikter kunna uppstå.

För att sådana olägenheter skall undvikas finns tillgång till s k *transaktionshantering* i MIMER/DB. Med detta avses att en viss användares databasoperationer samlas eller "buntas ihop" till enheter, s k *transaktioner*. Transaktionshanteringen fungerar enligt principen "allt eller intet": en transaktion är odelbar – antingen utförs hela transaktionen eller också sker ingenting alls i databasen. Det är en garanti för att databasen och dess innehåll förblir under kontroll.

Om transaktionshantering skall träda i kraft, måste man definiera en databank, som heter TRANSDB. När det har gjorts, betraktas varje kommando som uppdaterar en rad i tabellen som en transaktion.

Observera dock att följande operationer inte utförs radvis i en tabell och därför inte heller ingår i transaktioner:

- Definition och borttagning av tabeller
- Operationer som initieras med kommandona KOPIERA/LADDA eller RADERA utan DÄR-villkor (innebörden i detta framgår i avsnitt 9.2, 10.3)
- Ändringar i SYSDB

De nu nämnda operationerna kan dock bara utföras av dem som har ställning **X** som användare av databanken, dvs en databasadministratör (se avsnitt 2.4.2, 2.6 och kapitel 17).

2.5.1 Loggning av transaktioner

Alla transaktioner kan om man så önskar *loggas*, dvs registreras, i en speciell *loggfil*. Om ett maskinvarufel inträffar kan loggfilen användas tillsammans med en kopia av databanken för att återskapa databankens innehåll. Med hjälp av loggfilen kan man också se hur databankens innehåll har förändrats under en viss tid.

Om transaktionsloggning skall användas, måste man definiera en databank som heter LOGDB.

Som framgått av inledningen till detta avsnitt finns det vissa operationer, som inte kan ingå i transaktioner. Dessa operationer loggas således inte. När sådana operationer utförts, bör man därför ta en ny kopia av databanken. Detta arbete utförs av databasadministratören, vilken f ö som nämnts är den enda som har rätt att utföra ifrågavarande operationer.

2.6 Databasadministratören, DBA

Databasadministratören (DBA) har som framgått användarställning **X**. DBA har till skillnad från användare med ställning **S** befogenhet att ändra i SYSDB, dvs ändra uppgifter om databanker och användare. Det finns ett speciellt kommando – DBA – för sådana operationer.

Kommandot DBA kan emellertid också nyttjas av en vanlig användare med ställning **S** men då enbart för att kontrollera vad som finns i SYSDB,för att ändra det egna lösenordet samt för att ändra SYSDB-information, som rör databanker för vilka vederbörande användare har personlig behörighet **X**.

DBA är normalt den som ansvarar för säkerhetskopiering av databankerna.

Det kan finnas flera databasadministratörer, dvs flera användare med befogenhet **X**. Dessa är då gemensamt ansvariga för underhållet av SYSDB.

2.7 Enanvändar- och fleranvändarsystem

När man utvecklar en ny databas, bör man arbeta med den som ett *enanvändarsystem*, vilket innebär att databasen inte delas med någon annan användare. Då störs inte befintliga applikationer av att den nya databasen testas. Dessutom kan den som utvecklar det nya systemet tilldelas ställning **X** och därmed utföra alla nödvändiga ändringar i SYSDB under testfasen. Från terminalen märker användaren ingen större skillnad mellan att arbeta i enanvändar- och fleranvändarsystem.

Databasadministratörens uppgifter är desamma i enanvändarsystem som i fleranvändarsystem. Det är möjligt att låta andra användare, som också arbetar i enanvändarsystem, använda databasen för läsning. SYSDB måste då *öppnas* för enbart läsning vilket sker automatiskt när en användare med ställning **S** kör QL i enanvändarsystem.

Observera att när en databank har öppnats för användning i fleranvändarsystem, är den inte längre tillgänglig för användare, som arbetar i enanvändarsystem, och omvänt.

Kapitel 16 och Bilaga A innehåller ytterligare beskrivningar av enanvändar- och fleranvändarsystem.

3 Ett databasexempel

I handboken används ett genomgående exempel för att beskriva användningen av MIMER/QL. Det bygger på ett tänkt företag, MIMER HOTELL AB, som äger och driver en hotellkedja i mellansverige.

För att du ska kunna använda exempel och övningar utan omfattande förberedelser har en del grunddata om den fingerade verksamheten i MIMER HOTELL AB lagts in i företagets databas redan från början. Det finns t ex två databanker, HOTELLDB och ANSTDB, definierade. Ett antal tabeller är också upplagda i dessa båda databanker. En del rader är inlagda i tabellerna. Utgångsläget i form av ett fullständigt återgivande av tabellinnehållet framgår av bilaga F. Där finns också hela exemplet återgivet som ett sammanhängande arbetspass i MIMER/QL.

HOTELLDB innehåller uppgifter om de hotell som ingår i hotellkedjan: vilka rum som finns, vad de kostar, hur de är utrustade mm. ANSTDB innehåller uppgifter om hotellkedjans anställda.

Vi kan i nedanstående översikter se vilka tabeller som ingår i de båda databankerna. Namnen på olika kolumner samt kolumntyp/format framgår. En * vid typbeteckningen anger att kolumnen ingår i primärnyckeln. Innebörden i typ/formatangivelsen framgår av kapitel 2.

Hur Du startar exemplet framgår av inledningen till bilaga F.

3.1 Tabeller i databanken HOTELLDB

Tabellnamn	*Kolumn*	*Typ*	*Format*	*Innehåll*
HOTELL	HID	*C	3	Identifikation (hotellnr)
	HNAMN	C	20	Hotellnamn
	ADRESS	C	20	Hotellets utdelningsadress
	POSTNR	C	6	Postnummer
	POSTADR	C	15	Ortsadress
	TELE	C	15	Hotellets telefonnr
	STANDARD	C	5	Hotellklass
	CHEF	C	5	Kod för hotellchefen
	BÄDDAR	C	4	Antal bäddar

NÖJEN	HID	*C	3	Identifikation (hotellnr)
	VERKSAMH	*C	20	Speciella aktiviteter på hotellet
	STARTMÅN	C	2	Nr på månad när aktivi-
	SLUTMÅN	C	2	teter börjar resp slutar
	ÖVRIGT	C	30	Allmänna noteringar om nöjesanläggningar o dyl
PERSONAL	HID	*C	3	Identifikation (hotellnr)
	ANSTNR	*C	5	Anställningsnummer
PRIS	PRISKOD	*C	4	Kod för visst rumspris
	PRIS	I	2	Mot koden svarande pris
RUM	HID	*C	3	Identifikation (hotellnr)
	RUMSNR	*C	6	Rummets nummer
	TYP	C	15	Rumstyp (enkel, dubbel etc)
	PRISKOD	C	4	Rummets priskod
	INFO	C	40	Speciella uppgifter om rummet

3.2 Tabeller i databanken ANSTDB

Tabellnamn	*Kolumn*	*Typ*	*Format*	*Innehåll*
ANSTÄLLD	ANSTNR	*C	5	Anställningsnummer
	ANAMN	C	20	Den anställdas namn
	ADRESS	C	20	Den anställdas utdel- ningsadress
	POSTNR	C	6	Postnummer
	POSTADR	C	15	Den anställdas ortsadress
	ANKN	I	2	Anknytningsnr i växeln
	ANSTDAT	C	6	Anställningsdag
	BEFATTN	C	20	Tjänstebeteckning
	LÖNKOD	C	3	Kod för lönebelopp
LÖN	LÖNKOD	*C	3	Kod för lönebelopp
	LÖN	I	2	Lönebelopp svarande mot lönkod

4 Strukturen i MIMER/QL

MIMER/QL är ett fråge- och kommandospråk. Själva QL-språket består av ett 30-tal olika kommandon, ibland för klarhetens skull benämnda *QL-kommandon*. Därtill kommer ett 10-tal s k *procedurkommandon* och ett 30-tal *procedureditorkommandon* (kapitel 13). Den beskrivning som ges i det här kapitlet, förutsätter att MIMER/QL används interaktivt.

Kommandona används för att:

- Definiera, ändra och ta bort användare, databanker, användarställning och personlig behörighet.
- Definiera, ändra och ta bort tabeller.
- Söka och hämta data i befintliga tabeller.
- Lägga till data i befintliga tabeller.
- Ta bort och ändra data i befintliga tabeller.
- I förväg definiera ett antal kommandokedjor, som sedan kan återanvändas vid behov utan att skrivas in på nytt, s k *procedurer*.

4.1 Systemstruktur

Med *systemstruktur* menas här sambandet mellan MIMER/QL och de olika filer med vilka MIMER/QL kan kommunicera direkt eller via MIMER/DB. Kommunikationsprocessen innebär att antingen kommandon eller data – eller rättare uttryckt kopior av data – överförs.

MIMER/QL tar emot och översätter kommandon, som matas in via tangentbordet och utför därefter de begärda operationerna på data i databasen.

MIMER/QL är byggt kring den i kapitel 2 beskrivna databashanteraren MIMER/DB. MIMER/DB använder den information som finns i databankerna SYSDB, TRANSDB och LOGDB. SYSDB innehåller information om databanker, användare och användarbehörigheter (kapitel 2). SYSDB måste finnas för att systemet skall fungera. Om transaktions- och loggningsmöjligheterna skall användas, måste även databankerna TRANSDB respektive LOGDB finnas tillgängliga (avsnitt 2.5).

Databanken SYSDB definieras med hjälp av ett speciellt hjälpprogram eller med kommandon i MIMER/QL. För att kunna påverka alla delar av innehållet i SYSDB krävs att man har användarställning **X**, dvs ställning som databasadministratör (kapitel 2,17).

Endast de användare, som är definierade i SYSDB, kan använda MIMER/DB och MIMER/QL. Endast de databanker som är definierade i SYSDB kan användas av MIMER/QL (via MIMER/DB).

Det finns ytterligare en databank i systemet, SYSQL. Den innehåller bl a felmeddelanden och hjälptexter som kan tas fram på bildskärmen. Även denna information är lagrad i MIMER-tabeller.

Kommandot HÄMTA använder sig i vissa fall av en speciell databank, ARBDB. Denna *arbetsdatabank* används för att lagra och bearbeta tillfälliga datamängder.

I MIMER/QL finns som antytts möjligheter att bygga upp sk procedurer. En *procedur* innehåller ett antal QL-kommandon och/eller särskilda *procedurkommandon*. När man sedan anropar proceduren under dess speciella namn med ett speciellt kommando, kommer kommandona i proceduren att utföras utan att behöva skrivas in på nytt.

Även procedurerna lagras i tabeller i en databank. Systemets standardnamn på denna databank är PROCDB.

Det är också möjligt att i förväg i en vanlig sekvensiell fil lagra *initieringskommandon*, dvs ett antal kommandon som alltid ges vid start av ett arbetspass med MIMER/QL. Detta görs med hjälp av det *editeringsprogram*, som finns i systemet. Vi kallar här denna fil för INITFIL, men det verkliga filnamnet bestäms av vilket datorsystem som används (bilaga A). Om det finns en initieringsfil, läser MIMER/QL in den och utför de kommandon, som finns i filen. När filslutsmarkeringen (END-OF-FILE, EOF) påträffas, stängs filen och MIMER/QL väntar sig nu att få ett kommando från terminalen på vanligt sätt.

Vissa kommandon använder sig av en *skrivfil*, som också är en sekvensiell fil. Denna har här getts namnet SKRIVFIL (för information om det verkliga filnamnet, se bilaga A). Innehållet i filen är formaterat för utskrift på skrivaren.

MIMER/QL kan också kommunicera med andra sekvensiella filer innehållande data, som man önskar läsa in i MIMER-tabeller. Omvänt kan data överföras från MIMER-tabeller till sekvensiella filer.

För att sammanfatta – MIMER/QL kommunicerar med följande externa filer:

- Terminalen (tangentbordet och bildskärmen)
- Databanker och sekvensiella filer, som har definierats av användare
- INITFIL
- SKRIVFIL
- SYSQL
- PROCDB
- ARBDB
- SYSDB
- TRANSDB
- LOGDB

Figur 4:1 visar systemstrukturen i MIMER/QL schematiskt.

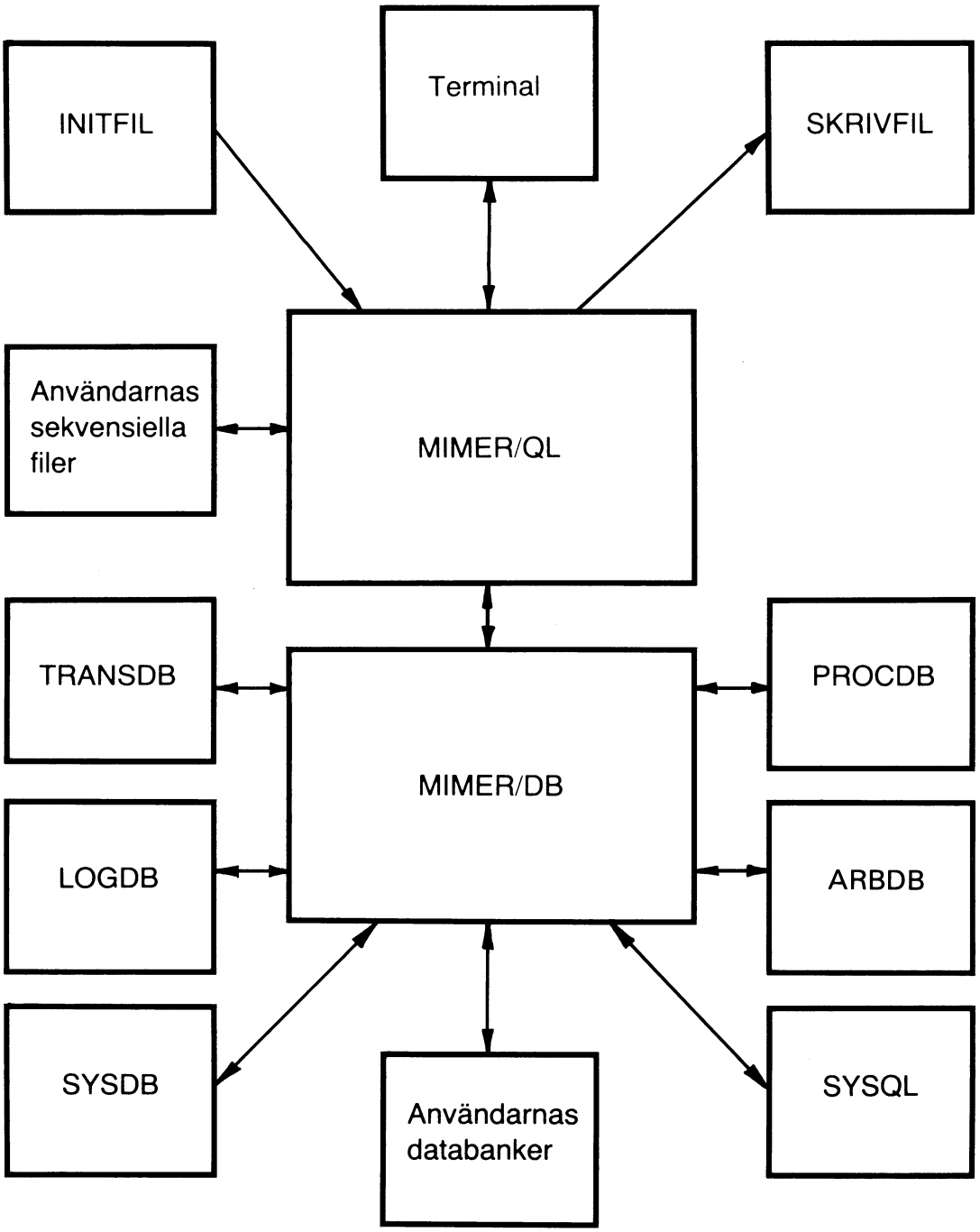


Fig 4:1. Systemstrukturen i MIMER/QL.

5 MIMER/QLs syntax

Med ett språks *syntax* menas de regler som anger hur språkets ord och tecken formas till meningsfulla satser. Man kan också uttrycka det så att syntaxen ger mönster för sättet att skriva språkets satser.

Liksom varje annat dataspråk har också MIMER/QL sina bestämda syntaxregler, som anger hur ett kommando skall skrivas för att fungera. Dessa regler beskrivs i det här kapitlet. Reglerna är utformade som mönster, vilka skall följas exakt vid inskrivningen av kommandona.

I det här kapitlet beskrivs bl a hur dessa mönster är utformade. Om du inte är van vid den här typen av beskrivningar, kan du möjligen hoppa över kapitlet vid första genomläsningen av handboken och i stället gå tillbaka till det senare, t ex när du vill tolka innebörden i en viss syntaxbeskrivning. Förståelse av hur syntaxmönstren skrivs underlättar dock avsevärt fortsatt läsning av handboken. I figur 5:2 i slutet av detta kapitel återges syntaxen för ett mer komplicerat kommando, med kommentarer.

5.1 Kommandoord, parametrar och skiljetecken

Fig 5:1 visar ett enkelt *kommando* i MIMER/QL. Det består av *kommandoord*, *parametrar* och *skiljetecken*.

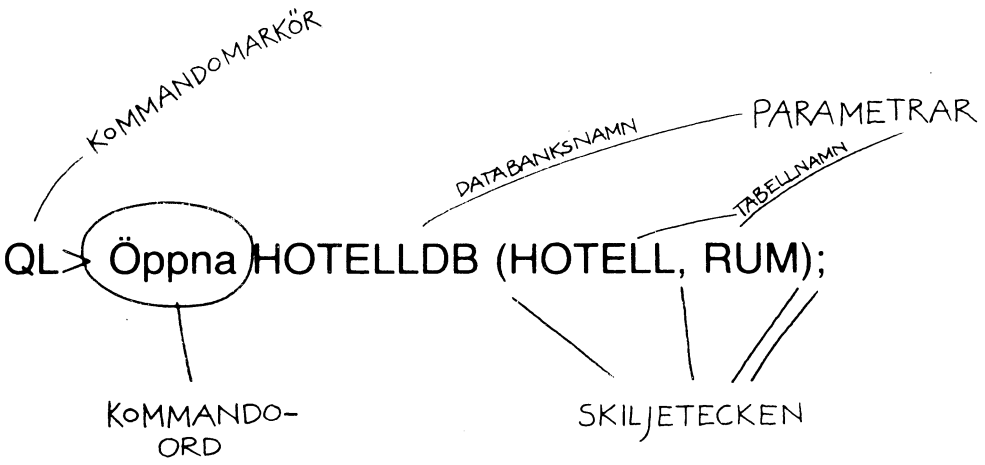


Fig 5:1. Ett enkelt kommando i MIMER/QL.

Kommandoorden identifierar kommandot och utgör dess *fasta del*. Exempel på ytterligare kommandoord är DEFINIERA TABELL och KOPIERA ... FRÅN, som är sammansatta ord. Kommandoorden kan skrivas i förkortad form. Normalt behöver endast de tre första bokstäverna i ordet anges. Man kan således skriva DEF TAB. Förkortningarna framgår av syntaxbeskrivningarna.

Parametrarna är kommandots *variabla* del och utgörs exempelvis av namn på databanker, tabeller eller kolumner. Parametern kan också vara ett *kommandotillägg* – i kommandot BESKRIV TABELL/S är /S ett sådant tillägg (anger att data skall placeras i skrivfilen).

Kommandoord och parametrar avgränsas med något av följande skiljetecken

Tecken	*Teckennamn*
.	punkt
,	komma
(	vänsterparentes
)	högerparentes
:	kolon
/	snedstreck
'	apostrof
"	citationstecken
=	likhetstecken
	mellanslag

Om ett skiljetecken måste användas inne i en parameter, skall det omges av apostrofer (') eller citationstecken (")(figur 5:1). Detta gäller dock inte när numeriska värden anges.

I de speciella syntaxbeskrivningarna i denna handbok anges kommandoord med fet stil och hela ordet skrivs ut, varvid den förkortade formen markeras med stora bokstäver, tex

ÖPPna databanksnamn;

5.2 Kommandomarkören

När MIMER/QL väntar sig ett kommando, visas en sk *kommandomarkör* (ibland kallad *kommandoprompt* efter motsvarande engelska term) på bildskärmen. Markörens utseende varierar med datorsystem. I tabellen som innehåller datorberoende information (bilaga A), anger variabeln PROMPTER vilken symbol som används. I handboken används QL> som kommandomarkör.

Om ett kommando sträcker sig över flera rader på skärmen måste det sista tecknet på varje rad vara ett skiljetecken. Symbolen +> skrivs av systemet i början på varje fortsättningsrad.

5.3 Avslutning av kommando, mellanslag

Varje kommando skall avslutas med semikolon (;) och vagnretur.

Man kan skriva obegränsat antal mellanslag mellan kommandoord, skiljetecken och parametrar utan att det påverkar kommandots funktion. Följande kommandon ger således samma resultat:

```
QL> Öppna BANK(TABELL1,TABELL2);
QL> Öpp     BANK     (TABELL1,
+>           TABELL2);
```

5.4 Uttryck som kan utelämnas

Om ett uttryck i en kommandobeskrivning omges av tecknen < ("mindre än") och > ("större än") kan uttrycket utelämnas. Exempel:

```
QL> Beskriv alias <tabellnamn>;
```

Kommandot BESKRIV ALIAS accepterar ett tabellnamn som parameter. Ett *alias* är i detta sammanhang ett alternativt, förkortat tabellnamn som användaren bestämt. Ett alias består av ett eller två alfanumeriska tecken. Inget hindrar att ett tabellnamn har flera alias. Med kommandot ovan visas på skärmen endast de alias som är definierade för den aktuella tabellen. Om inget tabellnamn anges visas alla alias som definierats för samtliga tabeller.

Kommandot ALIAS används för att koppla ihop alias med tabeller. I följande exempel kopplas tre alias – A, A1 och MT – till tabellen MINTAB:

```
QL> Alias MINTAB(A,A1,MT);
```

Ang ALIAS-kommandot, se vidare avsnitt 5.6 och bilaga C.

5.5 Namn och lösenord

Namn på användare, databanker, tabeller och kolumner får bestå av högst 8 alfanumeriska tecken (bokstäverna A-Ö och siffrorna 0–9). Namnen måste dock alltid inledas med en bokstav. Lösenord får likaledes bestå av högst 8 tecken – här tillåts dock vilka tecken som helst.

Tabellen nedan ger exempel på giltiga och ogiltiga namn och lösenord.

Parameter	*Giltig*	*Ogiltig*
Användar-namn	TOMMY, ANV0001, LENA, DBA, MARTIN, SALLY	1099, 1ANV, ANV 1, TIM%, A+:–()
Databanks-namn	SYSDB, SYSQL, ANVDB, MINDB, DB00001	1BANK, MIN BANK, 4587, DB(-;)?
Tabellnamn	ANSTÄLLD, ADRESS, MINTAB, TABELL99	2TABELL, HANS ADR, SUM+(SUM), 1022
Kolumnnamn	NAMN, AVDELNNR, SEKVNR, KOL123, TEMP	SYS&KOL, AVDELN , 123KOL, FÖRLÅNGTNAMN
Lösenord	HEMLIG, abc+++, 5598, Hallå, där	

Ett filnamn skall följa syntaxen för det filhanteringssystem, som används i det datorsystem där MIMER/QL körs.

5.6 Tabellreferens

En *tabellreferens*, vanligen förkortat *tabref*, är antingen ett tabellnamn eller ett alias. Exempel:

```
QL> Alias MINTAB(MT);
QL> Hämta MINTAB.KOLUMN1, MINTAB.KOLUMN2;
QL> Hämta MT.KOLUMN2;
```

ALIAS-kommandot kopplar här ihop alias MT med tabellen MINTAB. HÄMTA– kommandot läser data från tabeller och presenterar data på skärmen. Man måste specificera från vilka tabeller och vilka kolumner data ska hämtas. När man specificerar tabellerna kan man antingen använda tabellnamnet, MINTAB, eller dess alias, MT.

5.7 Formatspecifikation

I avsnitt 2.2 beskrevs formatbegreppet. En *formatspecifikation* definierar hur data lagras i kolumnen. Formatspecifikationen består av uttrycket Tn där T är lagringstypen och n är lagringsutrymmet. T kan ha ett av värdena C (alfanumeriska tecken), I (heltal) eller F (flyttal). Beroende på lagringstyp och vilket datorsystem som används, varierar värdet på n.

Lagringstyp (T)	*Lagringsutrymme (n)*
C	n = 1 – min(MMRL, MMRL – sum(kolumnlängder)) *)
I	n = 1 – ILÄNGDB
F	n = 3 – FLÄNGDB

*) Det minsta värdet på MMRL och (MMRL minus summan av storleken på kolumnerna).

MMRL betyder maximal radlängd, ILÄNGDB maximal längd på data av typ I och FLÄNGDB maximal längd på data av typ F.

De verkliga värdena på MMRL, ILÄNGDB och FLÄNGDB presenteras på skärmen med hjälp av ett speciellt kommando (bilaga A).

5.8 Alternativa parametrar

Om parametrar åtskiljs med tecknet |, innebär det att parametrarna utgör alternativ av vilka ett väljs.

Uttrycket /T|/S|/A används sålunda i vissa kommandon för att ange till vilken fysisk enhet i datorsystemet som data skall skickas. Vi har redan tidigare i det här kapitlet stött på denna typ av *kommandotillägg*. /T innebär att data skickas till terminalen, /S innebär att utmatning görs till en skrivfil och /A innebär att data skickas till både terminalen och skrivfilen. Om utenhet inte anges, skickas data till terminalen. Exempel:

```
QL> Beskriv/A tabell MINTAB;
```

Kommandot BESKRIV TABELL presenterar en beskrivning (namn och format) av alla kolumner i tabellen MINTAB. Utmatningen kan styras till terminal och/eller till fil. I exemplet skickas utskriften till både terminal och fil genom att parametern /A har angivits.

5.9 Villkorssatser, operatorer

En *villkorssats* är ett uttryck som definierar *sökvillkor* och *kopplingssväg* för dataåtkomst (beträffande dessa begrepp, se vidare avsnitt 8.3). Villkorssatsen begränsar antalet tabellrader som kommer att behandlas vid sökningen. En villkorssats skrivs enligt följande mönster:

Där tabref.kolumnnamn **ROP** "värde"

DÄR är själva markeringen av att en villkorssats föreligger. Tabref avser tabellnamn eller alias. Kolumnnamn är beteckningen på kolumn i en tabell.

ROP betyder *relationsoperator*. Vilka relationsoperatorer som finns i MIMER/QL framgår av nedanstående tabell. I tabellens andra kolumn anges en äldre, alternativ beteckning, vilken också godtas av systemet (beteckningen utgör en förkortning av motsvarande svenska uttryck):

Operator anges med	*eller med*	*Innebörd*	
=	LM	lika med	
>=	SL	större än eller lika med	
>	SÄ	större än	
<=	ML	mindre än eller lika med	
<	MÄ	mindre än	
/=	EL	ej lika med	
*	IN	innehåller	endast vid kolumntyp C
/*	EI	innehåller ej	
%	BM	börjar med	
/%	EB	börjar ej med	

Med relationsoperatorn jämförs de värden, som står på ömse sidor om operatorn. "Värde" till höger om ROP kan antingen vara ett bestämt värde (en konstant, tex en teckensträng eller ett numeriskt värde) eller tabref.kolumnnamn.

Uttrycket *tabref.kolumnnamn ROP "värde"* kallas för *relationsuttryck*. Flera relationsuttryck kan ingå i en villkorssats. De sammankopplas i så fall med hjälp av de *logiska operatorerna* (LOP) vilka är OCH och ELLER enligt mönstret:

relationsuttryck **LOP** relationsuttryck

Föreliggande version av MIMER/QL har ej sk *självoptimerande* val av kopplingsvägar till tabeller utan dessa måste definieras i villkorssatsen (kapitel 8).

Den tabellkolumn som man ställer villkor på, skall anges *till vänster* om ROP. Den kolumn eller det värde, som utgör jämförelsevärdet, ska stå *till höger* om ROP. Parenteser är inte tillåtna i villkorssatser.

Här följer exempel på villkorssatser:

```
Hämta ... där TABELL1.KOLUMN1 = 'abc';
Hämta ... där TABELL1.KOLUMN2 >= 2
           och TABELL1.KOLUMN2 <= 4;
Hämta ... där TABELL1.KOLUMN1 = TABELL2.KOLUMN5
           och TABELL1.KOLUMN2 = 10;
```

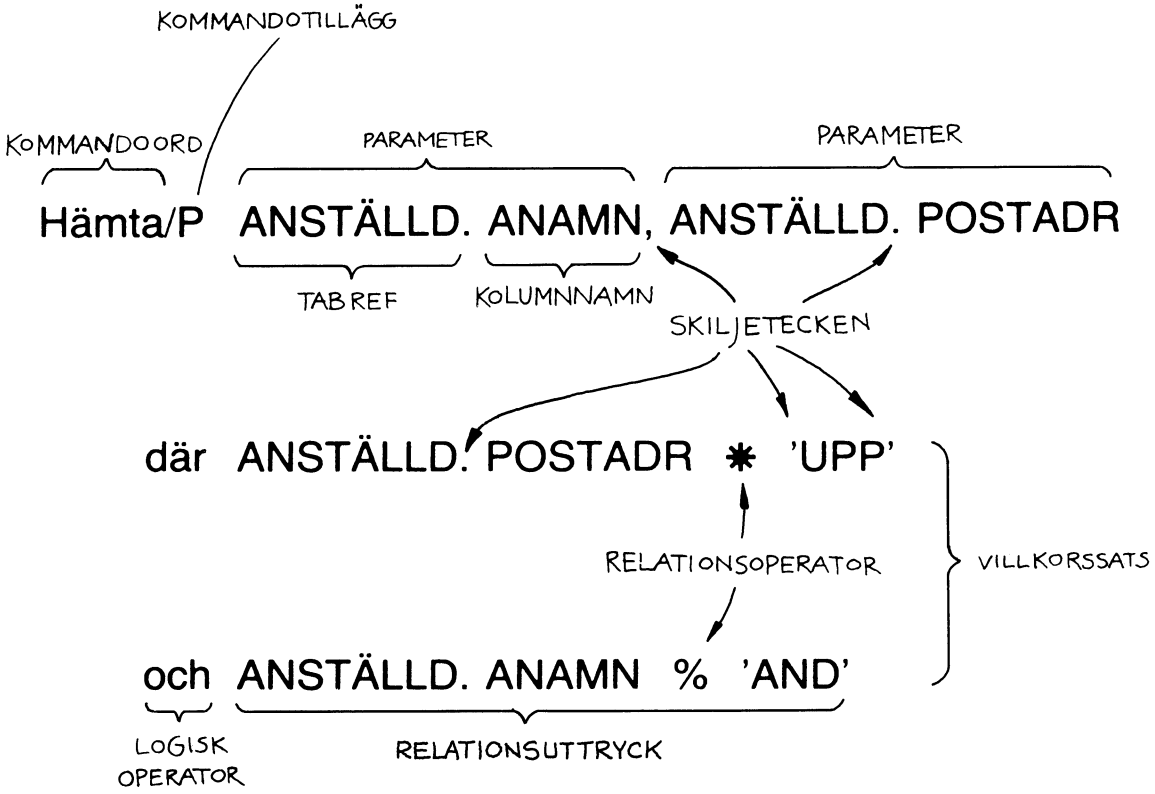


Fig 5:2. Exempel på syntax för ett MIMER/QL-kommando.

6 Olika typer av kommandon

Kommandona i MIMER/QL kan delas in i följande grupper:

- Kommandon som söker, hämtar och påverkar data (datapåverkande)
- Kommandon som definierar och beskriver data
- Kommandon som definierar och beskriver databaser
- Kommandon som definierar och beskriver vilka villkor, som skall gälla under ett arbetspass med MIMER/QL
- Övriga kommandon

Här ges en översikt över de olika kommandotyperna.

6.1 Datapåverkande kommandon

Datapåverkande kommandon påverkar inte själva tabellstrukturen utan bara data i tabellerna – data läggs till, tas bort eller ändras (uppdateras).

Det finns tre kommandon som *påverkar tabellraderna* och ett kommando som *kopierar tabeller*:

Kommando	*Funktion*
LAGRA	En ny rad läggs in i tabellen
ÄNDRA	En eller flera rader uppdateras
RADERA	En eller flera rader tas bort
KOPIERA	En hel tabell kopieras från en fil

KOPIERA-kommandot kan också användas för att *söka och hämta data*. Kommandot

Kopiera TABELL **till**

kopierar en hel tabell till en sekvensiell fil. Andra kommandon som söker och hämtar data är HÄMTA och SKRIV.

6.2 Kommandon som definierar och beskriver data

Dessa kommandon används för att definiera, ändra, ta bort och beskriva tabeller och index. DEFINIERA TABELL, DEFINIERA INDEX, OMDEFINIERA,

ELIMINERA TABELL och ELIMINERA INDEX är kommandon av denna typ. Kommandot BESKRIV TABELL presenterar en beskrivning av tabelldefinitionerna.

När en tabell definieras bestämmer man namn på kolumnerna, vilka kolumner som skall ingå i primärnyckeln och enligt vilket format data ska lagras (datatyp). Kolumnnamnen åtskiljs med kommatecken (,). Primärnyckelns slut markeras dock i stället med kolon (:) efter namnet på den sista kolumn som ingår i nyckeln. Sätts inget kolon bildar samtliga kolumner primärnyckel.

Data kan som framgått i kapitel 2 lagras med tre olika format: C (alfanumeriska tecken), I (heltal) eller F (flyttal). Alfanumeriska data lagras i format C. Numeriska heltal kan lagras antingen i format C eller format I. Numeriska decimaltal kan antingen lagras i format C eller format F.

Vid teckenlagring (C) lagras data som en teckensträng där varje tecken placeras i ett bestämt antal byte. Vid heltals- eller flyttalslagring (I respektive F) lagras de numeriska värdena i en maskinberoende internkod.

Ta som exempel att talet 12345 skall lagras:

Format	*Minnesutrymme*
C	5 byte
I	2 byte
F	4 byte

Att lagra talet med I- eller F-format är alltså mindre utrymmeskrävande än med C-format, något som gäller i de flesta fall.

Hur en *formatspecifikation* skrivs framgår av kapitel 5.

Data som lagras med I- eller F-format omvandlas till C-format innan de presenteras på skärmen. Ett tal behöver som framgått ofta mer utrymme i C-format än i I- eller F-format. Systemet reserverar det utrymme som krävs för att det största möjliga talet skall få plats. Värdet beror på kolumnstorleken. Storleken på det utrymme som reserveras är systemberoende men kan ändras med kommandot SÄTT.

Om inga data lagras i en kolumn eller en rad (eller flera rader), säger man att kolumnens innehåll är *odefinierat*. När man tittar på en sådan kolumn på skärmen finner man att den är markerad med punkter (...).

6.3 Kommandon som definierar och beskriver databaser

Dessa kommandon används för att definiera, ändra, ta bort och beskriva databanker, användare och användares personliga behörighet till olika databanker. Kommandot DBA används för detta ändamål. Endast vissa användare – DBA eller användare med ställning **X** – kan utföra operationerna definiera, ändra och ta bort.

6.4 Kommandon som definierar och beskriver villkoren för ett arbetspass

De *definierande* kommandona i denna grupp används för att öppna databankerna samt definiera tabellreferenser och tillståndsvariabler för ett arbetspass. De är:

ÖPPNA databank
STÄNG databank
ALIAS tabellnamn
SÄTT tillståndsvariabler|systemberoende variabler
ÅTERSTÄLL tillståndsvariabler|systemberoende variabler

Med ÅTERSTÄLL återställs tillståndsvariabler och systemberoende variabler till sina ursprungliga värden.

De *beskrivande* kommandona används när man vill ta reda på vilka definitioner, som gäller för ögonblicket. De är:

BESKRIV MILJÖ
BESKRIV SÄTTNINGAR
BESKRIV ALIAS

6.5 Övriga kommandon

Kommandon som inte kan hänföras till någon av ovanstående kommandogrupper ingår i denna grupp:

Kommando	*Funktion*
HJÄLP	Visar en syntax eller en förklaring av ett felmeddelande
UTFÖR	Utför kommandona i en tidigare definierad procedur
SLUTA	Avbryter körningen
EDITERA	Startar editeringsprogrammet, som används för att editera procedurer
STATISTIK	Startar statistikpaketet MIMER/ST
GRAFIK	Startar grafikpaketet MIMER/GR
KOMMANDO	Lägger till funktioner som inte finns i MIMER/QL

7 Start, identifiering av användare

Start av MIMER/QL går till på olika sätt i olika datorsystem. I bilaga A beskrivs hur man får fram information om startproceduren, liksom annan datorsystemberoende användarinformation.

7.1 Användarnamn och lösenord

När MIMER/QL har startats uppmanas användaren att mata in sitt användarnamn och lösenord. Dessa har lagts in i systemet av databasadministratören. Förutsatt att rätt namn och lösenord getts kan MIMER/QL nu börja användas. Vid fel svar får ett nytt försök göras, men efter tre misslyckade försök avslutas körningen automatiskt.

Användaren har tilldelats användarställning **S** (vanlig användare) eller **X** (databasadministratör). Databasadministratören kan exempelvis definiera och ta bort databanker och användare, som skall utnyttja systemet på olika nivåer. Sådana operationer kan inte utföras av en användare med ställning S. Alla användare kan dock ändra sitt eget lösenord (med DBA-kommandot), dock med ett undantag: en användare med ställning S som kör i enanvändarsystem kan inte ändra sitt eget lösenord.

7.2 Användning av INITFIL

Identifieringen av användaren kan också göras genom att man använder filen INITFIL (kapitel 4). Den första och andra posten i INITFIL ska skrivas enligt följande mönster:

Användare: "användarens namn"
<Lösenord: "lösenord">

eller med ett konkret exempel:

Användare: KING
Lösenord: KONG

Kolon (:) måste här åtföljas av minst ett mellanslag. Posten lösenord kan utelämnas i INITFIL, men då måste lösenordet i stället matas in via tangentbordet.

Observera att lösenordet av säkerhetsskäl inte visas på skärmen när det skrivs in.

7.3 Användning av vagnretur

Den som inte är så van vid databehandling får här en påminnelse om den viktiga funktionen hos tangentbordets tangent för vagnretur (vanligen betecknad RETURN, CARRIAGE RETURN, CR, el dyl). Den används för att bekräfta att en inmatning i MIMER/QL skall ske.

När användarnamnet skrivits in måste det följas av en vagnretur. Först när detta skett tar systemet emot ordet och undersöker om en användare med uppgivet namn finns. Samma sak gäller lösenord.

Vagnretur som bekräftelse används också när ett kommando eller en procedur skall utföras samt i samband med inmatning eller ändring av data.

7.4 Stora eller små bokstäver

När kommandon eller parametrar skrivs in har det ingen betydelse om det sker med stora eller små bokstäver. Systemet uppfattar ändå vad som avses.

8 Hur data hämtas från databasen

8.1 Deklarationer

För att man skall kunna hämta eller uppdatera data i databasen, måste man specificera vilka databanker och tabeller man vill arbeta med, dvs man måste öppna databanker och tabeller. Detta brukar kallas för *deklaration* och görs med kommandot ÖPPNA, som har följande syntax:

ÖPPna</**R**|/**S**|/**X**>databanksnamn<(tabellnamn,...)>,...;

Kommandot medför att angivna tabeller i angiven databank blir tillgängliga. Om tabellnamn utelämnas, blir samtliga tabeller i den angivna databanken tillgängliga.

Med parametrarna /R, /S och /X anger man vilken typ av operationer som skall kunna utföras under arbetspasset. /R betyder läsning (read-only), /S läsning och skrivning (shared access) och /X exklusiv användning (exclusive update). Om parameter utelämnas gäller /R. Om /X anges, kan ingen annan användare samtidigt arbeta med den specificerade databanken. Exempel:

```
QL> Öppna HOTELLDB,ANSTDB(LÖN);
QL> Öppna/R ANSTDB;
```

Om man är osäker på vilka tabeller som ingår i databanken, kan man använda kommandot BESKRIV DATABANK för att ta reda på det. Dess syntax är:

BESkriv</**T**|/**S**|/**A**>**DATabank** databanksnamn;

Parametrarna /T, /S och /A används för att styra presentationen till en speciell enhet. /T innebär terminal, /S skrivfil och /A både terminal och skrivfil. Om uppgiften utelämnas gäller /T. Exempel:

```
QL> Beskriv databank HOTELLDB;
QL> Beskriv/T databank ANSTDB;
QL> Beskriv/S databank HOTELLDB;
QL> Beskriv/A databank ANSTDB;
```

Med kommandot ALIAS kan man tilldela varje tabell ett alias, dvs ett alternativt namn (en förkortning). Kommandots syntax är:

ALIas tabellnamn (alias,...)<,tabellnamn (alias,...)>;

Exempel på ALIAS-kommandon:

```
QL> Alias ANSTDB (A1);
```

Med STÄNG-kommandot stängs databanker och tabeller, varvid de upphör att

vara tillgängliga. Syntaxen för STÄNG-kommandot är den samma som för ÖPPNA-kommandot.

Vi belyser det som sagts i avsnittet med att visa hur deklarationen kan gå till när en användare på MIMER HOTELL AB går in och arbetar med databanken HOTELLDB. Först vill användaren ha reda på vilka tabeller som finns i databanken:

```
QL> Beskriv databank HOTELLDB;

Databank: HOTELLDB          Fil: HOTELLDB.DBF
    Access                      Storlek
Allmän   Användare          Begärd  Använd
-------------------------------------------
   S          X              24       5
Tabeller:

HOTELL    NÖJEN    PERSONAL   PRIS    RUM
```

Databanken HOTELLDB innehåller alltså fem tabeller: HOTELL, NÖJEN, PERSONAL, PRIS och RUM. Databanken finns i filen med namnet HOTELLDB.DBF. Den generella behörigheten är S (shared access) medan den aktuella användaren har personlig behörighet X (exclusive). När databanken definierades, reserverades 24 sidor. Hittills har 5 sidor utnyttjats. Vi öppnar nu en databank:

```
QL> Öppna/S ANSTDB;
```

Kommandot medför att alla tabeller i databanken ANSTDB blir tillgängliga för läsning och skrivning (/S = shared access). Vi fortsätter och öppnar ett par tabeller i databanken HOTELLDB:

```
QL> Öppna HOTELLDB (NÖJEN,HOTELL);
```

Tabellerna NÖJEN och HOTELL i databanken HOTELLDB öppnas, dock endast för läsning. Om man senare skulle vilja skriva i dessa tabeller, måste man först stänga tabellerna med STÄNG-kommandot och därefter ge ett ÖPPNA-kommando med tillägget /S:

```
QL> Stäng HOTELLDB;
```

Samtliga tabeller i databanken HOTELLDB stängs. Därefter öppnar vi dem på nytt, nu med skrivbehörighet:

```
QL> Öppna/S HOTELLDB;
```

Vi går vidare och definierar :

```
QL> Alias PERSONAL (A1);

QL> Alias HOTELL (H);
```

Alias definieras för tabellerna PERSONAL (A1) och HOTELL (H).

8.2 Att hämta data från tabellerna

För att kunna hämta data från en tabell, måste man känna till vilka kolumner tabellen innehåller. Om man är osäker, kan man använda kommandot BESKRIV TABELL för att lista namnen på alla kolumner i tabellen. Kommandots syntax är:

BESkriv</**T**l/**S**l/**A**>></**KOR**tl/**FULI**> **TABell** tabellnamn;

Kommandot visar på skärmen en beskrivning av alla kolumner i den angivna tabellen. Om /FULL anges, visas även information om radlängd och aktuellt antal rader i tabellen. /KORT medför att sådan information inte presenteras. /KORT är standardvärde och gäller alltså om ingen parameter med avseende på rader ges in. Exempel:

```
QL> Beskriv tabell RUM;
QL> Beskriv/A tabell PERSONAL;
QL> Beskriv/Full tabell PRIS;
QL> Beskriv/Kort tabell HOTELL;
```

För att hämta data från tabeller används kommandot HÄMTA, vars syntax är:

HÄMta </TI/SI/A> tabref.kolumnnamn,tabref.kolumnnamn,...;

Man måste ange namnen på de kolumner från vilka man vill hämta data. Man måste också ange vilka rader, som skall användas (avsnittet 8.2.1).

Kolumner specificeras på följande sätt:

tabref.kolumnnamn | tabref.*

Tabref kan vara ett alias eller ett tabellnamn. När man vill att alla kolumnerna i en tabell skall visas, skriver man en asterisk (*) istället för kolumnnamn.

Exempel på det sagda är följande kommandon:

```
QL> Alias HOTELL(H);
QL> Hämta/T H.HNAMN;
QL> Hämta HOTELL.*;
QL> Hämta H.HNAMN, H.POSTADR, H.STANDARD ;
```

I ovanstående exempel används /T för att styra utskriften till skärmen. Man kan även styra utskriften till en skrivfil (med /S) eller till både terminal och skrivfil (med /A). Om man utelämnar utmatningsparametern gäller /T. Presentation och utskrifter behandlas utförligt i avsnitt 8.4.

Om kolumnen är tom, dvs inte innehåller något värde, visas punkter (...) på skärmen. Observera att blanktecken betraktas som värde.

Här följer nu några övningsexempel som belyser det ovan sagda.

Uppgift: Vilka kolumner ingår i tabellen RUM?

```
QL> Beskriv tabell RUM;
```

RUM	*1*	*HID*	**C*	*3*
	2	*RUMSNR*	**C*	*6*
	3	*TYP*	*C*	*15*
	4	*PRISKOD*	*C*	*4*
	5	*INFO*	*C*	*40*

Uppgift: Hämta alla rumsnummer i tabellen RUM.

```
QL> Hämta RUM.RUMSNR;
```

RUMSNR
R101
R102
R101
R102

R103
R101
R102
R103
R104
R101
R102
R101
R102
R101
R102

15 rader hittade

Uppgift: Hämta alla kolumner och rader i tabellen RUM.

QL> **Hämta** RUM.*;

HID	*RUMSNR*	*TYP*	*PRISKOD*	*INFO*
H01	*R101*	*ENKEL*	*P40*	*FÄRG-TV, BAD*
H01	*R102*	*DUBBEL*	*P60*	*FÄRG-TV, BAD*
H02	*R101*	*DUBBEL*	*P60*	*DUSCH*
H02	*R102*	*DUBBEL*	*P70*	*FÄRG-TV, BAD, KYLSKÅP*
H02	*R103*	*DUBBEL*	*P80*	*FÄRG-TV, DUSCH, MINIBAR*
H03	*R101*	*ENKEL*	*P70*	*FÄRG-TV, DUSCH, HAVSUTSIKT*
H03	*R102*	*DUBBEL*	*P90*	*VIDEO, DUSCH, MINIBAR*
H03	*R103*	*SVIT*	*P90*	*FÄRG-TV, DUSCH, TRE RUM*
H03	*R104*	*SVIT*	*P90*	*FÄRG-TV, POOL, FYRA SMÅ RUM, HAVSUTSIKT*
H04	*R101*	*ENKEL*	*P10*	*DUSCH I KORRIDOR*
H04	*R102*	*ENKEL*	*P20*	*DUSCH I KÄLLAREN, EXTRA SÄNG*
H05	*R101*	*DUBBEL*	*P80*	*FÄRG-TV, MINIBAR, HAVSUTSIKT*
H05	*R102*	*SVIT*	*P90*	*TRE RUM, MINIBAR, HAVSUTSIKT*
H06	*R101*	*ENKEL*	*P30*	*DUSCH*
H06	*R102*	*DUBBEL*	*P50*	*FÄRG-TV*

15 rader hittade

8.2.1 Villkorlig hämtning

Sista exemplet i föregående avsnitt visar hur man hämtar *alla* rader i en tabell. Om man bara vill ha med vissa rader, måste man specificera villkoren för att enbart dessa rader skall tas med, s k *villkorlig hämtning*. Man kan exempelvis välja ut uppgifter från och med ett visst datum, för en speciell personalkategori eller för en produktgrupp. Detta görs med en villkorssats inledd med DÄR, vilken läggs in i HÄMTA-kommandot. Syntaxen är följande:

HÄMta tabref.kolumn,...<**DÄR** villkorssats>;

Villkorssatsen består av ett eller flera sammankopplade uttryck enligt mönstret:

tabref.kolumnnamn ROP 'värde' <LOP...>

Om flera delvillkor ingår i villkorssatsen, måste de kopplas ihop med hjälp av de *logiska operatorerna* (LOP) OCH och ELLER.

ROP är en *relationsoperator*, som anger hur jämförelsen mellan uttrycken på ömse sidor om ROP skall göras. "Värde" till höger om ROP kan antingen vara ett bestämt värde (en konstant i form av en teckensträng eller ett numeriskt värde) eller ett uttryck av formen tabref.kolumnnamn. Det förra kallas för *explicit villkor*, det senare för *implicit villkor*. Med "värde" ställs villkor enligt ROP på kolumnen till vänster om ROP.

Relationsoperatorerna förtecknas nedan. I tabellens andra kolumn anges en äldre, alternativ beteckning, vilken också godtas av systemet (beteckningen utgör en förkortning av motsvarande svenska uttryck).

Följande relationsoperatorer kan användas vid alla kolumntyper (C,I och F):

Operator anges med	*eller med*	*Innebörd*
=	LM	lika med
>=	SL	större än eller lika med
>	SÄ	större än
<=	ML	mindre än eller lika med
<	MÄ	mindre än
/=	EL	ej lika med

Vid kolumntyp C finns dessutom möjlighet att använda följande relationsoperatorer:

Operator anges med	*eller med*	*Innebörd*
*	IN	innehåller
/*	EI	innehåller ej
%	BM	börjar med
/%	EB	börjar ej med

Om ett explicit värde är en sträng som innehåller skiljetecken (, : . etc), måste det omges av apostrofer (') eller citationstecken ("). Skiljetecknen behandlas i kapitel 5. Ett odefinierat värde – detsamma som att värde saknas – representeras med en *tom sträng* (", två apostrofer utan mellanslag). Ett explicit värde, som inte innehåller skiljetecken, kan skrivas utan omgivande apostrofer. Om man skriver värdet utan apostrofer, omvandlas alla små bokstäver till stora. Några exempel på explicita villkor:

```
QL> Hämta ...  där  HOTELL.BÄDDAR > 100;
QL> Hämta ...  där  HOTELL.HID = H04;
QL> Hämta ...  där  HOTELL.POSTADR % 'UPP';
QL> Hämta ...  där  ADRESS * 'GATAN';
QL> Hämta ...  där  HOTELL.BÄDDAR > 10
               och  HOTELL.BÄDDAR < 30;
QL> Hämta ...  där  HOTELL.POSTADR = 'GÄVLE'
               eller HOTELL.POSTADR = 'UPPSALA';
```

Här följer några exempel med explicita villkor.

Uppgift 1: Ta fram alla anställda med postadress i Stockholm.

```
QL> Hämta ANSTÄLLD.ANAMN,ANSTÄLLD.POSTADR
    där   ANSTÄLLD.POSTADR % 'STOCK';
```

ANSTNR	*POSTADR*
A130	*STOCKHOLM*
A164	*STOCKHOLM*
A174	*STOCKHOLM*
A176	*STOCKHOLM*
A178	*STOCKHOLM*

5 rader hittade

Uppgift 2: Ta fram alla priskoder där motsvarande pris ligger i intervallet 400–800 kr.

```
QL> Hämta PRIS.PRISKODER där PRIS.PRIS > 400 och PRIS.PRIS < 800;
```

PRISKOD
P40
P50
P60

4 rader hittade

Uppgift 3: Ta fram alla hotellnamn där 'E LUX' ingår i namnet.

```
QL> Hämta HOTELL.HNAMN där HOTELL.HNAMN * 'E LUX';
```

HNAMN
DE LUXE

1 rad hittad

Observera att 'E LUX' måste omslutas med apostrof, eftersom strängen innehåller skiljetecknet mellanslag.

Nu följer ett exempel på implicita villkor. När sådana ställs blir resultatet rader – vanligtvis från olika tabeller – där innehållet i de angivna kolumnerna är lika.

Exemplet utgår från följande två tabeller, HOTELL och RUM:

(HOTELL)

HNAMN	*HID*
LYKTAN	*H01*
HÖRNAN	*H02*
DE LUXE	*H03*
STANDARD	*H04*
REDCAP	*H05*
PLAZA	*H06*

(RUM)

HID	*TYP*
H01	*ENKEL*
H01	*DUBBEL*
H02	*DUBBEL*

H02	*DUBBEL*
H02	*DUBBEL*
H03	*ENKEL*
H03	*DUBBEL*
H03	*SVIT*
H03	*SVIT*
H04	*ENKEL*
H04	*ENKEL*
H05	*DUBBEL*
H05	*SVIT*
H06	*ENKEL*
H06	*DUBBEL*

De två tabellerna kan slås ihop genom kolumnerna med namnen HID, som är av samma typ. Det är kolumntypen som är avgörande i sammanhanget – att kolumnnamnen råkar vara desamma spelar ingen roll.

Villkoret för hopkoppling kan skrivas som ett implicit villkor:

```
RUM.HID = HOTELL.HID
```

Det fullständiga kommandot för att bilda en sammanslagning skrivs enligt följande:

```
QL> Hämta HOTELL.HNAMN, RUM.TYP där RUM.HID = HOTELL. HID;
```

Resultatet blir att följande skrivs ut på skärmen:

HNAMN	*TYP*
LYKTAN	*ENKEL*
LYKTAN	*DUBBEL*
HÖRNAN	*DUBBEL*
HÖRNAN	*DUBBEL*
HÖRNAN	*DUBBEL*
DE LUXE	*ENKEL*
DE LUXE	*DUBBEL*
DE LUXE	*SVIT*
DE LUXE	*SVIT*
STANDARD	*ENKEL*
STANDARD	*ENKEL*
REDCAP	*DUBBEL*
REDCAP	*SVIT*
PLAZA	*ENKEL*
PLAZA	*DUBBEL*

15 rader hittade

8.3 Sammansatta villkor

Villkorssatsen inledd med DÄR kan innehålla flera villkor, sk *delvillkor*. När detta är fallet talas om *sammansatta villkor*. Delvillkoren måste kopplas ihop med hjälp av de *logiska operatorerna* OCH och ELLER.

Delvillkoren behandlas från vänster till höger. ELLER har högre prioritet än OCH. Det senare innebär att om villkorssatsen innehåller både OCH och ELLER, kommer delvillkor hopkopplade med ELLER att behandlas före delvillkor hopkopplade med OCH. Observera att man inte kan ändra behandlingsordningen genom att sätta parenteser.

Om två delvillkor är hopkopplade med ELLER, måste tabrefen till vänster om relationsoperatorerna vara identiska i de båda delvillkoren. Man kan alltså skriva

```
A.KOL1 = 'XXX' ELLER A.KOL2 = 'YYY'
```

men inte

```
A.KOL1 = 'XXX' ELLER B.KOL2 = 'YYY'
```

Om data skall hämtas från flera tabeller vid ett och samma tillfälle, måste sambandet mellan tabellerna anges. Vi måste ange *kopplingsväg*, dvs den ordning i vilken systemet skall läsa tabellkolumner och utvärdera villkoren.

För varje tabref i kolumnlistan måste det finnas åtminstone ett delvillkor i DÄR-satsen, som specificerar sambandet mellan de tabeller som tabref representerar.

Detta är det enda sättet att ange samband mellan tabeller. Om sambandet inte anges, betraktas varje tabell oberoende av alla andra tabeller. Även om två tabeller innehåller två kolumner med lika innehåll och samma kolumnnamn, tex DATUM, måste sambandet specificeras. Om innehållet i kolumnerna är jämförbart, dvs innehåller samma typ av data, måste sambandet specificeras i en HÄMTA-sats.

Det kan finns flera olika kopplingsvägar mellan tabeller. Det är därför viktigt att man anger den rätta kopplingsvägen.

För de tre tabellerna A, B och C finns det sex olika kopplingsvägar:

Alternativ	*Kombination*
1	A B C
2	B C A
3	C A B
4	A C B
5	C B A
6	B A C

Användaren måste ange önskad kopplingsväg med hjälp av delvillkor i en DÄR-sats. Det villkor som motsvarar kopplingsvägen enligt alternativ 1 ovan skrivs på följande sätt:

```
B.KOL = A.KOL och C.KOL = B.KOL
```

Det första delvillkoret – B.KOL = A.KOL – specificerar kopplingsvägen från tabell A till B. Det andra delvillkoret anger kopplingsvägen från B till C. När villkoret utvärderas, börjar MIMER/QL med att läsa värdet på första raden i KOL i tabell A. Därefter försöker MIMER/QL hitta en eller flera rader i tabell B, som innehåller samma värde i KOL som KOL i tabell A. Pga denna ordningsföljd betecknar vi tabell A som FRÅN-tabell (FRÅN-tabref) och B som TILL-tabell (TILL-tabref). Om en eller flera rader uppfyller villkoret, kontrolleras det andra delvillkoret. För varje rad som uppfyller villkoret i tabell B, försöker systemet hitta rader i tabell C, med samma innehåll i KOL som KOL i tabell B. Sökningen upprepas för samtliga rader i tabell A. Här är C FRÅN-tabell och B TILL-tabell.

Även om delvillkoren i exemplet ovan skrivs omvänt erhålls samma resultat. När man anger kopplingsvägar gäller följande grundregler:

1 En kopplingsväg måste innehålla åtminstone ett villkor med följande form:

```
TILL-tabref.kolumn ROP FRÅN-tabref.kolumn
```

När MIMER/QL utvärderar ett sådant uttryck, byts FRÅN-tabref.kolumn ut mot värdet från angivna kolumnen i tabellen FRÅN-tabref. Villkoret är

uppfyllt om det finns något värde i den specificerade kolumnen i tabellen TILL-tabref, som uppfyller det villkor som definieras med ROP.

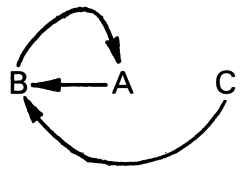
2 Delvillkoren som anger en kopplingsväg kan skrivas i godtycklig ordning i DÄR-satsen.

3 Det kan bara finnas en tabref i en DÄR-sats med flera tabrefer, som är en FRÅN-tabref och inte samtidigt en TILL-tabref. Denna tabell kommer att bli starttabell när villkoren skall utvärderas. Samtliga rader i denna tabell kommer att läsas. Tabell A i exemplet ovan är starttabell.

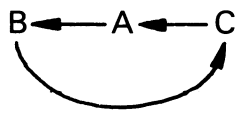
4 Alla tabeller som ingår i en kopplingsväg – utom start- och sluttabellen – måste ingå i minst två implicita delvillkor, i ena fallet som TILL-tabref och i andra som FRÅN-tabref.

5 Kopplingsvägar som är cirkulära kan inte förekomma. En cirkulär kopplingsväg föreligger om det finns ett implicit delvillkor i vilket TILL-tabref går tillbaka till en tabell som redan passerats. I följande exempel föreligger cirkulära kopplingsvägar – ett felaktigt villkor har helt enkelt lagts till i föregående exempel:

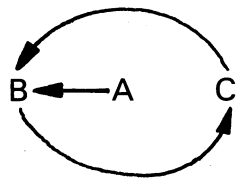
(a) B.KOL = A.KOL **och** C.KOL = B.KOL **och** A.KOL = B.KOL



(b) B.KOL = A.KOL **och** C.KOL = B.KOL **och** A.KOL = C.KOL



(c) B.KOL = A.KOL **och** C.KOL = B.KOL **och** B.KOL = C.KOL



Ovanstående regler kan exemplifieras med beskrivningar av kopplingsvägarna i alternativ 2–6 ovan.

Alternativ 2: B C A

C.KOL = B.KOL **och** A.KOL = C.KOL

alternativt

A.KOL = C.KOL **och** C.KOL = B.KOL

Alternativ 3: C A B

A.KOL = C.KOL **och** B.KOL = A.KOL

alternativt

B.KOL = A.KOL **och** A.KOL = C.KOL

Alternativ 4: A C B

C.KOL = A.KOL **och** B.KOL = C.KOL

alternativt

B.KOL = C.KOL **och** C.KOL = A.KOL

Alternativ 5: C B A

B.KOL = C.KOL **och** A.KOL = B.KOL

alternativt

A.KOL = B.KOL **och** B.KOL = C.KOL

Alternativ 6: B A C

A.KOL = B.KOL **och** C.KOL = A.KOL

alternativt

C.KOL = A.KOL **och** A.KOL = B.KOL

Det är viktigt att man väljer den "bästa" kopplingen mellan tabellerna. Målet är att minimera den mängd data som MIMER/QL måste hantera. Detta uppnås genom att man uteslutande använder *indexerade* kolumner, dvs primärnyckelkolumner eller kolumner med sekundärt index (avsnitt 11.4), vid både explicita och implicita delvillkor. Om man använder icke-indexerade kolumner, vilket är fullt möjligt, sker sökningen sekvensiellt, dvs alla rader i tabellen genomsöks. Om tabellerna är stora tar detta onödigt lång tid.

Förutom implicita delvillkor i en DÄR-sats, som anger sambandet mellan tabellerna, kan man ange explicita delvillkor, som selekterar vissa rader i de aktuella tabellerna. Delvillkoren kan skrivas i godtycklig ordning, utom i de fall den

logiska operatorn är ELLER. I detta fall måste TILL-tabref i de båda villkoren vara lika.

Vi belyser resonemanget om kopplingsvägar med ett par exempel:

1. Vilka hotell inom MIMER HOTELL AB, belägna i Uppsala erbjuder dubbelrum?

Hotellen finns lagrade i tabellen HOTELL (alias H) och rummen i tabellen RUM (alias R). Villkoret för att välja ut alla hotell i Uppsala är

```
HOTELL.POSTADR * 'UPPSALA'
```

och för att välja ut enbart dubbelrum

```
RUM.TYP = 'DUBBEL'
```

Kommandot för att svara på frågan kan då inledas med

```
QL> Hämta  HOTELL.HNAMN
    där    HOTELL.POSTADR * 'UPPSALA'
    och    RUM.TYP = 'DUBBEL';
```

Vi måste nu också ange sambandet mellan de båda tabellerna HOTELL och RUM. Det gör vi med ett implicit villkor där kolumnen HID ingår:

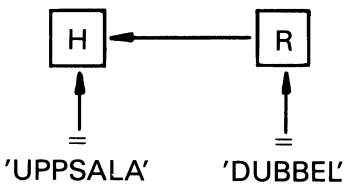
```
HOTELL.HID = RUM.HID
```

Det fullständiga kommandot blir då

```
QL> Hämta  HOTELL.HNAMN
    där    HOTELL.POSTADR * 'UPPSALA'
    och    RUM.TYP = 'DUBBEL'
    och    HOTELL.HID = RUM.HID;
```

När kommandot utförs kommer MIMER/QL att börja i tabellen RUM och där hämta alla rader som uppfyller villkoret TYP = 'DUBBEL'. För varje sådan rad systemet finner försöker det i tabellen HOTELL hitta en rad med samma HID-värde, vilken dessutom uppfyller villkoret PADRESS * 'UPPSALA'. Hotellnamn som motsvarar de rader som uppfyller dessa villkor skrivs ut.

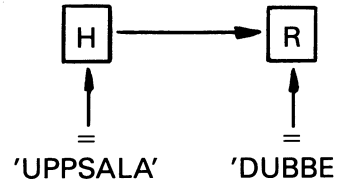
Samma sak skulle inträffa om de tre delvillkoren skrivits i en annan ordning. Vi kan beskriva villkoren enligt följande:



Vi kan ändra det implicita delvillkoret genom att kasta om ordningen mellan tabellerna:

```
RUM.HID = HOTELL.HID
```

MIMER/QL kommer då att starta läsningen i HOTELL-tabellen i stället för RUM-tabellen, enligt nedanstående figur.



2. Vilka hotell inom MIMER HOTELL AB erbjuder dubbelrum i Uppsala eller Stockholm?

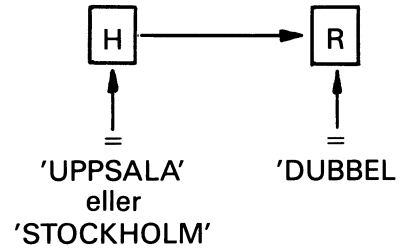
För att ta reda på detta krävs bara att vi lägger till ett ELLER-villkor till vårt föregående kommando:

```
eller HOTELL.POSTADR * STOCKHOLM
```

Eftersom vi också gärna vill se i vilken av städerna ett visst hotell i erhållen förteckning ligger kompletteras även HÄMTA-kommandot. Det fullständiga kommandot för operationen blir

```
QL> Hämta  HOTELL.HNAMN,HOTELL.POSTADR
    där    HOTELL.POSTADR * 'UPPSALA'
    eller  HOTELL.POSTADR * STOCKHOLM
    och    RUM.TYP = 'DUBBEL'
    och    RUM.HID = HOTELL.HID;
```

Vi kan åskådliggöra kommandot med nedanstående figur:



De olika delvillkoren kan placeras i godtycklig ordning, dock inte ELLER-villkoret. Vid ELLER-villkor måste tabref till vänster om aktuell relationsoperator vara densamma i de båda delvillkor mellan vilka ELLER står.

3. Vilka av hotellen inom MIMER HOTELL AB erbjuder dubbelrum i Uppsala eller Stockholm för högst 250 kr dygnet?

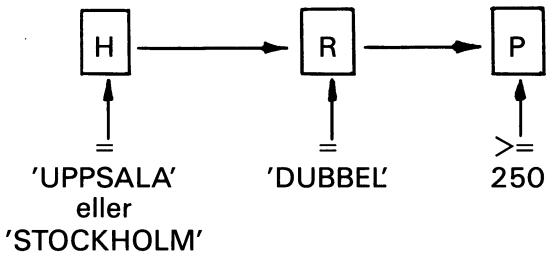
Dygnspriserna lagras i en tredje tabell, PRIS (alias P). Genom kolumnen PRISKOD i tabellen rum kan vi koppla oss till motsvarande kolumn i tabellen PRIS. Två ytterligare delvillkor läggs till vårt senaste kommando:

```
PRIS.PRIS <= 250
och PRIS.PRISKOD = RUM.PRISKOD
```

Det fullständiga kommandot blir således

```
QL> Hämta  HOTELL.HNAMN,HOTELL.POSTADR
    där    HOTELL.POSTADR * 'UPPSALA'
    eller  HOTELL.POSTADR * STOCKHOLM
    och    RUM.TYP = 'DUBBEL'
    och    RUM.HID = HOTELL.HID
    och    PRIS.PRIS <= 250
    och    PRIS.PRISKOD = RUM.PRISKOD;
```

Vi åskådliggör med en figur:



För att få bättre överblick över villkoret kan det dock vara lämpligt att dela upp det i en grupp explicita och en grupp implicita villkor, lämpligen med de förstnämnda först. Kommandot skulle i så fall skrivas enligt följande:

```
QL> Hämta  HOTELL.HNAMN,HOTELL.POSTADR
    där    HOTELL.POSTADR * 'UPPSALA'
    eller  HOTELL.POSTADR * STOCKHOLM
    och    RUM.TYP = 'DUBBEL'
    och    PRIS.PRIS <= 250
    och    RUM.HID = HOTELL.HID
    och    PRIS.PRISKOD = RUM.PRISKOD;
```

I detta kommando anges kopplingsvägen

$H \rightarrow R \rightarrow P$ (1)

Genom ändring av de implicita villkoren kan en annan väg tas, tex följande

$H \leftarrow R \leftarrow P$ (2)

som erhålls om villkoren skrivs

```
och RUM.PRISKOD = PRIS.PRISKOD
och HOTELL.HID = RUM.HID
```

Väg (1) är effektivare. Här utnyttjas enbart primärnyckelkolumner men så är inte fallet i väg (2). PRISKOD är inte primärnyckel till RUM-tabellen.

Vid väg (2) kommer resultatet att presenteras i stigande priskodordning, eftersom man börjar sökningen i PRIS-tabellen, som är sorterad i denna ordning. Väg (1) presenterar resultatet i HID-ordning, dvs den ordning som hotellen är lagrade i HOTELL-tabellen (som är sorterad i stigande ordning efter värdet i HID-kolumnen).

8.4 Arbetstabeller, arbetsdatabank, presentationer och utskrifter

Med kommandona HÄMTA och BESKRIV kan man välja vilka data som skall hämtas och hur resultatet skall presenteras. Med kommandot SKRIV erhålls en redigerad utskrift, som lagras i skrivfilen.

Vid kommandona HÄMTA och BESKRIV kan man med tillägg av en parameter specificera hur resultatet skall presenteras. /T används för att styra utskriften till terminalen, /S till en skrivfil och /A till både terminalen och skrivfilen. Om parametern utelämnas presenteras resultatet på skärmen.

Med kommandot HÄMTA kan också *arbetstabeller* läggas upp. Arbetstabellerna, som placeras i en *arbetsdatabank*, används för tillfällig lagring och bearbetning av data under själva körningen.

En arbetstabell skapas på följande sätt. Det resultat, som erhålls med HÄMTA-kommandot, kan betraktas som en tabell där kolumnerna definieras i en kolumnlista. Om man ramar in denna lista med parenteser och sätter ett tabellnamn framför den, skapas en motsvarande arbetstabell i arbetsdatabanken ARBDB. Namnet på arbetstabellen får bestå av högst 3 tecken.

Innan man kan använda HÄMTA med här beskriven funktion, måste dock arbetsdatabanken ARBDB definieras. Om MIMER/QL används i fleranvändarsystem, delar samtliga användare på ARBDB.

Man kan specificera vilka kolumner som skall utgöra primärnyckel i arbetstabellen. Det gör man på samma sätt som i kommandot DEFINIERA TABELL, dvs genom att skriva ett kolon (:) istället för ett kommatecken (,) i kolumnlistan efter den sista kolumn som skall ingå i primärnyckeln. Om kolon inte anges, kommer samtliga kolumner att utgöra primärnyckel.

Syntaxen för skapande av arbetstabeller är:

HÄMta tabellnamn(kolumnlista) <**DÄR** villkorssats>;

Arbetstabellen i ARBDB kan användas på samma sätt som vanliga MIMER–tabeller. Om man skriver ett nytt HÄMTA-kommando som innehåller samma namn som en redan skapad arbetstabell, kommer den först skapade arbetstabellen att raderas ut och ersättas med den som definieras av det nya HÄMTA-kommandot.

Tabellerna i ARBDB kan användas tills körningen avslutas.

Här följer några exempel och övningsuppgifter som gäller arbetstabeller. Först ett par exempel på kommandon (villkorssatser har inte lagts in):

```
QL> Hämta TAB1(ANSTÄLLD.*:LÖN.LÖNKOD)
    där   ... ;

QL> Hämta TAB2(ANSTÄLLD.ANAMN,
              ANSTÄLLD.BEFATTN:
              LÖN.LÖNKOD)
    där   ... ;
```

Uppgift: Gör en tabell med alla hotellnamn, postadresser och verksamheter som avser alla hotell som har golf i sin verksamhet.

Man börjar med att skapa en arbetstabell (AT1) som innehåller kolumnerna HNAMN, POSTADR och VERKSAMH. Därefter lagrar man de utvalda raderna i AT1. Utskriften åstadkoms genom att man med HÄMTA-kommandot tar fram

alla rader från AT1. Kopplingen mellan tabellerna utgörs av kolumnen HID i resp tabell. Kommandot får följande utformning:

```
QL> Hämta AT1(HOTELL.HNAMN,HOTELL.POSTADR,NÖJEN.VERKSAMH)
    där   NÖJEN.HID = HOTELL.HID
    och   NÖJEN.VERKSMH * 'GOLF';
```

Om vi nu med HÄMTA tar fram samtliga rader i arbetstabellen AT1 får vi följande resultat:

HNAMN	*POSTADR*	*VERKSAMH*
DE LUXE	*ÖSTHAMMAR*	*GOLF*
LYKTAN	*UPPSALA*	*GOLF*

Med kommandot SKRIV kan man lagra en tabell i en sekvensiell skrivfil. Innehållet i filen kan sedan skrivas ut på skrivaren. Namnet på filen sätts med hjälp av tillståndsvariabeln SKRIVFIL, se kapitel 14. Standardnamnet, som finns i systemet vid leverans, presenteras med kommandot UTFÖR SYSPROC(INFO), se även bilaga A.

Skrivfilen öppnas när det första SKRIV- eller HÄMTA/S-kommandot ges under en körning. Filen är öppen ända tills arbetspasset avslutas med SLUTA-kommandot. Den kan dock stängas under passet med något av kommandona SÄTT SKRIVFIL, ÅTERSTÄLL och ÅTERSTÄLL SKRIVFIL. Skrivfilen öppnas då på nytt när nästa SKRIV-kommando ges.

Sidornas storlek, dvs antal rader per sida, sätts med tillståndsvariabeln SL. Om tillståndsvariabeln RA är större än noll (>0), kommer kolumnernas namn att skrivas ut överst på varje sida. Den maximala radlängden (antal tecken) sätts med tillståndsvariabeln SB (1 < SB < 133). Tillståndsvariablerna finns beskrivna i kapitel 14.

I SKRIV-kommandot kan man även ange tabellens rubrik. Syntaxen för detta är:

```
SKRiv tabellnamn 'rubriktext';
```

Rubriken, som får bestå av upp till SB-2 tecken, inklusive mellanslag, måste skrivas inom apostrofer (' ').

Normalt skrivs också kolumnernas namn ut vid SKRIV.

Om vi vill skriva ut resultatet av föregående övningsuppgift på en lista ger således följande kommando:

```
QL> Skriv AT1;
```

2 rader utskrivna

Man kan även överföra en permanent tabell till skrivfilen, som i följande exempel:

```
QL> Skriv HOTELL 'Hotelltabell 15 oktober 1984';
```

6 rader utskrivna

8.5 Speciella funktioner

Det finns inget speciellt kommando om man exempelvis vill ta reda på hur många rader, som uppfyller ett visst villkor. Men om man använder HÄMTA-

kommandot för att skapa en arbetstabell lagras raderna utan att skrivas ut – endast meddelandet "n rader laddade" visas på skärmen. Det är vad som sker i följande exempel:

```
QL> Hämta TAB1(RUM.RUMSNR) där RUM.RUMSNR * 'R1';
                    15 rader laddade
```

8.6 Att avsluta ett arbetspass

Ett arbetspass i MIMER/QL avslutas med kommandot SLUTA, med följande syntax:

```
SLUta;
```

Kommandot medför att samtliga ordinarie tabeller sparas i det tillstånd de befinner sig i när kommandot ges. Arbetstabeller raderas ut.

9 Kopiering till och från sekvensiella filer

9.1 Kopiera rader till en sekvensiell fil

Hela innehållet i en tabell kopieras till en sekvensiell fil med kommandot KOPIERA ... TILL, med syntaxen:

KOPiera tabref **TILl** <'> filnamn <'><(fältspecifikation, ...)>;

Raderna i tabellen tabref kopieras över till den sekvensiella filen filnamn i den ordning de är sorterade i tabellen. (Raderna i en tabell sorteras alltid i stigande ordning på primärnyckeln.) Man kan välja ut vilka kolumner som ska kopieras och med vilket format data ska lagras i den sekvensiella filen, genom att ange en eller flera *fältspecifikationer.*

I fältspecifikationen anger man format, dvs typ och storlek, för fältet i posten i den sekvensiella filen samt vilken kolumn i tabellen som ska lagras i fältet. Om samtliga kolumner ska kopieras i samma ordning och med samma format som i tabellen, behöver man inte ange någon fältspecifikation. Om endast vissa kolumner ska kopieras eller om de ska kopieras i en annan ordning eller med ett format, som inte stämmer med tabelldefinitionen, måste en eller flera fältspecifikationer anges.

Ett fälts startposition i den sekvensiella filen bestäms av summan av de föregående fältens längder (storlekar).

Fältspecifikationen skrivs på samma sätt som kolumnspecifikationen i kommandot DEFINIERA TABELL (kapitel 11). Fältspecifikationen kan även innehålla operatorn nX, där n är ett tal som anger längden på ett fält som ska hoppas över och fyllas med mellanslag.

Posternas maximala storlek i den sekvensiella filen (MSRL) och andra begränsningar presenteras på skärmen med kommandot UTFÖR SYSPROC(INFO), se bilaga A. Observera t ex att det i vissa installationer bara är tillåtet att använda datatyp C.

I följande exempel visas hur en kopiering till sekvensiell fil går till. Tabellen HOTELL som skall kopieras har följande definition och innehåll:

```
HOTELL HID     * C  3
       HNAMN     C 20
```

Innehållet i tabellen framgår av bilaga F. Kopieringen kan nu göras med följande kommando:

```
QL> Kopiera HOTELL till 'HOTFIL'(HID är C3,HNAMN är C20);
```

Kommandot skapar en sekvensiell fil med namnet HOTFIL. Filen innehåller följande poster (tecknet – betyder mellanslag):

```
H01LYKTAN ----------
H02HÖRNAN ----------
H03DE LUXE----------
H04STANDARD ----------
H05REDCAP----------
H06PLAZA----------
```

Kopieringskommandot kan också få denna utformning:

```
QL> Kopiera HOTELL till 'HOTFIL'(2X,HID är C3,1X,HNAMN är C20);
```

Det senare kommandot resulterar i att mellanslag läggs in och att filen får följande innehåll:

```
--H01-LYKTAN ----------
--H02-HÖRNAN----------
--H03-DE LUXE ----------
--H04-STANDARD----------
--H05-REDCAP ----------
--H06-PLAZA ----------
```

Pröva också följande kommando:

```
QL> Kopiera PRIS till PRISFIL;
```

Detta kommando medför att filens innehåll kommer att se ut ungefär på följande sätt när den skrivs ut på skrivaren eller när den ska editeras (tecknet E symboliserar ett tecken som inte kan skrivas ut):

```
-P10EEEEEEEEEEE
-P20EEEEEEEEEEE
-P30EEEEEEEEEEE
-P40EEEEEEEEEEE
-P50EEEEEEEEEEE
-P60EEEEEEEEEEE
-P70EEEEEEEEEEE
-P80EEEEEEEEEEE
-P90EEEEEEEEEEE
```

Varför blir resultatet detta? Om inte någon fältspecifikation anges, kommer den datatyp som finns i kolumndefinitionen att användas. I ovanstående exempel är kolumnen PRISKOD av typ I (heltal). Värdet i denna kolumn kommer att lagras i binärformat och inte i teckenformat, som editeringsprogrammet och skrivaren förväntar sig.

9.2 Kopiera rader från en sekvensiell fil

Med kommandot KOPIERA ... FRÅN kan man kopiera data från en sekvensiell fil till en tabell. En post i filen kommer att utgöra en ny rad i tabellen. Syntaxen är:

```
KOPiera</LAGral/LADda><I/DLOggI/EJDlogg>
tabref FRÅn <'>filnamn<'> <(fältspecifikation, ...)>;
```

Kommandot kopierar poster från den sekvensiella filen filnamn och lagrar dem som rader i tabellen tabref. Om fälten i posten stämmer med tabellradens definition (kolumndefinitionerna) vad avser ordning, typ och storlek (antal tecken), behöver ingen fältspecifikation anges.

Fältspecifikationen anger typ och storlek på fältet i posten samt vilken kolumn i tabellen som fältet ska kopieras till. Ett fälts startposition ges av summan av fältstorlekarna för de fält som föregår det aktuella fältet.

Fältspecifikationen skrivs på samma sätt som kolumnspecifikationen i kommandot DEFINIERA TABELL (kapitel 11), dock med skillnaden att endast komma (,) kan användas som skiljetecken mellan fältspecifikationerna. Data kan alltså lagras med vilket format som helst i den sekvensiella filen och omvandlas till den datatyp, som specificeras i kolumndefinitionen, när data hämtas med KOPIERA-kommandot. Kolumner i tabellen som man inte refererar till i listan med fältspecifikationer tilldelas "odefinierat värde", vilket innebär att värde kommer att saknas i dessa kolumner.

Den eller de kolumner, som utgör primärnyckel, måste alltid specificeras med fältspecifikationer.

Fältspecifikationen kan innehålla operatorn nX, där n anger storleken (antal tecken) på ett fält som ska hoppas över.

Parametrarna LAGRA och LADDA bestämmer hur data ska läggas in i tabellen. När LAGRA används, läggs de nya raderna in direkt i tabellen. Skulle en rad redan finns i tabellen, dvs om primärnyckeln för två rader är densamma, sker ingen överföring (dublett). När LADDA används, töms tabellen på sina befintliga rader, varefter dessa läggs in igen tillsammans med de nya raderna från den sekvensiella filen. När är det då lämpligt att använda LAGRA respektive LADDA?

Om antalet rader som ska kopieras är mycket större än antalet som redan finns i tabellen eller om tabellen är tom, är det bättre att använda LADDA. Observera dock att om det finns flera rader som är lika (dubletter), dvs har identisk primärnyckel, har man ingen kontroll över vilken av dem som lagras. Man kan inte heller logga de rader, som inte lagras.

När man använder LAGRA, kopieras som framgått inte rader som redan finns i tabellen (dubletter). Dessa rader kan dock loggas i en särskild fil (SKRIVF).

LAGRA är standardvärde, dvs gäller om inget annat specificeras.

Användaren måste ha minst S-behörighet för den aktuella databanken för att kunna använda LAGRA och X-behörighet för att kunna använda LADDA. Observera också att transaktionshantering ej förekommer vid kommandotillägg LADDA.

Loggningen av dubletter styrs med parametrarna DLOGG och EJDLOGG där DLOGG är standardvärde om inget annat anges.

Begränsningar i poststorleken (MSRL) och andra systemberoende faktorer presenteras på skärmen med kommandot UTFÖR SYSPROC(INFO), se bilaga A. I en del installationer tillåts endast datatyp C.

Anta att vi ska lagra uppgifter från den sekvensiella filen HOTFIL i tabellen HOTELL (beträffande tabellens definition, se början av detta avsnitt, och beträffande dess innehåll, se avsnitt 8.2):

Den sekvensiella filen HOTFIL innehåller följande poster (tecknet markerar mellanslag):

```
--H01-LYKTAN ----------
--H07-CENTRALHOTELLET--------
--H08-RITZ ----------
--H09-SAVOY----------
--H10-AVENYN----------
--H11-DIPLOMAT----------
```

Följande kommando ges nu för kopiering till tabellen:

```
QL> Kopiera HOTELL från 'HOTFIL'(2X,HID är C3,1X,HNAMN är C20);
```

Kommandot medför att poster i den sekvensiella filen HOTFIL kopieras och lagras som rader i tabellen HOTELL. Den första posten i HOTFIL kopieras dock inte pga att det redan finns en rad i HOTELL som innehåller värdet H01 i primärnyckelkolumnen HID. Posten är en dublett. Eftersom parametern DLOGG är standardvärde, kommer posten att loggas i filen SKRIVFIL. De övriga posterna i HOTFIL kopieras och lagras som rader i HOTELL.

Följande kommandon ger exakt samma resultat som ovanstående kommando:

```
QL> Kopiera/Dlogg HOTELL
          från 'HOTFIL'(2X,HID är C3,1X,HNAMN är C20);

QL> Kopiera/Lagra/Dlogg HOTELL
          från 'HOTFIL' (2X,HID är C3,1X,HNAMN är C20);

QL> Kopiera/Lagra HOTELL
          från 'HOTFIL'(2X,HID är C3,1X,HNAMN är C20);
```

Att dessa kommandon ger samma resultat som det första kommandot beror på att parametrarna LAGRA och DLOGG är standardvärden.

Anta i stället att filen HOTFIL innehåller följande poster (det är bara den första som skiljer sig från föregående exempel):

```
--H01-TORNET ----------
--H07-CENTRALHOTELLET--------
--H08-RITZ ----------
--H09-SAVOY----------
--H10-AVENYN----------
--H11-DIPLOMAT----------
```

Vi ger nu följande kopieringskommando:

```
QL> Kopiera/Ladda HOTELL från
          'HOTFIL'(2X,HID är C3,1X,NAMN är C20);
```

Post 2—5 i HOTFIL kopieras med säkerhet in i HOTELL. Den första posten som är en dublett *kan* komma att ersätta den i tabellen befintliga raden med identiskt lika primärnyckel (HID H01) men det är också möjligt att den senare blir kvar. Eftersom parametern LADDA används, kan man nämligen inte vara säker på vilken av de två raderna som kommer att användas. När LADDA används sorteras raderna i tabellen ihop med posterna från den sekvensiella filen och vid dubletter finns det inget sätt att styra vilken av raderna som slutligen lagras i tabellen. Den som råkar komma först läggs in.

Operatorerna 2X resp 1X anger att två fält, som omfattar 2 tecken resp ett tecken, ska hoppas över i varje post från HOTFIL när filen läses.

När parametern LADDA används, loggas inte dubletterna i filen SKRIVFIL.

Om filen HOTFIL skapades med kommandot KOPIERA ... TILL utan någon fältspecifikation, ser kommandot som vi redan konstaterat ut på följande sätt:

```
QL> Kopiera HOTELL till HOTFIL;
```

Resultatet i HOTFIL blir data i binärt heltalsformat som inte går att läsa med ett editeringsprogram (se exempel ovan i samband med kommandot KOPIERA ... TILL). Anledningen till att man ändå gör en sådan kopiering kan vara att man vill spara en kopia i en extern fil, därefter definiera en ny tabell identisk med tabellen HOTELL i en annan databank, för att sedan kopiera över data från HOTFIL till den nya tabellen med kommandot KOPIERA ... FRÅN:

```
QL> Kopiera TEMPHOT från HOTFIL;
```

Eftersom definitionerna för tabellerna TEMPHOT och HOTELL är identiska, behöver man inte specificera fälten och numeriska data kan lagras i binärt heltalsformat i den tillfälliga filen HOTFIL.

10 Hur man ändrar innehållet i tabeller

Med hjälp av kommandona i MIMER/QL kan man ändra, ta bort och lägga till rader i tabellerna. Innan man kan ändra innehållet i en databank, måste man öppna databanken med kommandot ÖPPNA och ange att man vill skriva, exempelvis

Öppna/Skriv MINBANK;

10.1 Att lägga till nya rader

Nya rader kan läggas till i en tabell med kommandot LAGRA. Detta kommando kan användas på två sätt:

- Systemet frågar efter värdena som skall läggas till. I detta fall behöver man inte känna till namnen på kolumnerna i tabellen. Systemet visar dem.
- Användaren specificerar i vilka kolumner värden skall läggas in.

Syntax för LAGRA-kommandot är således:

LAGra tabref;

eller

LAGra tabref (kolumnnamn=data,...);

Oavsett hur man använder LAGRA-kommandot, skapar MIMER/QL den nya raden och sorterar in den på rätt plats i tabellen efter värdet i primärnyckelkolumnen(-erna). Se vidare kapitel 11, kommandot DEFINIERA TABELL. Raderna i en tabell lagras sorterade enligt värdet på primärnyckeln. Om det redan finns en rad i tabellen med samma värde på primärnyckeln, accepteras inte den nya raden.

För att man skall kunna använda kommandot LAGRA måste man minst ha behörighet S till den aktuella databanken.

10.1.1 Systemet frågar efter värden och kolumner

Systemet frågar efter värdena genom att visa kolumnnamnen i den tabell, som specificeras med tabref. Användaren skall då mata in data följt av vagnretur (RETURN eller Carriage Return,CR). Förutom kolumnnamn visas även format, dvs datatyp och kolumnstorlek – C, I eller F följt av ett tal, allt inom parenteser.

När data har matats in för alla kolumner i tabellen, sorteras raden in i tabellen. Därefter visas den första kolumnen på nytt och inmatningen kan fortsätta.

När alla kolumner visats skall inmatningen avslutas med ett eller två utropstecken (! eller !!). *Två* utropstecken gör att systemet övergår till kommandoläge (kommandomarkören QL> visas) och väntar på ett nytt kommando. *Ett* utropstecken gör att inmatningen av den aktuella raden avbryts och att en ny rad kan matas in (första kolumnen i tabellen visas på nytt).

Om man skriver fel och inte vill spara raden, kan man annullera den genom att i någon kolumn i stället för ett värde skriva ett utropstecken. Därvid avbryts inmatningen och man kan göra om den. Om ett utropstecken matas in redan i den första kolumnen, avbryts inmatningen och systemet övergår till kommandoläge. Ett utropstecken har alltså i detta fall samma effekt som två utropstecken efter avslutad inmatning.

Om man svarar med enbart vagnretur (RETURN eller CR) på en fråga händer något av följande:

- Om datatypen är **C**: en sträng med enbart blanktecken lagras i kolumnen.
- Om datatypen är **I** eller **F**: inmatningen accepteras inte och användaren uppmanas att mata in ett nytt värde.

Om man vill lagra "odefinierat värde", dvs om inget värde finns, måste man skriva två apostrofer (''). Observera dock att detta är omöjligt om kolumnen ingår i primärnyckeln.

Här följer nu exempel på kommandot LAGRA. Anta att vi har en tabell NÖJEN som har följande definition:

```
NÖJEN   HID          *C  3
        VERKSAMH     *C 20
        STARTMÅN      C  2
        SLUTMÅN       C  2
        ÖVRIGT        C 30
```

Om man vill lägga till rader i tabellen skriver man först följande kommando:

```
QL> Lagra NÖJEN;
```

varpå inmatning av rader kan ske (vagnretur har markerats med <VR>):

```
Systemet visar          Du matar in
HID          (C3):      H01<VR>
VERKSAMH     (C20):     DANS<VR>
STARTMÅN     (C2):      01<VR>
SLUTMÅN      (C2):      11<VR>
ÖVRIGT       (C30):     BUGGTÄVLING VARJE MÅNAD<VR>

HID          (C3):      H04<VR>
VERKSAMH     (C20):     TENNIS<VR>
STARTMÅN     (C2):      05<VR>
SLUTMÅN      (C2):      08<VR>
ÖVRIGT       (C30):     GRUSBANA<VR>

HID          (C3):      !!<VR>

QL>
```

Tabellen NÖJEN har nu utökats med två rader. Inmatningen avslutas genom att två utropstecken matas in när systemet frågar efter nästa anställningsnummer. (I detta läge hade det dock räckt med ett utropstecken för att avsluta inmatningen eftersom utropstecknet ges in i första kolumnen.)

Om data som skall lagras i en kolumn med datatyp C innehåller färre tecken än vad som maximalt ryms i kolumnen, sker utfyllnad med mellanslag.

Om det inmatade värdet skall lagras i en kolumn med datatyp I eller F, är mellanslagen inte signifikanta. Ett helt "blankt" värde, dvs enbart mellanslag, accepteras dock inte.

10.1.2 Kolumnerna specificeras av användaren

I stället för att låta systemet fråga efter data på det sätt som beskrivits i föregående avsnitt, kan man skriva kolumn och motsvarande data inne i själva LAGRA-kommandot. Ett LAGRA-kommando måste ges för varje rad med data. Man måste då veta vilka kolumner som finns i tabellen. Kommandot BESKRIV TABELL kan användas för att ta reda på vilka kolumner som finns i tabellen.

Om den första av de rader som lades in i tabellen NÖJEN i föregående exempel i stället hade matats in på detta sätt skulle kommandot ha sett ut så här:

```
QL> Lagra NÖJEN (HID = 'H03',
+>              VERKSAMHET = 'FISKDAMM',
+>              STARTMÅN = '05',
+>              SLUTMÅN = '09',
+>              ÖVRIGT = 'UTHYRNING AV BÅT');
```

Observera att data som innehåller skiljetecken (vilka dessa är framgår av kapitel 5) måste omges med strängskiljetecken, dvs apostrofer (' ') eller citationstecken (" "). Det gäller enbart alfanumeriska data.

Det spelar ingen roll i vilken ordning kolumnerna kommer. Inledande och avslutande mellanslag som lagras i F- eller I-kolumner är ej siginifikanta.

Kolumner, som utelämnas i LAGRA-kommandot, tilldelas "odefinierat värde". När en sådan kolumn visas på skärmen, tex med HÄMTA-kommandot, visas värdet som en rad punkter (...). Observera att primärnyckelkolumnerna måste tilldelas värden.

De två sätten att använda LAGRA-kommandot kan användas växelvis enligt följande exempel:

```
QL> Öppna/S HOTELLDB (PRIS);
QL> Lagra PRIS;
PRISKOD      (C4):  P15<VR>
PRIS         (I2):  345<VR>

PRISKOD      (C4):  !!<VR>

QL> Lagra PRIS (PRISKOD = 'P25');
QL> Alias PRIS (P1);
QL> Hämta P1.* där P1.PRISKOD = 'P15'
                  eller P1.PRISKOD = 'P25';
```

Resultatet blir:

PRISKOD	*PRIS*
P15	*345*
P25	....

2 rader hittade

10.2 Att ändra rader

Med kommandot ÄNDRA kan man ändra de rader i tabellen, som uppfyller ett angivet villkor. Syntaxen är:

ÄNDra tabref (kolumnnamn=data,...)
<**DÄR** villkorssats>;

För alla rader, som uppfyller villkoret, byts innehållet i de angivna kolumnerna mot 'data' i kommandot. Man kan inte ändra värdet i någon av de kolumner som ingår i primärnyckeln, dvs det är inte tillåtet att ange dessa kolumner i ÄNDRA-kommandot. I villkorssatsen däremot kan man givetvis använda primärnyckelkolumnerna.

Villkorssatsen skrivs på samma sätt som i HÄMTA-kommandot (kapitel 8). Referensen till den tabell som skall uppdateras (tabref), måste återfinnas i åtminstone ett delvillkor till vänster om relationsoperatorn i villkorssatsen. Om villkorssatsen utelämnas, ändras samtliga rader i tabellen. För att man skall kunna använda ÄNDRA-kommandot måste man minst ha S-behörighet. Exempel:

```
QL> Öppna/S HOTELLDB;
QL> Alias HOTELL (H);
QL> Ändra H (STANDARD = '***')
    där    H.HID = 'H06';
```

1 rad ändrad

10.3 Att ta bort rader

Med hjälp av kommandot RADERA kan man radera ut rader i tabellen. Syntaxen är:

RADera tabref <**DÄR** villkorssats>;

Kommandot raderar samtliga rader i tabellen "tabellnamn" som uppfyller angivet villkor. Villkorssatsen skrivs på samma sätt som i HÄMTA-kommandot, se kapitel 8. Referensen till tabellen (tabellnamn) måste återfinnas i åtminstone ett delvillkor till vänster om relationsoperatorn. Om villkorssatsen utelämnas raderas samtliga rader i tabellen, något som dock endast kan göras av användare med X-behörighet. I detta fall förekommer ej transaktionshantering. För att man skall kunna använda villkorssatsen, måste man minst ha S-behörighet. Exempel:

```
QL> Radera NÖJEN där NÖJEN.HID = H01;
```

1 rad raderad

11 Att definiera, ändra och ta bort tabeller och index

Det här kapitlet innehåller en beskrivning av hur man definierar, ändrar och tar bort tabeller samt definierar och tar bort sk *sekundärindex*.

11.1 Att definiera tabeller

Så snart en databank är definierad, kan man definiera en tabell i databanken, dvs ange namn och format för varje kolumn i tabellen. Det görs med kommandot DEFINIERA TABELL, som kräver att användaren har X-behörighet till den aktuella databanken. Hur man definierar databanker behandlas i kapitel 12.

Syntax för kommandot DEFINIERA TABELL är:

DEFiniera TABell tabellnamn (kolumnnamn **ÄR** format
<:kolumnnamn **ÄR** format>
<,kolumnnamn **ÄR** format>)
I databanksnamn;

Angående regler för tabell- och kolumnnamn, se kapitel 5. Varje tabellnamn måste vara unikt inom databanken, dvs ett tabellnamn får bara användas en gång i samma databank. Tabeller med samma namn i olika databanker kan inte användas samtidigt.

På motsvarande sätt måste varje kolumnnamn vara unikt inom tabellen. En tabell får ha maximalt 99 kolumner.

Uttrycket "kolumnnamn är format" kallas *kolumnspecifikation*. Alla kolumnspecifikationer tillsammans, dvs allt som står inom parentes i syntaxbeskrivningen, brukar kallas *kolumnlista*.

Om flera kolumner ingår i tabellen skall namnen på dessa skiljas åt med kommatecken (,).

Kolumner som ingår i primärnyckeln måste placeras först i kolumnlistan. Efter specifikationen av den sista kolumnen i primärnyckeln skriver man ett kolon (:) i stället för komma (,) för att skilja primärnyckelkolumnerna från övriga kolumner. Sätts inget kolon i kolumnlistan betraktas samtliga kolumner som ingående i primärnyckeln.

Raderna kommer att sorteras in i tabellen enligt värdet på *hela* primärnyckeln. Den inbördes ordningen mellan kolumnerna i primärnyckeln är alltså mycket viktig. Den första kolumnen i primärnyckeln (=kolumnlistans första kolumn) är *mest signifikant* och kolumnen före kolontecknet är *minst signifikant*. Detta innebär att en rad i första hand sorteras in efter värdet i den första kolumnen i primärnyckeln, i andra hand efter värdet i den andra kolumnen osv.

Ordet "är" måste finnas mellan kolumnnamn och format.

"Format" i syntaxbeskrivningen ovan anger kolumnens datatyp och storlek (beträffande format, se vidare avsnitt 2.2). Formatet skrivs med 2–4 tecken, t ex C10. Första tecknet, som måste vara en bokstav, anger datatypen, dvs någon av följande:

C	Alfanumerisk teckensträng
I	Heltal
F	Flyttal

De därpå följande tecknen, 1–3 siffror, anger kolumnens storlek, dvs det utrymme den tar på raden. I C-format är format lika med antalet tecken som kan lagras i kolumnen. I I- och F-format däremot lagras värdena i binär form och i detta fall betyder kolumnens storlek i stället ett antal *byte*. Varje byte består av ett antal *bitar*. Antalet bitar per byte beror på vilket datorsystem som används. Det finns standardvärden inlagda i systemet som anger hur stora eller små tal som kan lagras i I- respektive F-format. Denna information presenteras på skärmen med kommandot UTFÖR SYSPROC(INFO), se bilaga A.

En kolumn har alltid samma format genom alla rader i tabellen. Alla rader i en tabell är lika långa. Radlängden (=summan av samtliga kolumners storlek) har en övre gräns som är beroende av datorsystem – variabeln MMRL, se bilaga A.

Slutligen måste i definitionen anges namn på den databank där tabellen ingår. Databanksnamnet måste föregås av ordet "i".

Här följer ett exempel på tabelldefiniering:

```
QL> Definiera tabell NYTAB (KOL1 är I4,
+>                          KOL2 är I2:
+>                          KOL3 är C30)
+>  I ANSTDB;
```

Med detta kommando skapas en ny tabell med namnet NYTAB i den tidigare definierade databanken ANSTDB. Tabellen innehåller tre kolumner: KOL1, KOL2 och KOL3. KOL1 och KOL2 utgör primärnyckel, varför skiljetecknet efter specifikationen av KOL2 är ett kolon (:).

Med kommandot BESKRIV TABELL får man en presentation av tabelldefinitionerna:

```
QL> Beskriv tabell NYTAB;
```

När detta kommando ges kommer följande information att presenteras på skärmen:

```
NYTAB 1 KOL1   * I  4
      2 KOL2   * I  2
      3 KOL3     C 30
```

Tecknet * markerar att kolumnerna ingår i primärnyckeln.

11.2 Tabellradering

Tabellradering kan antingen avse *hela tabellen* med både innehåll och tabelldefinition eller *enbart innehållet*.

11.2.1 Att radera hela tabeller

Med kommandot ELIMINERA TABELL kan man ta bort en tabell. Syntaxen är:

ELIminera TABell tabellnamn;

Kommandot ELIMINERA TABELL raderar inte bara ut innehållet i tabellen, utan tar även bort tabelldefinitionen (tabellnamn och kolumnspecifikationer), dvs tabellen försvinner helt. Om vi tex ger kommandot

QL> **Eliminera tabell** NYTAB;

kommer hela tabellen NYTAB att försvinna. Om man arbetar i bekräftelseläge (se kommandot SÄTT BEKRÄFTA, kapitel 14), vilket är standard, måste användaren dock bekräfta att tabellen verkligen ska tas bort helt. Det sker genom att besvara följande fråga som visas på skärmen när kommandot ELIMINERA TABELL getts in:

Tabellen förstörs? (J/N): J<VR>

Om man svarar J (Ja) raderas såväl tabellinnehåll som tabelldefinition ut. Om man däremot svarar N (Nej) avbryts operationen och ingen radering sker.

För att man skall kunna radera ut en tabell måste man minst ha X-behörighet till den aktuella databanken.

Exempel på radering av hel tabell finns i avsnitt 11.3.2.

11.2.2 Att radera innehållet i en tabell

Om man bara vill radera innehållet i tabellen men behålla kolumnspecifikationerna, skall man använda kommandot RADERA utan villkorssats. Syntaxen är:

RADera tabref;

Om vi tex ger kommandot

QL> **Radera** PERSONAL;

kommer allt innehåll, dvs samtliga rader i tabellen PERSONAL att tas bort. Däremot finns tabelldefinitionen kvar. Om man arbetar i bekräftelseläge (se kommandot SÄTT BEKRÄFTA, kapitel 14), måste man även i detta fall bekräfta raderingen innan den utförs genom att besvara följande fråga som visas på skärmen:

Alla rader tas bort? (J/N): J<VR>

Om man svarar J (Ja) raderas samtliga rader. Om man däremot svarar N (Nej) avbryts operationen och ingen radering sker.

För att man skall kunna radera alla rader måste man ha X-behörighet till den aktuella databanken.

Om vi nu vill se resultatet av de båda borttagningar som vi nyss gjort kan vi använda kommandot BESKRIV TABELL, först på tabellen NYTAB. Resultatet blir detta:

QL> **Beskriv/full tabell** NYTAB;

****Fel (34): Tabell med angivet namn har ej öppnats*

Hela tabellen med innehåll och definition har försvunnit. Motsvarande operation på tabellen PERSONAL ger ett annat resultat:

```
QL> Beskriv/full tabell PERSONAL;
PERSONAL  1   HID        *C   3
          2   ANSTNR      C   5
Rad längd: 8      Antal rader: 0
```

I detta fall finns tabelldefinitionen kvar men innehållet har försvunnit.

11.3 Att ändra tabelldefinitioner

Med kommandot OMDEFINIERA kan man ändra tabelldefinitioner, t ex lägga till eller ta bort kolumner. Kommandot kan bara användas när tabellen är tom, dvs när den inte innehåller några data. Därför måste man tömma tabellen, innan OMDEFINIERA-kommandot kan användas.

Omdefiniering kan delas upp i följande steg:

Först kopieras tabellen till en sekvensiell fil med kommandot KOPIERA ... TILL. Därefter raderas innehållet i tabellen med kommandot RADERA utan villkorssats. Tabellen är nu tom och dess utseende kan ändras med kommandot OMDEFINIERA.

När man sedan skall kopiera tillbaka data från den sekvensiella filen till tabellen, använder man kommandot KOPIERA ... FRÅN.

För att man skall kunna använda OMDEFINIERA-kommandot måste man ha X-behörighet till den aktuella databanken.

Om man vill lägga till nya kolumner *i slutet* av en tabell, använder man LÄGG-TILL-formen av kommandot OMDEFINIERA, med syntaxen:

OMDefiniera tabellnamn **LÄGg-till** (kolumnnamn **ÄR** format,...);

Om man i stället vill lägga in en eller flera kolumner *mitt i* en tabell, för att exempelvis öka antalet kolumner i primärnyckeln, måste man göra på följande sätt.

Först tar man bort samtliga kolumner till höger om den plats, där den nya kolumnen skall infogas. Det görs med kommandot OMDEFINIERA RADERA (se nedan). Därefter definierar man om kolumnerna med hjälp av kommandot OMDEFINIERA LÄGG-TILL, se ovan. Kolumnspecifikationen skrivs precis som vid kommandot DEFINIERA TABELL. Kolon (:) sätts efter specifikationen av den sista kolumn som ingår i primärnyckeln. I övriga fall åtskiljs specifikationerna med kommatecken (,).

Borttagning av kolumner utförs med kommandot OMDEFINIERA RADERA. Man kan även ta bort kolumner, som ingår i primärnyckeln. Det måste dock alltid finnas minst en primärnyckelkolumn i en tabell. Syntaxen är:

OMDefiniera tabellnamn **RADera** (kolumnnamn,...);

Endast kolumner vilkas namn anges kommer att tas bort.

Här följer exempel på ändring av tabelldefinition. Vi börjar med att definiera en tabell, TEST, som från början har två kolumner, KOL1 och KOL2:

```
QL> Definiera tabell TEST (KOL1 är C6:
+>                         KOL2 är C10)
+>  i ANSTDB;
```

Vi tar bort KOL2 och lägger till en ny KOL2 som skall ingå i primärnyckeln,

lägger in den gamla KOL2 igen, nu som KOL3, och lägger slutligen till en ny KOL4:

```
QL> Omdefiniera TEST radera (KOL2);
QL> Omdefiniera TEST lägg-till (KOL2 är F4:
                                KOL3 är C10,
                                KOL4 är I2);
```

Om vi nu tittar på definitionen av tabellen finner vi följande:

```
QL> Beskriv tabell TEST;
TEST      1   KOL1      * C    6
          2   KOL2      * F    4
          3   KOL3        C   10
          4   KOL4        I    2
```

För att lägga in den nya primärnyckelkolumnen KOL2 måste vi alltså först med OMDEFINIERA RADERA ta bort alla kolumner som skall komma efter den nya kolumnen, i exemplet således KOL3. När kolumnerna sedan läggs in med OMDEFINIERA LÄGG-TILL markeras att KOL2 skall ingå i primärnyckeln med kolon efter specifikationen. KOL3 och KOL4 är vanliga kolumner som skiljs åt med komma.

11.4 Sekundära index

För att underlätta sökningen av rader som specificeras med en villkorssats kan man förse kolumner i tabeller med sk *sekundära index*. Metoden innebär att sortering och sökning kan ske enligt en annan och effektivare ordning än den som följer av tabellens definierade primärnyckel.

Sekundära index skapas med kommandot DEFINIERA INDEX och tas bort med med kommandot ELIMINERA INDEX. För att skapa eller ta bort sekundära index måste man ha minst X-behörighet till den aktuella databanken.

11.4.1 Att skapa sekundära index

För att skapa ett sekundärt index använder man kommandot DEFINIERA INDEX, vars syntax är:

```
DEFiniera INDex tabellnamn (kolumnnamn,...);
```

Kommandot skapar ett sekundärt index för kolumnen kolumnnamn i tabellen tabellnamn. Den senare kallas *bastabell*. Man kan indexera ett godtyckligt antal kolumner i en tabell.

Det som sker vid sekundär indexering är att det skapas en ny, intern MIMER-tabell, en *indextabell*. I denna tabell kommer den indexerade kolumnen först och följs sedan av de kolumner som ingår i primärnyckeln i bastabellen. I indextabellen är alla kolumner primärnyckelkolumner.

Ett sekundärt index behöver bara skapas en gång för en kolumn. Indexet underhålls och uppdateras sedan automatiskt när bastabellen uppdateras. Man kan givetvis när som helst ta bort ett sekundärt index, med kommandot ELIMINERA INDEX (se nedan).

Den första kolumnen i primärnyckeln har redan från början ett index som ligger i bastabellens fysiska struktur.

När ett sekundärt index väl är definierat, kommer systemet automatiskt att utnyttja detta index. Man behöver således inte vidta några ytterligare åtgärder i denna riktning.

Observera dock att ett sekundärt index kommer att användas av systemet bara om det läggs ett villkor på den indexerade kolumnen och om detta villkor avser ett s k *slutet intervall*. De följande exemplen på villkor avser just sådana slutna intervall:

1. tabell.indexkol = värde
2. tabell.indexkol % värde
3. tabell.indexkol >= värde och tabell.indexkol <= värde
4. tabell.indexkol >= värde och tabell.indexkol > värde
5. tabell.indexkol > värde och tabell.indexkol <= värde
6. tabell.indexkol > värde och tabell.indexkol < värde

Vi kallar villkor av typ 1 ovan för A-villkor, villkor av typ 2 B-villkor och villkor av typ 3–6 C-villkor. C-villkoren har lägsta prioritet och A-villkoren har högre prioritet än B-villkoren.

Om DÄR-satsen innehåller villkor ställda på mer än en indexerad kolumn och slutna intervall föreligger, gäller följande regler:

- endast ett index kommer att utnyttjas
- villkoret med högsta prioritet avgör valet av index
- om det finns flera kolumner villkorade med samma prioritet väljs index för den av kolumnerna som har minsta kolumnlängden
- om det finns flera villkorade kolumner som inte bara har samma prioritet utan också samma kolumnlängd väljs index för den av kolumnerna som följer närmast på primärnyckeln i kolumndefinitionen

I en DÄR-sats kan det givetvis förekomma villkor både på primärnyckelkolumner och sekundärt indexerade kolumner. I dessa fall kommer en sekundärt indexerad kolumns index att gälla bara då villkoret på kolumnen är av högre prioritet än villkoren på primärnyckelkolumnerna, dvs

- villkoret på den sekundärt indexerade kolumnen är ett A-villkor, medan villkoren på primärnyckelkolumnerna är av annan typ
- villkoret på den sekundärt indexerade kolumnen är ett B-villkor, medan villkoren på primärnyckelkolumnerna är C-villkor eller avser ej slutet intervall
- villkoret på den sekundärt indexerade kolumnen är ett C-villkor, medan villkoren på primärnyckelkolumnerna avser ej slutet intervall

I alla övriga fall kommer primärnyckelindex att gälla.

Om vi ger kommandot

```
QL> Hämta ANSTÄLLD.* där ANSTÄLLD.POSTNR = '753 35';
```

kommer MIMER/DB automatiskt att använda det sekundära indexet för kolumnen POSTNR — om det finns något — för att utvärdera villkoret. Dessutom kommer raderna att sorteras i stigande ordning med avseende på innehållet i kolumnen POSTNR. Om det inte finns något sekundärt index definierat kommer sorteringen att bestämmas på vanligt sätt av primärnyckeln för tabellen.

Nedanstående kommando uppdaterar inte bara bastabellen utan även de indextabeller som eventuellt finns definierade:

```
QL> Lagra ANSTÄLLD(ANSTNR = A105,ANAMN = 'Anders Svensson', ...);
```

Det finns ett par nackdelar med sekundära index som bör beaktas, nämligen att uppdateringar kommer att ta längre tid och att indexen tar mer plats i datorns yttre minne. Om bastabellen innehåller många rader tar det också lång tid för systemet att skapa ett sekundärt index. Man bör alltså tänka sig för innan sekundära index läggs upp.

Här följer ett exempel som belyser effekten av sekundära index.

Med kommandot BESKRIV TABELL kan vi först se vilka kolumner i en tabell som är indexerade. Vi tittar på tabellen ANSTÄLLD och får fram följande:

```
ANSTÄLLD  ANSTNR      * I  4
          ANAMN         C 20
          ADRESS        C 20
          POSTNR        C  6
          POSTADR       C 15
          ANKN          I  2
          ANSTDAT       C  6
          BEFATTN       C 20
          LÖNKOD        C  3
```

Vi vet att tabellen innehåller följande rader:

ANSTNR	ANAMN
32	Sylvén
64	Svärd
128	Johansson
256	Hansson
512	Vernersson
1024	Samuelsson

Här visas endast de kolumner som är betydelsefulla för exemplet. Vi skall nu ta fram anställningsnummer och namn på alla personer i tabellen vars namn börjar med 'S'. Vi skriver in motsvarande kommando och får fram ett resultat:

```
QL> Hämta ANSTÄLLD.ANSTNR,ANSTÄLLD.ANAMN
    där ANSTÄLLD.ANAMN bm S;
```

ANSTNR	ANAMN
32	Sylvén
64	Svärd
1024	Samuelsson

3 rader hittade

Raderna är sorterade efter kolumnen ANSTNR som är primärnyckel till tabellen ANSTÄLLD.

Ett sekundärt index för kolumnen ANAMN skapas med kommandot:

```
QL> Definiera index ANSTÄLLD(ANAMN);
```

Om man därefter ger samma HÄMTA-kommando som ovan erhålls följande resultat:

ANSTNR	*ANAMN*
1024	*Samuelsson*
64	*Svärd*
32	*Sylvén*

3 rader hittade

Observera att raderna nu är sorterade efter kolumnen ANAMN, eftersom det finns ett sekundärt index för denna kolumn. Förutom att sorteringsordningen nu är ändrad, sker också sökningen av namn som börjar på 'S' mer effektivt. Om vi hade haft en mycket stor datamängd hade vi kunnat observera stor skillnad i söktid.

Med kommandot BESKRIV TABELL kan vi också se att kolumnen ANAMN har ett sekundärt index — det anges med ordet "indexerad" efter formatdefinitionen:

```
QL> Beskriv tabell ANSTÄLLD;

ANSTÄLLD  ANSTNR      * I  4
          ANAMN         C 20 *Indexerad*
          ADRESS        C 20
          POSTNR        C  6
          POSTADR       C 15
          ANKN          I  2
          ANSTDAT       C  6
          BEFATTN       C 20
          LÖNKOD        C  3
```

11.4.2 Att ta bort sekundära index

Borttagning av ett sekundärt index sker med kommandot ELIMINERA INDEX, vars syntax är:

ELIminera INDex tabellnamn (kolumnnamn,...);

Tabellnamn refererar till *bastabellen* (se föregående avsnitt) och kolumnnamn till den kolumn, vars sekundära index skall tas bort.

Om vi exempelvis vill ta bort sekundärt index för kolumnen NAMN i tabellen ANSTÄLLD ger vi följande kommando:

```
QL> Eliminera index ANSTÄLLD(NAMN);
```

Använd kommandot BESKRIV TABELL för att förvissa dig om att sekundärindex för NAMN inte längre finns kvar.

12 Hur man definierar en databas

Att definiera en databas innebär att man definierar databanker, användare och användarbehörighet. Kommandot DBA används för detta ändamål. För att man skall kunna utföra dessa operationer måste man ha ställning X (DBA).
Med kommandot DBA kan man också se vilka användare, databanker och behörigheter som är definierade. Alla användare oavsett om de har ställning S eller X kan få denna information. Innan databanker definieras måste en SYSDB skapas med hjälpprogrammet SYSDBGEN (se bilaga F, början).

12.1 Att definiera och ta bort databanker

Databanken skapas med ett speciellt hjälpprogram eller med kommandot DBA. DBA-kommandot ger en meny på skärmen. Menyn innehåller funktioner med vilka man kan definiera, ändra, ta bort och beskriva databanker, användare och användarbehörighet.

När man väljer att definiera eller ändra en databank, frågar systemet efter följande:

- *Databanksnamn*
  Namnet får bestå av högst åtta tecken (bokstäverna A-Ö och siffrorna 0–9). Namnet måste alltid börja med en bokstav.
- *Fysiskt filnamn*
  Operativsystemets namn på den fysiska fil, som innehåller databanken.
- *Databankens storlek*
  Ett numeriskt värde >=0. Ett värde >0 anger databankens storlek i antal sidor. Sidornas storlek framgår av den maskinberoende informationen (bilaga A), variabeln DBPSIZE. Om värdet = 0, indikerar man att något skall ändras i en befintlig databank, tex generell behörighet eller filnamn.
- *Generell behörighet*
  Anger vilka operationer användare utan definierad *personlig behörighet* (avsnitt 12.3) kan utföra på data i databanken. Följande generella behörigheter finns:
  X Alla operationer är tillåtna.
  S Skrivoperationer med exempelvis LAGRA, ÄNDRA och RADERA med villkorssatser (DÄR-satser) är tillåtna.
  R Enbart läsoperationer med exempelvis HÄMTA, SKRIV och KOPIERA TILL är tillåtna.

P Inga operationer kan utföras på data i databanken. Databankerna TRANSDB och LOGDB kräver denna generella behörighet.
B Databanken används endast för backup.

När man skall ta bort en databank, väljer man först borttagningsfunktionen på DBA-menyn och anger därefter namnet på den databank, som ska tas bort. Detta kräver dock att man har ställning X (DBA).

Här följer exempel på definiering av databanker. Med DBA-kommandot tar vi fram DBA-menyn och gör ett val i denna, varefter vi besvarar ett antal ledordsfrågor (nödvändiga vagnreturer efter menyn har markerats med <VR>):

```
QL> Dba;

****************** Databasdefinition-funktioner ********************
*                                                                  *
* 1. Skapa användare   2. Skapa databanker    3. Skapa access     *
* 4. Ta bort användare 5. Ta bort databanker  6. Ta bort access   *
* 7. Visa användare    8. Visa databanker     9. Visa access      *
* 0. Avsluta                                                       *
********************************************************************

Ange val: 2<VR>

Skapa/omskapa databanker

Databank (max 8 tecken)      :NYBANK<VR>
Filnamn                      :NYBANK.TYP<VR>
Storlek (0=befintlig fil)    :25<VR>
Allmän access (X/S/R/S/B)    :X<VR>

<<<< Skapad >>>>

Databank (max 8 tecken )     :PRIVAT<VR>
Filnamn                      :PRIVAT.TYP<VR>
Storlek (0=befintlig fil)    :0<VR>
Allmän access (X/S/R/S/B)    :P<VR>

<<<< Omskapad >>>>

Databank (max 8 tecken)      :<VR>
```

Vi tar också ett exempel på borttagning av databank. Menyn är förstås densamma som ovan. Vi väljer i den och får en fråga som besvaras:

```
Ange val: 5<VR>

Databank (max 8 tecken)      :GAMLBANK<VR>

<<<< Borttagen >>>>

Databank (max 8 tecken)      :<VR>
```

När detta är gjort går vi ur DBA-menyn och får upp kommandomarkören:

```
Ange val: 0<VR>

QL>
```

I ovanstående exempel skapas först en ny databank – NYBANK (storleken >0). NYBANK skall lagras i filen NYBANK.TYP. Om denna fil redan finns och innehåller information, raderas informationen. NYBANK åsätts den generella behörigheten S.

Därefter omdefinieras en befintlig databank, PRIVAT (storleken = 0). PRIVAT tilldelas den generella behörigheten P, dvs inga operationer på data i databanken kan utföras.

För att avbryta rutinen trycker man på vagnretur, när systemet frågar efter databankens namn. DBA-menyn visas igen och när man väljer 0 (avsluta) går systemet över till kommandoläge (QL>).

För att ta bort en databank, väljer man alternativ 5 (ta bort databanker) på DBA-menyn. Därefter skriver man namnet på den databank, som ska tas bort (GAMLBANK). Databanken försvinner inte fysiskt från skivan. Endast databankens definition i biblioteket tas bort. En borttagen databank kan definieras på nytt, se PRIVAT i ovanstående exempel. Databankens storlek måste i så fall anges som 0 (noll).

12.2 Att definiera och ta bort användare

Endast databasadministratören (DBA) kan definiera och ta bort användare. Systemet frågar då efter följande:

- *Användarnamn*
  Namnet får bestå av upp till åtta alfanumeriska tecken. Endast versaler (stora bokstäver) får användas. Första tecknet måste vara en bokstav. Om ett redan befintligt namn anges, omdefinieras användaren.
- *Lösenord*
  Lösenordet får bestå av högst åtta tecken. Lösenordet lagras i chiffrerad form.
- *Ställning*
  Två möjligheter finns:
  X Exklusiv, vilket är detsamma som DBA. Användare med ställning X kan utföra alla operationer, dvs även definiera databanker, användare etc.
  S Standard, tillkommer alla användare som inte har DBA-ställning.

När en ny användare är definierad, kan denna själv ändra sitt lösenord med kommandot DBA.

DBA-kommandot kan även användas för att ta bort en användare. Först väljer man "ta bort användare" på DBA-menyn och därefter anger man namnet på den användare, som skall tas bort.

Här följer exempel på definiering av användare. Med DBA-kommandot tar vi fram DBA-menyn och gör ett val i denna, varefter vi besvarar ett antal ledordsfrågor (nödvändiga vagnreturer efter menyn har markerats med VR):

```
QL> Dba;

**********Databasdefinition-funktioner**********
*                                               *
* 1. Skapa användare   2. Skapa databanker   3. Skapa access *
* 4. Ta bort användare 5. Ta bort databanker  6. Ta bort access *
* 7. Visa användare    8. Visa databanker     9. Visa access *
* 0. Avsluta                                    *
*                                               *
*************************************************

Ange val: 1<VR>

Skapa/omskapa användare

Användare (max 8 tecken )        :BLOM<VR>

Lösenord (max 8 tecken)          :DAHL<VR>
Ställning (X/S)                  :S<VR>

<<<< Skapad >>>>
```

Användare (max 8 tecken) :TIB<VR>
Lösenord (max 8 tecken) :53330<VR>
Ställning (X/S) :X<VR>

<<<< *Skapad* >>>>

Användare (max 8 tecken) :<VR>

Vi tar också ett exempel på borttagning av användare (menyn är förstås densamma som ovan):

Ange val: 4<VR>

Ta bort användare

Användare (max 8 tecken) :BLOM<VR>

<<<< *Borttagen* >>>>

I det här exemplet läggs en ny användare in i systemet: BLOM. Hans lösenord är DAHL och hans ställning S. Användaren TIB läggs också in. Han tilldelas DBA-ställning (X). Flera nya användare kan läggas in vid ett och samma tillfälle. Inmatningen avbryts genom att man trycker på vagnretur vid "användare".

För att ta bort en användare väljer man alternativ "4" på DBA-menyn. I ovanstående exempel tas användaren BLOM bort.

12.3 Att definiera och ta bort personlig användarbehörighet

När man definierar en ny databank eller ändrar en befintlig, bestämmer man samtidigt om *generell behörighet* till databanken (se avsitt 12.1). Den generella behörigheten kan upphävas genom att användaren tilldelas en *personlig användarbehörighet* med kommandot DBA. Systemet frågar efter nedanstående information när man väljer alternativet "skapa access" på DBA-menyn.

- *Användarens namn*
  Namnet på en tidigare definierad användare.
- *Databank*
  Namnet på en tidigare definierad databank.
- *Personlig behörighet*
  Följande personliga behörigheter finns:
  X Exklusiv behörighet. Användare med X-behörighet kan tilldela andra användare behörighet till den aktuella databanken och kan också utföra operationer för sökning och manipulering av data samt definiering av tabeller etc.
  S Tillåter användaren att söka och manipulera data i databanken.
  R Tillåter användaren att söka och hämta data i databanken.
  P Användaren har inte tillgång till den aktuella databanken.

  För att ta bort en personlig behörighet väljer man alternativet "ta bort access" på DBA-menyn. Därefter anger man namnet på användaren och databanken. Funktionen kan utföras av användare med DBA-ställning eller sådana användare med ställning S som har användarbehörighet X till den aktuella databanken.

Här följer exempel på definiering av personlig behörighet. Med DBA– kommandot tar vi fram DBA-menyn och gör ett val i denna, varefter vi besvarar ett antal ledordsfrågor (nödvändiga vagnreturer har markerats med VR):

Ange val: 3<VR>

Skapa/omskapa access

Användare (max 8 tecken) :BLOM<VR>
Databank (max 8 tecken) :PROCDB<VR>
Behörighet (X/S/R/S) :X<VR>

<<<<Skapad >>>>

Användare (max 8 tecken) :TIB<VR>
Databank (max 8 tecken) :SYSQL<VR>
Behörighet (X/S/R/S) :P<VR>

<<<< Skapad >>>>

Användare (max 8 tecken) :<VR>

Vi ger också ett exempel på borttagning av personlig behörighet (menyn är förstås densamma som ovan):

Ange val: 6<VR>

Ta bort access

Användare (max 8 tecken) :BLOM<VR>
Databank (max 8 tecken) :PROCDB<VR>

<<<< Borttagen >>>>

I det här exemplet tilldelas två användare, BLOM och TIB, personlig behörighet till vissa databanker. BLOM tilldelas X-behörighet till databanken PROCDB. Han kan därmed skapa procedurbibliotek, (kapitel 13). Han kan också, även om han inte har X-ställning (DBA), tilldela andra användare behörighet till databanken PROCDB (med kommandot DBA). TIB tilldelas P-behörighet till databanken SYSQL. Detta leder till att QL-körningen avbryts när TIB försöker använda systemet. Användare måste nämligen ha minst R-behörighet till databanken SYSQL.

För att ta bort en personlig behörighet, väljer man först alternativet "ta bort access" på DBA-menyn. I exemplet ovan förlorar BLOM sin behörighet till databanken PROCDB.

13 Procedurer

13.1 Allmänt om procedurer

Det är vanligt att man regelbundet vill använda ett antal kommandon utförda på samma sätt och i samma ordning. För att slippa skriva in sådana kommandoföljder på nytt varje gång de används skapar man s k *procedurer* som vid behov kan anropas på ett enkelt sätt. Med hjälp av procedurer är det också lätt att skapa menyer, något som starkt underlättar arbetet för användarna.

Procedurerna passar för rutinmässig, planerad datahantering och kompletterar alltså direktanvändning av QL-kommandon för spontan, oplanerad datahantering.

En procedur består av ett antal MIMER/QL-kommandon och/eller speciella *procedurkommandon*. Proceduren lagras i en MIMER-tabell.

I en procedur kan man använda *procedurvariabler* och/eller *frågekommandon*. Användaren tillfrågas angående dessa och utförandet påverkas sedan av de svar som ges.

De speciella procedurkommandona används till följande:

- tilldelning av värden till variabler
- överföring av data från tabeller till variabler
- villkorlig exekvering
- skrivning av text på skärmen
- anrop av en annan del av proceduren
- anrop av en annan procedur

Ett procedurkommando inleds till skillnad från vanliga QL-kommandon alltid med ett plustecken (+), tex +TILLDELA, +FRÅGA. Proceduren består av en eller flera *procedurrader*. I figur 13:1 finns ett exempel på en procedur med flera rader.

Procedurerna skrivs in, ändras och tas bort med hjälp av en speciell programvara i MIMER/QL, *procedureditorn* (avsnitt 13.14).

```
 10  +Kommentar   Procedur som beräknar medellönen för alla
 20  +Kommentar      anställda på ett hotell.
 30  Öppna ANSTDB(ANSTÄLLD,LÖN),HOTELLDB(PERSONAL,HOTELL);
 40  +Tilldela &MEDEL  =0;
 50  +Tilldela &SUMMA=0;

 .. ......

 90  +Inled Medellön Hämta &Lön=LÖN.LÖN,&NAMN= ANSTÄLLD.ANAMN)
100      där HOTELL.HNAMN BM '&HOTEL'
110      och HOTELL.HID=PERSONAL.HID
120      och PERSONAL.ANSTNR=ANSTÄLLD.ANSTNR
130      och ANSTÄLLD.LÖNKOD=LÖN.LÖNKOD;
140  +Nästa Medellön;

... ......

230  +Skriv 'Medellön för ',&HOTEL, ' är ' , &MEDEL, 'kr';
240  +Stäng HOTELLDB(HOTELL,PERSONAL), ANSTDB(ANSTÄLLD,LÖN);
```

Fig 13:1. Exempel på procedur.

13.2 Utförande av en procedur

En procedur utförs med kommandot UTFÖR, vars syntax är:

UTFör <procedurbibliotek><(procedurnamn)>;

Procedurbibliotek är namnet på den tabell där proceduren finns lagrad. Normalt lagras procedurerna i procedurdatabanken PROCDB. Men med kommandot SÄTT PROCDB kan användaren själv definiera ett namn på procedurdatabanken (kapitel 14). Om procedurbibliotekets namn utelämnas, tas användarens namn som standardvärde. Om procedurens namn utelämnas, används TEMP som standardvärde. Man kan ändra standardvärdet för procedurbiblioteket genom att använda kommandot SÄTT PROCEDURBIBLIOTEK. Observera att samma standardvärden används för EDITERA-kommandot, som aktiverar *procedureditorn* (avsnitt 13.14).

En procedur avslutas genom att man skriver ett eller två utropstecken (! eller !!) följt av vagnretur.

Här följer några exempel på utförande av procedurer:

1. Proceduren INMATA i procedurbiblioteket PROCB i databanken PROCDB utförs:

```
QL> Utför PROCB(INMATA);
```

2. Proceduren LISTA i procedurbiblioteket med samma namn som användaren utförs:

```
QL> Utför (LISTA);
```

Om användaren är BLOM erhålls samma resultat med följande kommando:

```
QL> Utför BLOM(LISTA);
```

3. Proceduren TEMP i procedurbiblioteket PROCB utförs:

```
QL> Utför PROCB;
```

Samma resultat erhålls med följande kommando:

```
QL> Utför PROCB(TEMP);
```

4. Proceduren TEMP i procedurbiblioteket med samma namn som användaren utförs:

```
QL> Utför;
```

Om användaren är TIB är föregående kommando likvärdigt med följande:

```
QL> Utför TIB(TEMP);
```

5. Proceduren INMATA i procedurbiblioteket PROCB i databanken NPROCDB utförs:

```
QL> Sätt PROCDB='NPROCDB';
QL> Utför PROCB(INMATA);
```

13.3 Procedurformat

Procedurerna lagras i en tabell med följande format:

```
PROCNAME     * C   8
SEQNO        * I   2
COMMAND        C  68
```

Tabellens kolumner (namnen är på engelska) innehåller procedurnamn, radnummer och kommando. Tabellens namn är detsamma som procedurbibliotekets namn, se ovan. Procedurens namn lagras i kolumnen PROCNAME medan de kommandon som ingår i proceduren lagras i kolumnen COMMAND. Kommandona i en procedur utförs i stigande ordning efter numren i kolumnen SEQNO.

Här följer ett exempel. Ett procedurbibliotek listas med HÄMTA-kommandot, med nedanstående resultat:

```
QL> Öppna/r PROCDB(PROC1);
QL> Hämta PROC1.*;
```

PROCNAME	*SEQNO*	*COMMAND*
INMATA	*10*	*Procedurrad 1;*
INMATA	*20*	*Procedurrad 2;*
.	.	.
.	.	.
.	.	.

n rader hittade

Procedurbiblioteket PROC1 innehåller proceduren INMATA som består av n *procedurrader*.

Procedurbiblioteket kan definieras och användas som vilken annan tabell som helst. Det finns dock ett antal hjälpmedel, *systemprocedurer*, som kan användas när man exempelvis skall skapa och lagra procedurer (avsnitt 13.4). Det finns också som nämnts en speciell *procedureditor* (avsnitt 13.14).

De MIMER/QL-kommandon som ingår i en procedur följer samma syntax som när de används för spontan datahantering. Alla procedurkommandon måste avslutas med ett semikolon (;).

13.4 Systemprocedurer

Systemprocedurerna är ett antal färdiga procedurer som följer med MIMER/QL. De används för att utföra regelbundet återkommande operationer som alla kategorier användare utnyttjar. Systemprocedurerna kan anropas från en meny, som visas på skärmen med följande kommando:

```
QL> Utför SYSPROC(MENY);
```

En systemprocedur avslutas på samma sätt som en vanlig procedur, dvs genom att man skriver ett eller två utropstecken (! eller !!) följt av vagnretur.

Man kan dessutom anropa en systemprocedur direkt med ett kommando:

```
QL> Utför SYSPROC (systemprocedur);
```

Följande systemprocedurer är tillgängliga (systemprocedurens namn anges inom parentes):

- *Definiera procedurbibliotek (DEFPRBIB)*
  Kolumndefinitionerna sätts till standardvärden. Användaren måste ha X-behörighet till procedurdatabanken.
- *Ta bort procedurbibliotek (ELIPRBIB)*
  Användaren måste ha X-behörighet till procedurdatabanken.
- *Ladda ett procedurbibliotek från en sekvensiell fil (LADDBIB)*
  Användaren måste ha S-behörighet till procedurdatabanken. Procedurerna måste först skapas med någon av det aktuella datasystemets editorer. Därefter kan filen laddas in i ett procedurbibliotek. Den sekvensiella filen skall ha följande format:

Position	*Fältinnehåll*
1 – 8	Procedurens namn (vänsterjusterat) – PROCNAME
9 –12	Radnummer – SEQNO
13–80	Kommandon – COMMAND

- *Kopiera ett procedurbibliotek till en sekvensiell fil (KOPBIB)*
  Filen får samma format som vid LADDBIB ovan.
- *Presentera information om MIMER-systemet (INFO)*

13.5 Procedurvariabler

Generellt är en *variabel* ett symboliskt *namn* knutet till ett lagringsutrymme. Värdet på variabeln är det värde som för tillfället finns lagrat i lagringsutrym-

met. Värdet kan ändras genom att man *tilldelar* variabeln ett nytt värde.

Variabler kan tilldelas alfanumeriska *värden*. Storleken på en variabels värde är godtyckligt.

I de fall en variabels värde utgörs av ett tal är det möjligt att utföra aritmetiska heltalsoperationer på variabeln. I värdet får i så fall endast ingå tecknen + och – samt siffrorna 0–9.

När en variabel används i en procedur, måste variabeln föregås av tecknet &.

Ett *variabelnamn* kan innehålla bokstäver, siffror och understrykningstecken (_). Första tecknet måste alltid vara en bokstav. Variabelnamnet får bestå av högst åtta tecken.

Små bokstäver i ett variabelnamn, omvandlas till stora. Namn på *systemvariabler* (avsnitt 13.8) inleds med en asterisk (*).

Nedan ges exempel på tillåtna och ej tillåtna variabelnamn.

Tillåtna	*Ej tillåtna*	*Kommentar*
&LÖN	&FÖRLÅNGTNAMN	För långt variabelnamn, max åtta tecken.
&vAr_1	&1VAR	Namnet måste börja med en bokstav.
&*FEL		Systemvariabel.

Samtliga variabler definieras för genomgående användning – så snart en variabel är definierad är den tillgänglig ända tills systemet övergår till kommandoläge (kommandomarkören QL> visas på skärmen).

13.5.1 Att tilldela variabeln ett värde

Variabeln tilldelas ett värde med procedurkommandot +TILLDELA, vars syntax är:

+**TILldela** &variabel=sträng I aritmetiskt uttryck I +**FRÅGA**-kommando I &variabel;

En *sträng* är en antal tecken omgivna av skiljetecken (' " eller ?). En sträng får inte vara uppdelad på flera rader och den måste omges av samma slags skiljetecken. Nedan ges några enkla exempel på tilldelning av värden i form av strängar.

Kommando	*Kommentarer*
+**Tilldela** &NAMN = 'GUNNAR JOHANSSON';	
+**Tilldela** &ADRESS = 'BRYNÄSVÄGEN 10';	
+**Tilldela** &ORT= /GÄVLE";	Felaktig sats – det måste vara samma skiljetecken såväl före som efter strängen.
+**Tilldela** &NAMN = 'GUNNAR SVENSSON'	Felaktig sats – strängen får inte vara uppdelad på flera rader.
+**Tilldela** &TELEFON = 026–123456;	Felaktig sats – strängen måste vara omgiven av skiljetecken (' " / eller ?).

Aritmetiska uttryck kan innehålla numeriska konstanter, numeriska variabler,

aritmetiska uttryck omgivna av parenteser samt operatorer. Värden som åtskiljs av operatorer kallas *operander.*

Förekommande *operatorer* är följande:

Tecken	*Funktion*
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Uttrycken behandlas med normal prioritetsordning, dvs multiplikation och division före addition och subtraktion. Prioritetsordningen kan ändras med parenteser. En eventuell rest vid division trunkeras (faller bort). Nedan följer några exempel på tilldelning av värden i form av aritmetiska uttryck.

Kommando	Kommentarer
+**Tilldela** &A1=12+3*4;	Medför att A1 = 24
+**Tilldela** &A2=(12+3)*4;	Medför att A2 = 60
+**Tilldela** &A3=12/3*4;	Medför att A3 = 16
+**Tilldela** &A4=12/(3*4);	Medför att A4 = 1
+**Tilldela** &A5=11/12+1/12;	Medför att A5 = 0, eftersom båda divisionerna ger resultatet 0 pga trunkering
+**Tilldela** &A6=-12+2*6+(-6);	Medför att A6 = -6
+**Tilldela** &A7=6*-12;	Felaktig sats – operatorerna måste vara åtskilda med antingen en operand eller också måste -12 sättas inom parentes
+**Tilldela** &A8=(1+2));	Felaktig sats – två högerparenteser men bara en vänsterparentes
+**Tilldela** &A9=(&A1+16)-&A2*&A4;	Medför att A9 = -20. (A1, A2 och A4 enligt ovan)
+**Tilldela** &A10=&NAMN*5;	Felaktig sats – variabeln NAMN innehåller 'Gunnar Johansson' (se strängexemplet ovan) och kan alltså inte ingå i en beräkning

Med hjälp av +FRÅGA-kommandot kan man tilldela variablerna värden via inmatning från tangentbordet. Kommandot medför att en ledtext visas på skärmen och systemet förväntar sig att användaren skriver in ett värde. Syntaxen är:

... +**FRÅga**</**Versaler**> sträng ...

Stränguttrycket måste vara omgivet av skiljetecken (' / " eller ?).

Som exempel tar vi kommandot

+**Tilldela** &STAD = +**Fråga** 'Ange stad: ';

som medför att följande ledtext visas på skärmen:

Ange stad:

Systemet förväntar sig i exemplet att operatören matar in ett värde. Variabeln STAD tilldelas det inmatade värdet. Om strängen 'GÄVLE' anges kommer variabeln STAD att få värdet GÄVLE.

Om man bara trycker på vagnretur eller skriver ett eller flera mellanslag följt av vagnretur när ledtexten visas på skärmen, tolkas detta som en ett tecken lång sträng, vilken innehåller ett mellanslag, dvs ' '.

Data som matas in via tangentbordet kan innehålla såväl stora som små bokstäver. Om små bokstäver inte är signifikanta, kan man begära att inmatade data skall omvandlas till enbart stora bokstäver genom att skriva /VERSALER i kommandot. Vi kan tex skriva in följande kommando:

```
+Tilldela &POSTANST=+Fråga/Versaler'Ange postanstalt:';
```

som medför att nedanstående ledtext visas på skärmen:

Ange postanstalt: Säter

Variabeln POSTANST tilldelas här värdet SÄTER (versaler).

Observera att man inte får skriva mellanslag mellan kommandoordet +FRÅGA och parametern /VERSALER, eftersom tecknet / även används som skiljetecken.

13.5.2 Att presentera variabelvärden

Variablernas värden kan presenteras på skärmen med kommandot +SKRIV. Det är också möjligt att specificera strängar, tex texter, som ska visas på skärmen. Strängar och variabler kan blandas på godtyckligt sätt. Syntaxen är:

```
+SKRiv stränglvariabel <,stränglvariabel> < ... >;
```

Kommatecken (,) mellan strängar och/eller variabler kan ersättas med ett mellanslag.

Här följer några exempel.

Exempel 1:

```
+Skriv ' Anställds namn : ',&NAMN,'      Adress : ' ,&ADRESS;
```

Kommandot medför att följande visas på skärmen:

Anställds namn: GUNNAR JOHANSSON *Adress*: BRYNÄSVÄGEN 10

Exempel 2:

```
+Skriv 'A1,A2,A3',',A4: ',&A1,&A2,&A3 &A4;
```

Kommandot medför att följande visas på skärmen:

A1,A2,A3,A4: 2460161

Exempel 3:

```
+Skriv ;
```

Kommandot medför att en blankrad skrivs på skärmen, dvs <CR><LF>.

13.5.3 Kommentarer i en procedur

För att göra proceduren mer lättläst, kan man använda kommandot +KOMMENTAR för att lägga in kommentarer. Syntaxen är:

```
+KOMmentar text;
```

+KOMMENTAR-kommandot lagras tillsammans med proceduren, men ignoreras när proceduren utförs. Kommentartexten kan innehålla vilka tecken som helst.

13.6 Att använda QL-kommandon i en procedur

Samtliga QL-kommandon kan utföras från en procedur. Det är också möjligt att använda variabler och/eller +FRÅGA-kommandon i ett QL-kommando. I så fall byts variabeln ut mot sitt värde eller, vid +FRÅGA– kommandot, mot det värde som matas in via tangentbordet.

Det finns ingen begränsning när det gäller antalet variabler eller +FRÅGA-kommandon, som kan ingå i ett QL-kommando. Däremot får inte QL-kommandot inledas med ett +FRÅGA-kommando.

Här följer några exempel på QL-kommandon i procedurer.

Exempel 1:

Öppna/s +Fråga 'Ange databank som skall öppnas :';

Kommandot medför att följande ledtext visas på skärmen:

Ange databank som skall öppnas :

Om exempelvis namnet MINDB anges, kommer följande kommando att skickas till systemet:

Öppna/s MINDB;

Exempel 2:

Hämta ANSTÄLLD.* **där** ANSTÄLLD.ANAMN = &NAMN;

Om variabeln &NAMN har värdet 'GUNNAR JOHANSSON', kommer följande kommando att överföras till QL-tolken:

Hämta ANSTÄLLD.* **där** ANSTÄLLD.ANAMN = GUNNAR JOHANSSON;

Kommandot är felaktigt därför att en sträng, som innehåller mellanslag, måste vara omsluten av skiljetecken. Kommandot skall i stället skrivas på följande sätt:

Hämta ANSTÄLLD.* **där** ANSTÄLLD.ANAMN = '&NAMN';

Det senare kommandot ger följande resultat:

Hämta ANSTÄLLD.* **där** ANSTÄLLD.NAMN = 'GUNNAR JOHANSSON';

Exempel 3:

Villkoret i exempel 2 ovan kan bytas ut mot villkor som innehåller ett +FRÅGA-kommando, enligt följande:

Hämta ANSTÄLLD.*
där ANSTÄLLD.ANAMN = **+Fråga** 'Ange den anställdas namn :';

Kommandot medför att nedanstående ledtext visas och vi skriver då in ett namn:

Ange den anställdas namn : GUNNAR JOHANSSON

Vi kommer då att få samma fel som ovan. Det beror även i detta fall på att det

finns ett mellanslag mellan för- och efternamn. Namnet måste därför matas in på följande sätt:

Ange den anställdas namn : 'GUNNAR JOHANSSON'

För att slippa mata in skiljetecknen kan man i stället skriva kommandot på följande sätt:

```
Hämta ANSTÄLLD.*
där ANSTÄLLD.ANAMN = '+Fråga 'Ange den anställdas namn :' ';
```

Nu behöver man inte längre skriva skiljetecknen när namnet matas in från tangentbordet.

Om man använder parametern /VERSALER i +FRÅGA-kommandot, omvandlas små bokstäver till stora:

```
Hämta ANSTÄLLD.*
där ANSTÄLLD.NAMN = '+Fråga/Versaler 'Ange den anställdas namn :' ';
```

Om namnet Gunnar Johansson matas in, omvandlas det nu till GUNNAR JOHANSSON.

13.7 Villkorligt utförande (loopar)

Ibland skall ett QL-kommando eller procedurkommando utföras endast om ett visst villkor föreligger, sk *villkorligt utförande*. Detta åstadkoms med +OM... SÅ-kommandot, vars syntax är:

+OM logiskt uttryck **SÅ** kommando;

Det *logiska uttrycket* har följande form:

Variabel **ROP** variabel I sträng

ROP betyder *relationsoperator* och skall vara någon av följande:

Operator	*Innebörd*
=	lika med
>=	större än eller lika med
>	större än
<=	mindre än eller lika med
<	mindre än
/=	ej lika med
%	inleds med
/%	inleds ej med

Ett logiskt uttryck är antingen *sant*, dvs uppfyllt, eller *falskt*, dvs ej uppfyllt. Om villkoret i det logiska uttrycket är sant, utförs kommandot till höger om SÅ. Om det är falskt utförs ej kommandot till höger om SÅ. Kommandot kan utgöras av godtyckligt QL–kommando eller procedurkommando.

Om båda strängarna kan konverteras till heltal (strängar av heltalstyp), sker en strikt numerisk jämförelse. I annat fall jämförs strängarna tecken för tecken från vänster till höger. Om två strängar i en jämförelse har olika längd, fylls den kortare strängen med avslutande mellanslag.

Här följer några exempel på villkorligt utförande.

Exempel 1:

```
+Tilldela &NAMN='Gunnar Johansson';
+Om &NAMN = 'Gunnar Johansson' så +Skriv 'Namnet är korrekt';
```

SKRIV-kommandot utförs, dvs texten visas på skärmen, eftersom det logiska uttrycket är sant.

Exempel 2:

```
+Tilldela &A1 = 'BBBB';
+Om &A1 < 'ACCC' så +Skriv 'Sant';
```

SKRIV-kommandot utförs inte, eftersom det logiska uttrycket är falskt. Jämförelsen görs från vänster till höger och B är större än A.

Exempel 3:

```
+Tilldela &LÖN = 4500;
+Om &LÖN < '10000' så +Skriv 'Sant';
```

Uttrycket är sant, eftersom 4500 är mindre än 10000.

Exempel 4:

```
+Om '4500' = &LÖN så +Skriv &NAMN;
```

Kommandot är felaktigt, eftersom man måste ange en variabel till vänster om relationsoperatorn.

Exempel 5:

```
+Tilldela &A2 = 'DDD';
+Om &A2 > 'DD' så +Skriv 'Sant';
```

Uttrycket är sant. Den kortaste strängen (DD) fylls ut med mellanslag till höger.

Exempel 6:

```
+Tilldela &NAMN = 'Anders Karlsson';
+Tilldela &NYTTNAMN = 'King Kong';
+Om &NAMN /= &NYTTNAMN så +Skriv 'Olika';
```

Det logiska uttrycket är sant. Strängarnas innehåll är olika.

13.8 Ovillkorliga hopp

Ibland skall ett hopp vid utförandet ske till ett annat ställe i proceduren utan att något speciellt villkor behöver vara uppfyllt. För att åstadkomma ett sådant *ovillkorligt hopp* använder man +HOPPA-kommandot. Syntaxen är:

```
+HOPpa radnummer;
```

Radnumret är det nummer som finns i kolumnen RADNR i procedurtabellen (avsnitt 13.3). Om det angivna radnumret inte finns i proceduren, fortsätter utförandet med det första utförbara kommandot med närmast högre radnummer. Endast heltal får förekomma som radnummer.

Här följer några exempel på ovillkorliga hopp.

Exempel 1:

```
10 +Skriv 'Loop !';
20 +Hoppa 10;
```

Exempel 2:

```
10 +Tilldela &N = 2;

20 +Tilldela &N = &N*&N;
30 +Om &N > '32768' så +Hoppa 60;
40 +Skriv &N;
50 +Hoppa 20;
60 +Skriv ' Klart ! ';
```

13.9 Systemvariabler

De sk *systemvariablerna* innehåller information om olika företeelser under det aktuella arbetspasset. Namn på systemvariabler inleds med en asterisk (*). Systemvariablerna skiljer sig från andra variabler på en viktig punkt: de kan inte tilldelas värden av användaren. Om man ändå försöker, visas ett felmeddelande på skärmen.

Följande systemvariabler finns i MIMER-QL:

Variabelnamn	*Variabeln anger*
*ANTAL	Antalet rader som påträffades vid senaste sökning/uppdatering.
*FEL	Senaste felkoden eller noll (0), om inget fel har inträffat.
*VEM	Användarens namn (det namn som angavs vid inloggningen).
*DATUM	Datum (i formen år-månad-dag).

Systemvariablerna kan utnyttjas i procedurer för åstadkommande av villkorliga hopp. Följande exempel ger prov på detta:

```
10 +Skriv ' Anställda vid datum ',&*DATUM;
20 Öppna/s +Fråga 'Vilken databank skall öppnas?: ';
30 +Om &*FEL /= '0' så +Hoppa 80;
```

Ovanstående kontrollerar att operationen gick bra. Vi fortsätter:

```
35 +Tilldela &NAMN = 'Gunnar Johansson';
40 Hämta ANSTÄLLD.* där ANSTÄLLD.ANAMN = '&NAMN';
50 +Om &*ANTAL > '0' så +Hoppa 100;
```

vilket säger att om inga rader påträffades, så försök igen:

```
60 +Tilldela &NAMN = +Fråga 'Ge nytt namn : ';
70 +Hoppa 40;
80 +Skriv 'Databanken är inte definierad!';
90 +Stopp;
```

Hela ovanstående procedur medför att följande text visas på skärmen – vi ger också in ett svar:

Anställda vid datum 1984–10–18

Vilken databank skall öppnas? : ANSTDB

13.10 Att överföra data till procedurer

Med speciella kommandon kan man överföra data från tabeller till en procedur. För detta finns en motsvarighet till HÄMTA-kommandot för radvis åtkomst.

Vid det vanliga QL-kommandot HÄMTA skrivs rad efter rad ut på terminalen. När data skall överföras till procedurer kan man i stället se det som att kolumnvärdena först lagras i en imaginär tabell. Samtliga rader som uppfyller angivet villkor överförs till denna tabell. Från tabellen kan sedan raderna läsas in till proceduren.

Den imaginära tabellen måste definieras/öppnas med +INLED-kommandot. Den imaginära tabellen måste ha en *identifierare*, ett namn som får bestå av högst åtta tecken. Dessutom kan ett sökvillkor anges. Om sökvillkor utelämnas, överförs samtliga rader till den imaginära tabellen. Syntaxen är:

```
+INLed identifierare HÄMta (variabel=tabref.kolumnnamn<,...>)
                    <DÄR ...>;
```

Villkorssatsen till höger om DÄR, skrivs på samma sätt som villkorssatsen i HÄMTA-kommandot. Kolumnvärden från flera olika tabeller kan överföras, förutsatt att tabellerna finns med i villkorssatsen. Tabellerna måste vara öppnade (med ÖPPNA-kommandot) innan man kan använda dem.

+NÄSTA-kommandot används för att läsa en rad från den imaginära tabellen. Systemvariabeln *ANTAL har värdet 1 när en ny rad har lästs in till proceduren och värdet 0 (noll) när det inte finns några fler rader. Syntaxen är:

```
+NÄSta identifierare;
```

När inga fler rader skall läsas från tabellen, dvs när det inte finns fler rader, måste den imaginära tabellen stängas. Syntaxen är:

```
+AVSluta identifierare;
```

När det inte finns fler rader i tabellen, dvs *ANTAL = 0, kommer ett försök att läsa nästa rad att resultera i ett felmeddelande. Den imaginära tabellen är öppen ända tills den uttryckligen stängs eller arbetspasset avslutas (systemet övergår till kommandoläge, QL>). När man försöker öppna en redan öppnad imaginär tabell, visas ett felmeddelande på skärmen.

För varje imaginär tabell som öppnas reserverar systemet minnesutrymme, som kan återanvändas för annat när tabellen har stängts. Därför bör man stänga alla imaginära tabeller när de inte längre behövs. På så sätt friställs utrymme i minnet.

Här följer ett exempel. Vi skriver följande procedur:

```
 10 +Kommentar  Procedur som beräknar medellönen för alla
 20 +Kommentar      anställda på ett hotell.
 30 Öppna ANSTDB(ANSTÄLLD,LÖN),HOTELLDB(PERSONAL,HOTELL);
 40 +Tilldela &MEDEL  =0;
 50 +Tilldela &SUMMA=0;
 60 +Tilldela &N      =0;
 70 +Tilldela &HOTEL  =+Fråga/Versaler 'Ange hotellnamn :';
 80 +Skriv;
 90 +Inled Medellön Hämta &Lön=LÖN.LÖN,&NAMN=ANSTÄLLD.ANAMN)
100    där HOTELL.HNAMN BM '&HOTEL'
110    och HOTELL.HID=PERSONAL.HID
120    och PERSONAL.ANSTNR=ANSTÄLLD.ANSTNR
130    och ANSTÄLLD.LÖNKOD=LÖN.LÖNKOD;
140 +Nästa Medellön;
```

```
150 +Om &*ANTAL='0' Så +Hoppa 200;
160 +Tilldela &SUMMA=&SUMMA+&LÖN;
170 +Tilldela &N=&N+1;
180 +Skriv &NAMN ,'har lön : ', &LÖN;
190 +Hoppa 140;
200 +Avsluta Medellön;
210 +Om &N /='0' Så+Tilldela &MEDEL=&SUMMA/&N;
220 +Skriv;
230 +Skriv 'Medellön för ',&HOTEL, ' är ' , &MEDEL, 'kr';
240 +Stäng HOTELLDB(HOTELL,PERSONAL), ANSTDB(ANSTÄLLD,LÖN);
```

När proceduren utförs visas först en ledtext — vi svarar med hotellnamnet PLAZA och får som resultat:

Ange hotellnamn: PLAZA

KARLSSON ÅKE	*har lön : 6400*
MATTSSON HELENA	*har lön : 6400*
MATTSSON OLLE	*har lön : 7100*
JANSSON ANDERS	*har lön : 7200*
JOHANSSON GUNNAR	*har lön : 8100*
GUNNARSSON PIA	*har lön : 8800*

Medellönen för PLAZA är 7316 kr

Ovanstående procedur visar också hur rader från olika tabeller överförs till en imaginär tabell som inläses radvis med hjälp av Nästa-kommandot.

13.11 Menyer

Med hjälp av +MENY-kommandot kan man tillfälligt anropa en procedur från en annan. Procedurerna kan väljas i en meny med ett alternativ för varje procedur. Observera dock att man inte kan anropa en procedur i ett annat procedurbibliotek. Syntaxen är:

+MENy procedurnamn sträng;

När +MENY-kommandot påträffas, räknas *menyvalsräknaren* upp ett steg (den är från början noll). Aktuellt nummer samt den menytext som anges i +MENY-kommandot visas på skärmen. Detta upprepas för varje nytt +MENY-kommando och upphör först när något annat kommando än +MENY påträffas. Användaren skall därefter ange vilken procedur som ska anropas genom att ange numret för proceduren. Utförandet fortsätter därefter med vald procedur.

Följande procedur ger angivet resultat i form av en meny:

```
+Meny PUPPDAT ' Uppdatera.';
+Meny PINMAT ' Mata in.';
+Meny PLISTA ' Lista.';
```

1 Uppdatera.
2 Mata in.
3 Lista.
-?

Om vi svarar med siffran 1 utförs proceduren PUPPDAT etc.

Om användaren matar in ett felaktigt värde, tex en siffra som inte finns i menyn, upprepas uppmaningen att ange ett värde (-?). Om man trycker på vagnretur i stället för att ge ett värde upprepas hela menyn.

QL- eller procedurkommandon som kommer efter ett antal +MENY-kommandon utförs inte, som i följande exempel:

```
+Meny PROC1 ' Starta PROC1';
+Meny PROC2 ' Starta PROC2';
+Skriv ' Efter menyn !';
```

Texten 'Efter menyn !' kommer aldrig att visas på skärmen.

När alla kommandon i den valda proceduren har utförts, fortsätter utförandet i början på den procedur, som innehåller menyn. Här följer ett exempel:

```
MENY    10   +Skriv ' Ange alternativ :';
MENY    20   +Meny PROC1 ' Starta PROC1';
MENY    30   +Meny PROC2 ' Starta PROC2';
PROC1   10   +Skriv 'Detta är PROC1';
PROC2   10   +Skriv 'Detta är PROC2';

QL> Utför (MENY)
-?2

Ange alternativ :

1  Starta PROC1
2  Starta PROC2
-?2

Detta är PROC2

Ange alternativ :

1  Starta PROC1
2  Starta PROC2
-?
```

Enda sättet att lämna menyn utan att starta en ny procedur är att skriva ett utropstecken (!) som svar på frågan -?, som i nedanstående exempel:

```
MENY    10   +Skriv 'Ange alternativ :';
MENY    20   +Meny PROC1 ' Starta PROC1';
MENY    30   +Meny PROC2 ' Starta PROC2';
```

För att komma tillbaka till kommandoläge, skriver man ett utropstecken (!) istället för en siffra.

```
QL> Utför (MENY);

Ange alternativ :
1  Starta PROC1
2  Starta PROC2
-?!
QL>
```

13.11.1 Att bygga upp ett menyträd

Man kan skapa menyer och undermenyer på olika nivåer i en trädstruktur genom att koppla samman procedurer, som innehåller +MENY-kommandon. Genom att anropa en meny från en annan meny, kommer man till nästa lägre nivå. Genom att skriva ett utropstecken (!) istället för att välja ett alternativ i

menyn, får man fram föregående meny, dvs menyn på närmast högre nivå. Antalet nivåer är obegränsat.

Vi ger ett exempel på byggande av menyträd. Anta att procedurbiblioteket innehåller följande:

```
MENY    10   +Meny PUPPDAT ' Uppdatera.';
MENY    20   +Meny PINMAT ' Mata in.';
MENY    30   +Meny PLISTA ' Lista.';
.       .    .
.       .    .
PINMAT  10   +Meny INANST ' Inmatning i tabellen ANSTÄLLD.';
PINMAT  20   +Meny INHOT 'Inmatning i tabellen HOTELL.';
.       .    .
.       .    .
PLISTA  10   +Meny LISTANST ' Lista data från tabellen ANSTÄLLD.';
PLISTA  20   +Meny LISTHOT 'Lista data från tabellen HOTELL.';
.       .    .
.       .    .
```

Så här kan det se ut på skärmen när proceduren MENY uförs och val i menyer sker:

```
QL> Utför (MENY);

1  Uppdatera.
2  Mata in.
3  Lista.
-?2

1  Inmatning i tabellen ANSTÄLLD.
2  Inmatning i tabellen HOTELL.
-?1
```

Här startar proceduren INANST. När detta är klart visas följande och val görs:

```
1  Inmatning i tabellen ANSTÄLLD.
2  Inmatning i tabellen HOTELL.
-?!

1  Uppdatera.
2  Mata in.
3  Lista.
-?3

1  Lista data från tabellen ANSTÄLLD.
2  Lista data från tabellen HOTELL.
-?2
```

Här startar proceduren LISTAVD. När detta är klart visas följande och val görs:

```
1  Lista data från tabellen ANSTÄLLD.
2  Lista data från tabellen HOTELL.
-?!

1  Uppdatera.
2  Mata in.
3  Lista.
-?!

QL>
```

Ovanstående exempel kan beskrivas grafiskt i en hierarkisk trädstruktur som i figur 13:2.

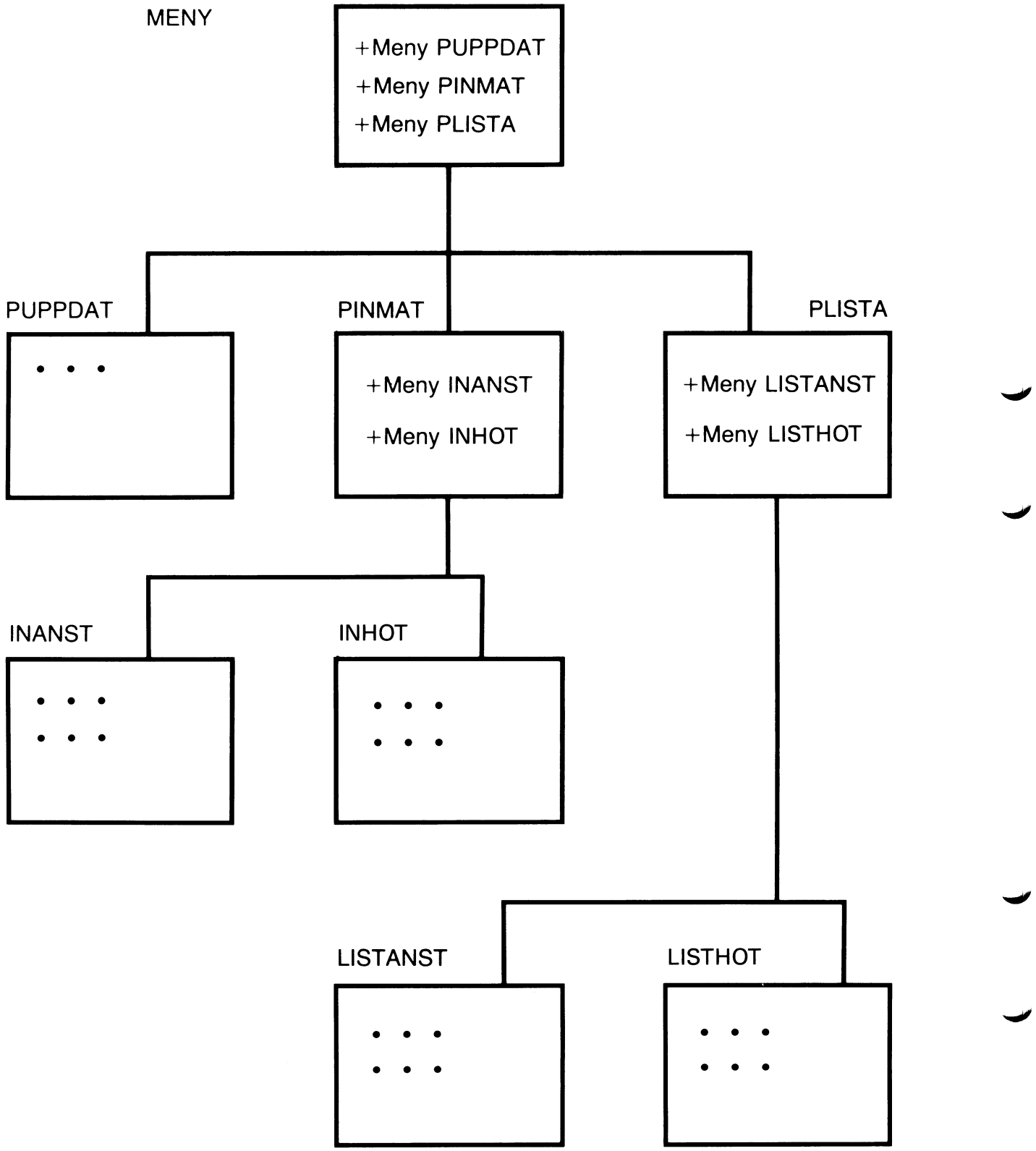


Fig. 13:2. Exempel på menyträd.

För att avbryta utförandet och återgå till kommandoläge, skriver man ett utropstecken (!) när man befinner sig i den första menyn, dvs menyn på högsta nivån. För att avbryta nuvarande procedur från en meny på en lägre nivå krävs två utropstecken (!!). Vi ger ett exempel:

```
QL> Utför (MENY);

1  Uppdatera.
2  Mata in.
3  Lista.
-?2

1  Inmatning i tabellen ANSTÄLLD.
2  Inmatning i tabellen HOTELL.
-?!!

QL>
```

Man kan också gå tillbaka till föregående meny genom att skriva ett utropstecken (!) som svar på frågan vid ett +FRÅGA-kommando:

```
LISTANST   10 +Tilldela &NAMN = +Fråga 'Ange namn på anställd :';
.              . .
.              . .

QL> Utför (MENY);

1  Uppdatera.
2  Mata in.
3  Lista.
-?3

1  Lista data från tabellen ANSTÄLLD.
2  Lista data från tabellen HOTELL.
-?1

Ange namn på anställd : !
1  Lista data från tabellen ANSTÄLLD
2  Lista data från tabellen HOTELL.
-?
```

Med två utropstecken som svar på frågan i ett +FRÅGA-kommando avbryts utförandet av den nuvarande proceduren.

13.12 Procedurnivåer (procedurkapsling)

Eftersom man kan använda alla QL-kommandon i en procedur, är det också möjligt att starta en procedur från en annan procedur. Begreppet *procedurnivå* används för att ange antalet aktiva procedurer.

När tex UTFÖR-kommandot ges i kommandoläge, startas den första proceduren (procedurnivå 1). Om det finns ett nytt UTFÖR-kommando i denna procedur, startas nästa procedur (procedurnivå 2). När denna procedur är klar, utförs kommandot som kommer efter UTFÖR-kommandot i proceduren på nivå 1. Detta kan jämföras med att man i ett högnivåspråk som COBOL, FORTRAN, BASIC el dyl anropar en subrutin, dvs en avgränsad del av ett datorprogram med en bestämd uppgift.

Det maximala antalet procedurnivåer beror på vilket datorsystem som används. Genom att använda kommandot UTFÖR SYSPROC(INFO) kan man se vad som gäller för aktuellt system härvidlag.

Användaren kan avbryta utförandet av den aktuella proceduren genom att ange två utropstecken (!!) som svar på en fråga i proceduren.

När ett +STOPP-kommando påträffas i proceduren avbryts den. Om det finns en högre procedurnivå fortsätter utförandet där. I annat fall övergår systemet till kommandoläge. Syntaxen är:

+STOpp;

Här följer några exempel på procedurnivåer.

Exempel 1:

```
PROC1   10   +Skriv 'Detta är procedur 1';
PROC1   20   Utför (PROC2);
PROC1   30   +Skriv 'Tillbaka till procedur 1';
.       .    .
.       .    .
PROC2   10   +Skriv 'Detta är procedur 2';
PROC2   20   +Tilldela &FORTS = +Fråga/Versaler 'Vill du fortsätta :';
PROC2   30   +If &FORTS % 'N' så +Stopp;
PROC2   40   +Hoppa 10;
QL> Utför (PROC1);
Detta är procedur 1
Detta är procedur 2
Vill du fortsätta : Ja
Detta är procedur 2
Vill du fortsätta: Nej
Tillbaka till procedur 1
QL>
```

STOPP-kommandot medför att körningen av procedur 2 avbryts.

Exempel 2:

```
QL> Utför (PROC1);
Detta är procedur 1
Detta är procedur 2
Vill du fortsätta : Ja
Detta är procedur 2
Vill du fortsätta : !!
Tillbaka till procedur 1
QL>
```

Två utropstecken medför att proceduren PROC2 avbryts och systemet övergår till kommandoläge.

Exempel 3:

Om man ändrar PROC2 till

```
PROC2   10   +Skriv 'Detta är procedur 2';
PROC2   20   +Tilldela &FORTS = +Fråga 'Vill du fortsätta :';
PROC2   30   +Om &FORTS % 'J' så +Hoppa 10;
```

blir resultatet följande:

```
QL> Utför (PROC1);
Detta är procedur 1
Detta är procedur 2
Vill du fortsätta : Ja
Detta är procedur 2
Vill du fortsätta : Nej
```

```
Tillbaka till procedur 1
QL>
```

PROC2 avbryts när det inte finns fler kommandon.

13.13 Procedureditorn

I MIMER/QL finns en *procedureditor* som används för att skapa procedurer och för att underhålla procedurer, dvs göra ändringar, borttagningar och tillägg i dem. Den är av typen *radorienterad editor*, vilket innebär att man kan editera en procedurrad i taget. Procedureditorn har en egen uppsättning kommandon (beskrivs detaljerat i bilaga D).

Procedurerna lagras i *bibliotek*, som i sin tur finns i databanker. Biblioteken är i själva verket tabeller och procedurerna utgörs av rader i dessa tabeller.

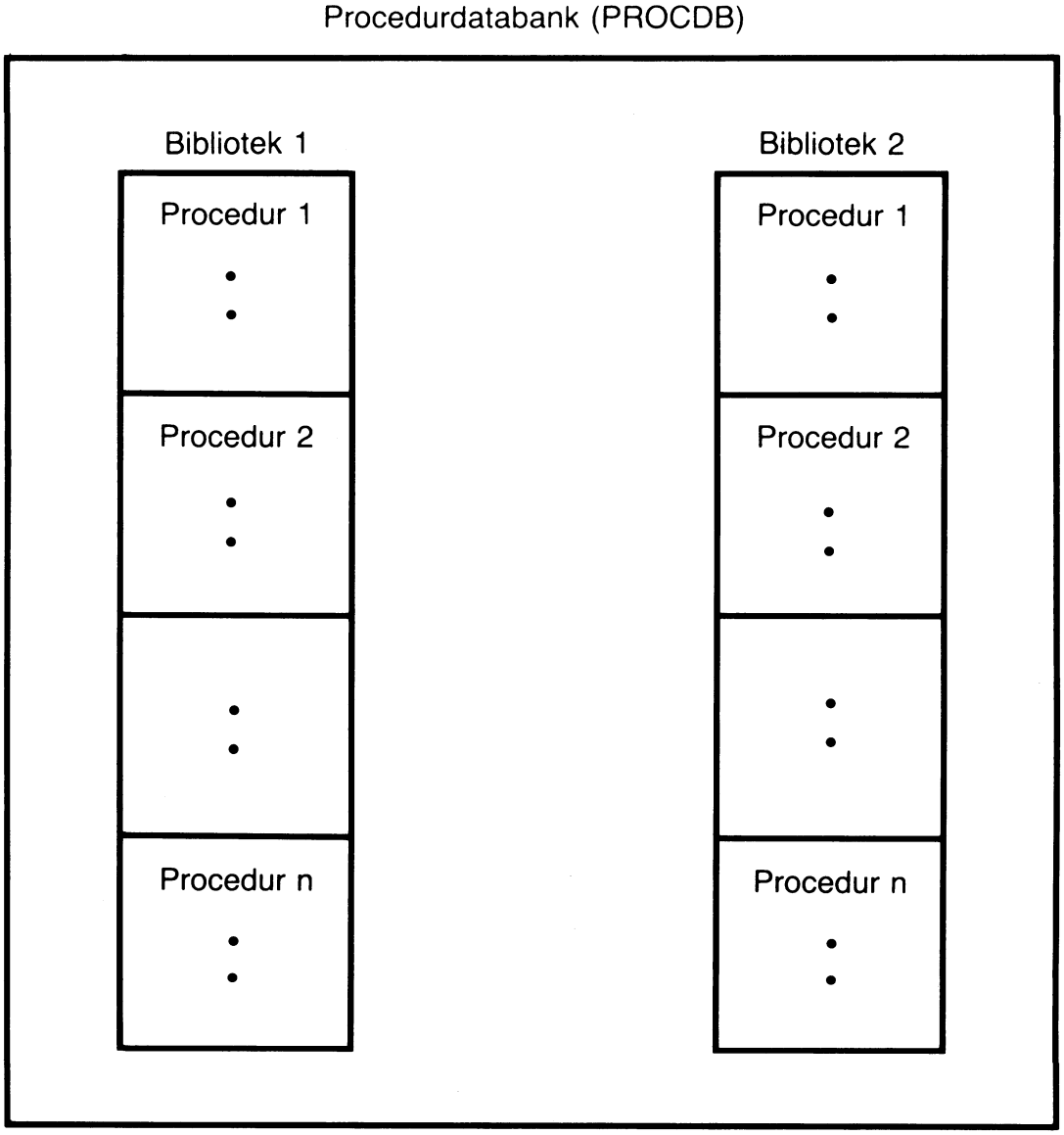


Fig 13:3. Procedurdatabankens struktur.

13.13.1 Att starta editorn

Editorn startas med QL-kommandot EDITERA som har följande syntax:

EDItera <bibliotek> <(procedur)>;

Databank, procedurbibliotek och procedur väljs på följande sätt. Databank förutsätts vara den som anges av tillståndsvariabeln PROCDB (kapitel 14). Om bibliotek utelämnas förutsätts bibliotek vara det som anges av tillståndsvariabeln PROCBIBLIOTEK (kapitel 14). Standardvärden för databank är PROCDB och för bibliotek användarens namn. Om procedurens namn utelämnas i kommandot, väljs proceduren ARB. Detta procedurnamn är *reserverat* av systemet, vilket innebär att ingen annan procedur får börja med bokstäverna ARB. ARB11 och ARBETE är alltså inte tillåtna. Däremot kan ARB och AR användas. Det finns ytterligare två namn som är reserverade – SPAR och SIST.

Här följer några exempel på EDITERA-kommando:

QL> **Editera** ALLMÄNT(INIT);

Editorn kommer att söka efter proceduren INIT i procedurbiblioteket ALLMÄNT. Editorn förväntar sig att biblioteket finns i den databank som anges av tillståndsvariabeln PROCDB. Med kommandot BESKRIV SÄTTNINGAR kan man ta reda på databankens namn.

QL> **Editera** PRIVAT;

Editorn kommer att söka efter proceduren TEMP i biblioteket PRIVAT, som förväntas finnas i den databank som anges av PROCDB.

QL> **Editera** (BUDGET);

Editorn kommer att söka efter proceduren BUDGET i det bibliotek som anges av tillståndsvariabeln PROCBIBLIOTEK. Med kommandot BESKRIV SÄTTNINGAR kan man ta reda på bibliotekets namn. Standardvärde är som nämnts användarens namn, dvs inloggad användare. Biblioteket förväntas finnas i den databank som anges av tillståndsvariabeln PROCDB.

QL> **Editera**;

Editorn kommer att söka efter proceduren TEMP i det bibliotek som anges av tillståndsvariabeln PROCBIBLIOTEK. Databankens namn är definierat i PROCDB.

QL> **Sätt** PROCDB='FÖRSÄLJ',BIBLIOTEK='KALKYL';
QL> **Editera** (BUDGET);

Editorn kommer att söka efter proceduren BUDGET i biblioteket KALKYL, som finns i databanken FÖRSÄLJ.

Innan man kan använda editorn, måste man definiera ett procedurbibliotek. Det enklaste sättet är att använda systemproceduren DEFPRBIB (avsnitt 13.4). Definieringen av ett bibliotek innebär att en tabell skapas vilket kräver att användaren har X-behörighet till procedurdatabanken. Men för att använda editorn krävs inte X-behörighet, det räcker med S-behörighet.

En användare som vill skapa ett procedurbibliotek men saknar X-behörighet kan be databasadministratören (DBA) att skapa biblioteket åt sig.

När man skapar nya procedurer eller uppdaterar gamla är det praktiskt att utnyttja TEMP i ett första steg. Om man i stället skulle skriva in eller uppdatera proceduren direkt, går det nämligen inte att ändra sig under arbetets gång och upphäva något som redan skrivits in. Därför är det bättre att först skriva in proceduren i preliminär form som ARB eller uppdatera en kopia av proceduren.

Vid uppdatering kopierar man således först den procedur som skall uppdateras till TEMP med KOPIERA-kommandot. Därefter editerar man kopian. Först när man är nöjd med uppdateringen, kanske efter upprepade tester, ersätter man den gamla proceduren med den uppdaterade. Detta kan utföras med editeringskommandona RADERA och SPARA eller kommandot DÖP-OM. Proceduren i TEMP sparas alltid när man går från editorn till kommando eller procedurläge. Observera att proceduren i TEMP sparas när man avslutar arbetspasset och därför kan användas även i nästa arbetspass.

Vi skall nu beskriva editorns två lägen, *inmatningsläget* som används vid inmatning av nya procedurer, och *editeringsläget* som används vid uppdatering av procedurer.

13.13.2 Inmatningsläget

När man har gett EDITERA-kommandot, visas editeringssymbolen ED> på skärmen. Det innebär att editorn är klar att användas, dvs väntar på ett editeringskommando.

Om man nu skriver ett utropstecken (!), går editorn över till inmatningsläge. Radnummer i proceduren anges automatiskt av systemet i steg om tio, således 10,20,30 etc.

Om man skriver två utropstecken (!!) eller !SLUTA, övergår systemet till kommandoläge, dvs kommandosymbolen QL> visas på skärmen.

Om man skriver ett utropstecken (!) eller ett utropstecken direkt följt av ett editeringskommando *utan några mellanslag emellan*, övergår editorn till editeringsläge (avsnitt 13.14.3).

Här ges ett exempel på inmatning av en procedur.

På skärmen	*Kommentarer*
QL> **Editera** (TEST)	Proceduren TEST i biblioteket BIBLIOTEK skall editeras
ED>!	Gå över i inmatningsläge
10 Inmatning	1 Skriv in Inmatning 1
20 Inmatning 2	Skriv in Inmatning 2
30 **!Lista**	Gå över i editeringsläge och lista hela proceduren.
10 Inmatning 1	
20 Inmatning 2	
ED>	Editorn väntar på nästa EDIT-kommando

13.13.3 Editeringsläget

I editeringsläge förväntar sig editorn att ett editeringskommando matas in. Editeringssymbolen ED> visas på skärmen.

Här följer en funktionsgrupperad uppräkning av kommandon som kan ges i editeringsläge:

Kommandon	*Användning*
LAGRA, KORRIGERA, ORDNA, SUDDA, ANDRA	För operationer på strängar inom raden
LAGRA, ERSÄTT, RADERA	För operationer på en hel rad.
KOPIERA, FLYTTA	För operationer på flera rader.
NUVARANDE, RADERA, DÖP-OM, SPARA, OMNUMRERA	För operationer på hela proceduren.
INVENTERA	För operationer på procedurbiblioteket.
HJÄLP eller ?	För att visa hjälpinformation.

Exempel på operationer i editeringsläge ges i avsnitt 13.14.4.

13.13.4 Editeringskommandonas syntax

Den generella syntaxen för editeringskommandona ser ut på följande sätt:

Kommandoord <radnummer,><procedur,><radnummerintervall> </sträng1>/<sträng2/>

eller

Kommandoord <radnummer,><procedur,><radnummerintervall> #kolumn#sträng2#

Syntaxen för LÄGG-TILL- och ERSÄTT-kommandona ser ut på följande sätt:

Kommandoord radnummer

Här beskrivs de olika delarna av ovan angivna syntaxer.

Begrepp	*Innebörd*
Kommando	Editeringskommandot i kortform (avsnitt 13.14.4).
Radnummer	Radnummer som skall bearbetas, tex kopieras eller flyttas. Används ihop med kommandona KOPIERA, FLYTTA, LÄGG-TILL och ERSÄTT.
Procedur	Beskrivs enligt syntaxen <<databank.>procedurbibliotek.>procedur Om databank och/eller procedurbibliotek utelämnas, gäller aktuell databank och aktuellt bibliotek.
Databank	Namn med högst 8 alfanumeriska tecken.
Procedurbibliotek	Namn med högst 8 alfanumeriska tecken.
Procedur	Namn med högst 8 alfanumeriska tecken.
Radnummer-intervall	Skrivs <låg><-<hög>>. Definierar radnumren på följande sätt: låg | låg- | låg-hög | -hög | låg, n1 Låg = upp till fyra siffror som definierar den undre gränsen eller *. Hög = upp till fyra siffror som definierar den övre gränsen eller *. n1 = antal rader. * = aktuell rad, dvs radnumret för den senaste raden som har påverkats med ett editeringskommando.

Följande exempel åskådliggör vilka standardvärden som gäller:

```
10      låg = 10 och hög = 10
50—     låg = 50 och hög = 9999
20–60   låg = 20 och hög = 60
  —70   låg =  0 och hög = 70
(tomt)  låg =  0 och hög = 9999
```

Sträng1 — Högst 68 tecken omgivna av skiljetecken (/ ' ? eller ").

Sträng2 — Högst 68 tecken omgivna av skiljetecken (samma som vid Sträng1). Om 'kolumn' anges kan även mellanslag användas som Skiljetecken.

\# — Indikerar en bestämd kolumnposition.

Kolumn — Kolumnens nummer inom raden. Första kolumnen har nummer 1. Kan endast användas med kommandona LAGRA, KORRIGERA och SUDDA.

Här ges några exempel på strängar.

```
/GAMMAL TEXT/NY TEXT/
?GAMMAL TEXT?NY TEXT?
'GAMMAL TEXT'NY TEXT'
"GAMMAL TEXT"NY TEXT"
```

Om en rad som inleds med ett radnummer följt av ett mellanslag matas in händer endera av följande:

- Om radnumret redan existerar, ersätts den gamla raden med den nya.
- Om radnumret är nytt läggs den nya raden in.

Exempel:

Rad	*Kommentar*
120 NY TEXT TILL RAD 120	Det finns redan en rad 120. Denna ersätts med den nya rad 120.
125 DETTA ÄR EN NY RAD	Ingen rad 125 finns, varför raden läggs in som ny.

Kommandoorden som används i editeringskommandona skrivs i kortform i likhet med alla andra kommandoord i MIMER/QL-systemet. Kortformen av ett kommandoord är alltid de tre första bokstäverna.

I kommandon som arbetar på strängar används skiljetecknet /. Tecknet kan bytas ut mot något av tecknen " ' eller ? (avsnitt 13.14.3).

Editeringskommandona beskrivs detaljerat i bilaga D enligt följande mönster:

- syntax
- funktionsbeskrivning
- parameterbeskrivning
- exempel

14 Tillståndsvariabler

I detta kapitel beskrivs systemets *tillståndsvariabler*. Det är ett antal variabler vilkas värden påverkar MIMER/QL:s arbetssätt i olika avseenden, tex om viss utskrift ej skall göras, om transaktionshantering skall användas, vilken radlängd skrivfilen skall ha osv.

När man startar systemet har alla de här variablerna vissa standardvärden, av vilka en del är beroende av aktuellt datorsystem. Man kan ändra värdena med kommandona SÄTT och ÅTERSTÄLL. Med SÄTT-kommandot tilldelar man variablerna önskade värden och med ÅTERSTÄLL-kommandot återställer man variablerna till deras ursprungsvärden, standardvärdena.

Med hjälp av kommandona BESKRIV SÄTTNINGAR och VISA SÄTTNINGAR presenteras variablerna och deras värden på skärmen.

Syntaxen för SÄTT-kommandot är:

SÄTt variabelspecifikation <,variabelspecifikation,...>;

Variabelspecifikationerna beskrivs nedan. Om variabelvärdet angivet efter =-tecken är ett namn, tex SB='SB', och inte ett bestämt värde betyder det att värdet är beroende av datorsystem. De faktiska värdena på sådana variabler presenteras på skärmen med hjälp av kommandot UTFÖR SYSPROC(INFO) (bilaga A).

Med ÅTERSTÄLL-kommmandot återställs variablernas värden till ursprungsvärdena. Syntaxen är:

ÅTErställ <variabel><,variabel,...>;

Om variabel utelämnas återställs samtliga variabler. Dessutom stängs alla öppna databanker och skrivfilen som används för utskrifter. Dessutom tas samtliga alias bort.

Kommandot ÅTERSTÄLL SKRIVFIL stänger skrivfilen.

Här följer förteckningen över variabler. Kortformen vid användning av variabeln i ett kommando anges med versaler.

Variabel		**Kommentar**
Standard-värde	*Alternativvärde*	
EJUtskrift	UTSkrift	I läge UTSKRIFT undertrycks viss information, som normalt skickas till terminalen (skärmen). Det som undertrycks är ru-

briker och information om borttagna, uppdaterade eller funna rader, som normalt visas vid användning av HÄMTA- och LAGRA-kommandona. Läge UTSKRIFT kan exempelvis användas i procedurläge om man vill dölja viss information för den, som kör proceduren.

RA=RA n

RA-variabeln anger antalet rader som skall skrivas ut mellan två rubriker på skärmen. RA anger alltså sidlängden. Om RA sätts till 0 (noll), skrivs sidrubriken bara en gång. Detta gäller tillsammans med HÄMTA-kommandot.

RA-variabeln används också vid listningar. En del kommandon, tex HÄMTA och LISTA, genererar i vissa fall en sådan mängd information att den inte får plats på en skärm (20 rader). Om RA > 0, avbryts presentationen när angivet antal rader har matats ut till terminalen. Operatören kan då välja att

- fortsätta presentationen genom att skriva ett N(=nästa) och trycka på vagnretur. Det räcker med att trycka på vagnretur.
- fortsätta presentationen genom att skriva ett F(=fortsätt) och trycka på vagnretur. I detta fall listas resten av informationen, dvs presentationen avbryts inte nästa gång skärmen är full osv.
- avbryta presentationen genom att skriva ett H(=hoppa) och trycka på vagnretur.

Om RA sätts lika med 0 (noll), avbryts inte presentationen på skärmen, dvs hela listan visas i en följd.

LÅNg KORt

Anger om detaljerad (LÅNG) eller kortfattad (KORT) hjälptext skall presenteras på skärmen vid begäran om hjälp.

RB=RB n

Radlängd – värdet på n anger antal tecken per rad, som skall skickas till terminalen.

MI=100 n

Anger hur ofta meddelande om antal hittills utförda operationer skall skickas till terminalen. Denna funktion kan användas ihop med kommandon som arbetar på stora mängder information (stort antal rader eller poster). Sådana operationer kan ta lång tid och då är det värdefullt att få meddelande om att någonting verkligen händer. Kan användas tillsammans med kommandona KOPIERA, SKRIV, ÄNDRA, RADERA (med DÄR-sats) och HÄMTA. Om MI sätts lika med 0 (noll) eller om läge UT-

		SKRIFTER är satt, skickas inga meddelanden till terminalen.
ARBdb=ARBDB	databanksnamn	Anger namnet på den databank som ska användas som arbetsdatabank. Observera att databanken måste vara definierad som en vanlig MIMER-databank. Se kommandot DBA.
SKRivfil= SKRIVF	filnamn	Anger namnet på den fil som skall användas av SKRIV-kommandot. Om det redan finns en öppen utskriftsfil när SÄTT SKRIVFIL ges, stängs denna fil och den nya öppnas.
BIBliotek= BIBLIOTEK	biblioteksnamn	Anger namnet på aktuellt procedurbibliotek.
PROCdb= PROCDB	databanksnamn	Anger namnet på den databank som ska användas för lagring av procedurerna.
SPÅra	EJSpåra	I spårläge kommer samtliga QL-kommandon upp på skärmen.
ILÄngdm= ILÄNGDm	n	Anger antal tecken, n, som ska reserveras för kolumnvärden av typ I_m (heltal) när de skall visas på skärmen eller skrivas ut på skrivaren. Antalet siffror som kan lagras i en kolumn av typ I, dvs precisionen, beror på kolumnens storlek. Antalet siffror som behövs för presentationen varierar alltså med kolumnstorleken. Storleken på m är systemberoende. Värdet skall ligga mellan 1 och ILÄNGDB. Standardvärdena för n och m (ILÄNGD1, ILÄNGD2, osv.) visas på skärmen genom att man ger kommandot UTFÖR SYSPROC(INFO)(bilaga A).
FLÄngdm= FLÄNGDm	n	Anger antalet tecken, n, som ska reserveras för kolumnvärden av typ F_m (flyttal) när de skall visas på skärmen eller skrivas ut på skrivaren. Antalet siffror, som kan lagras i en kolumn av typ F, dvs precisionen, beror på kolumnens storlek. Därför varierar antalet siffror som behövs för presentationen med kolumnstorleken. Storleken på m är systemberoende. Värdet skall ligga mellan 3 och FLÄNGDB. Standardvärdena för n och m (FLÄNGD3, FLÄNGD4, osv.) visas på skärmen genom att man ger kommandot UTFÖR SYSPROC(INFO)(bilaga A).
EJTransaktion	TRAnsaktion	Anger om läsoperationer i arbetstabellen, som initieras med HÄMTA-kommandot, skall utföras inom ramen för en transaktion eller inte. Denna variabel ignoreras om databanken TRANSDB inte är definierad.
EJEko	EKO	Anger om angivna kommandon skall ekas på skärmen eller inte, dvs om de ska visas som datorn tagit emot dem eller ej. Detta är

		användbart i system som tillåter loggning av kommandon på skrivaren eller en fil. På detta sätt kan både det som går till MIMER/QL och det som kommer från MIMER/QL loggas. Om EKO-läge gäller när en procedur utförs, kommer alla procedurkommandon att visas på skärmen. Detta kan tex användas för felsökning i procedurer.
AEXitc1=20	n	Anger antalet felmeddelanden från MIMER/DB som är tillåtna innan körningen avbryts.
BEKräfta	EJBekräfta	Anger om kommandon för borttagning och uppdatering av data måste bekräftas av användaren innan kommandot utförs. Om BEKRÄFTA-läge är satt, måste kommandona ELIMINERA TABELL, RADERA och ÄNDRA bekräftas av operatören. För de två senare kommandona gäller detta endast om samtliga rader i tabellen skall tas bort eller uppdateras.
SL=SL	n	Anger antalet rader per sida i utskriftsfilen (SKRIVFIL). När SKRIV– kommandot exekveras, skrivs kolumnrubriker och sidnummer ut överst på varje ny sida. Om SL sätts lika med 0 (noll), skrivs all information ut i en följd, dvs utskriften delas inte in i sidor.
SB=SB	n	Anger antalet tecken per rad i utskriftsfilen.
SPRÅK=SPRÅK	språk	Ändrar kommandospråk. Önskat språk måste finnas definierat och lagrat i systemdatabanken SYSQL. Lagringen sker med hjälp av speciell programvara.
PLS=PLS	n	Anger hur många blankrader som ska läggas in mellan varje rad när innehållet i skrivfilen skrivs ut med SKRIV-kommandot.
HUVud	EJHuvud	Anger om huvudet i skrivfilen skall skrivas ut eller ej. Variabeln måste sättas innan skrivfilen öppnas första gången. Skrivfilen öppnas med något av kommandona HÄMTA/S/A, BESKRIV/S/A eller SKRIV.

15 Transaktioner och loggning

I detta avsnitt behandlas *transaktionshantering*, som är av vikt i fleranvändarsystem, samt *loggning*, som är en säkerhetsåtgärd om fel skulle uppstå och data förstöras.

15.1 Transaktionshantering

En *transaktion* är en följd av logiskt kopplade databasoperationer. Sådana operationer är läsa, skriva och radera. Från användarens synpunkt uppträder dock alla operationerna som en enda operation.

15.1.1 Transaktionsdatabank måste finnas

Databanken TRANSDB måste definieras för att transaktionshantering skall kunna användas. TRANSDB används av den speciella *transaktionshanteraren* i MIMER/DB. Det är databasadministratören (DBA) som skapar och underhåller TRANSDB.

15.1.2 Begränsningar vid transaktionshantering

Det är bara operationer, som arbetar på radnivå, som kan ingå i en transaktion. Följande operationer kan *inte* ingå i en transaktion:

- LOAD, funktion i MIMER/DB som laddar en tabell med rader
- DROP, funktion i MIMER/DB som tar bort samtliga rader i en tabell
- Ändringar av användarnas eller databankernas definitioner
- Ändringar av användarnas behörighet

Innan man utför någon av dessa operationer, bör man ta en kopia av databanken.

15.1.3 Transaktioner i MIMER/QL

Principen är att alla kommandon, som utför skrivoperationer på raderna i tabellen betraktas som en transaktion. Vissa kommandon som endast utför läsoperationer på databasen kan också inkluderas i en transaktion.

Följande kommandon betraktas av systemet som en transaktion:

- KOPIERA/LAGRA ... FRÅN
- RADERA ... DÄR ...
- LAGRA
- ÄNDRA

HÄMTA-kommandot betraktas normalt inte som en transaktion. Men detta förhållande kan ändras med SÄTT-kommandot.

HÄMTA-kommandot utför endast läsoperationer. Transaktionshantering väljs i detta fall med kommandot SÄTT TRANSAKTION. Standardvärdet för denna tillståndsvariabel är EJTRANSAKTION (kapitel 14).

15.2 Loggning

Med *loggning* menas att en kopia av alla uppdateringar (LAGRA, RADERA och ÄNDRA) av databasen, sparas i databanken LOGDB. Om ett hårdvarufel uppstår, tex fel på sekundärminnet, kan loggfilen användas tillsammans med en kopia av databasen för att återskapa databasen.

Endast uppdateringsoperationer som ingår i en transaktion loggas. För att loggningsfunktionen skall kunna användas måste alltså transaktionshanteringen vara påkopplad. Endast operationer som arbetar på radnivå kan loggas. Följande kommandon kan *inte* loggas:

- KOPIERA/LADDA ... FRÅN
- RADERA utan DÄR-sats
- DEFINIERA, OMDEFINIERA och ELIMINERA TABELL
- DBA

16 Fleranvändar- och enanvändarsystem

Det finns vissa begreppsmässiga skillnader mellan MIMER/QL i fleranvändar- och enanvändarsystem. Skillnaderna beskrivs i detta kapitel. Databanken SYSQL skall om möjligt åsättas generell behörighet "shared read-only", även när flera användare använder MIMER/QL i enanvändarsystem. SYSQL kommer i annat fall att uppta onödigt mycket utrymme på sekundärminnet.

16.1 MIMER/QL som fleranvändarsystem

I ett fleranvändarsystem delar användarna och applikationsprogrammen på gemensamma resurser, tex databankerna SYSQL, TRANSDB och LOGDB. Fleranvändarsystemet innehåller också funktioner för synkronisering av operationerna när flera användare använder dem.

De gemensamma resurserna har vidare kontroll över alla läs- och skrivoperationer i databankerna via SYSDB. I ett fleranvändarsystem kan databasadministratören (DBA) utgöras av en eller flera personer. DBA ansvarar för att informationen i SYSDB underhålls. Sett ur systemets synvinkel, kan man säga att DBA är ägare till samtliga databanker i databasen. DBA är behörig att tilldela andra användare behörighet till databankerna.

DBA är ansvarig för att användare och databanker definieras och att användarna tilldelas behörighet till databankerna. DBA skall också se till att de gemensamma databankerna definieras – TRANSDB, LOGDB och ARBDB. För att användarna skall kunna skapa temporära tabeller måste ARBDB åsättas generell behörighet X. DBA ansvarar dessutom för att databankerna kopieras med jämna mellanrum (kapitel 17).

16.2 MIMER/QL som enanvändarsystem

I ett enanvändarsystem har den ensamma användaren det ansvar som åvilar DBA i ett fleranvändarsystem. Användaren måste i detta fall av operativsystemet ges behörighet att använda databankerna (figur 16:1).

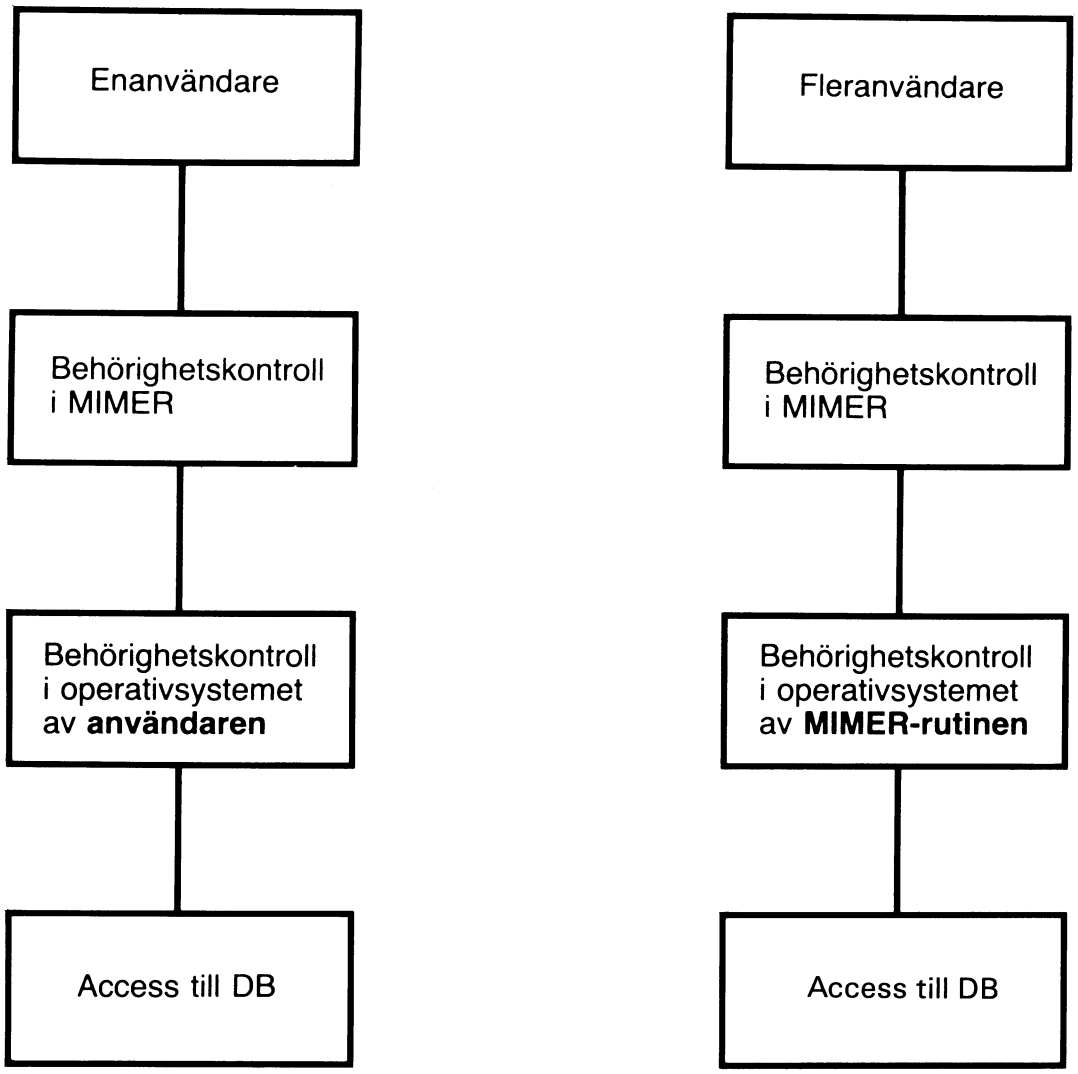


Fig 16:1. Att använda databasen i ett enanvändar- respektive fleranvändarsystem.

I ett enanvändarsystem måste användaren själv definiera SYSDB med ett hjälpprogram, SYSDBGEN, liksom de databanker som skall användas. Det fysiska namnet (filnamnet) på SYSDB måste också definieras i MIMER/QL. Detta sker med hjälp av en speciell rutin, EXITA1. Hur detta går till beskrivs i bilaga A.

Transaktionshantering och loggning kan även användas i ett enanvändarsystem. På samma sätt som i fleranvändarsystemet, måste databankerna TRANSDB respektive LOGDB definieras i SYSDB.

Enanvändarsystemet är ett äkta enanvändarsystem om användaren har X-befogenhet. Om användaren istället har S-befogenhet säger man att systemet är ett *skenbart fleranvändarsystem* (avsnitt 16.3).

16.3 Skenbart fleranvändarsystem

Det finns applikationer där flera användare endast hämtar data från databasen. För att man skall kunna hantera denna situation även i ett enanvändarsystem, finns det något som kallas "skenbart" fleranvändarsystem.

I detta fall kan flera användare med ställning S köra på ett enanvändarsystem med en gemensam SYSDB. När en användare med ställning S loggar in i systemet, öppnas SYSDB för delad läsning. Följande restriktioner gäller i detta fall:

- Användare med ställning S kan inte ändra sitt eget lösenord eller tilldela andra användare behörighet till de databanker som de senare har X-behörighet till (det är inte möjligt att utföra skrivoperationer i SYSDB).
- Transaktionshantering och loggning kan inte användas även om databankerna TRANSDB och LOGDB definierats.
- Alla databasoperationer är antingen endast för läsning eller exklusiva. Om flera användare skall använda samma databank, ska databanken öppnas för enbart läsning med kommandot ÖPPNA/R. Så snart en begäran om skrivoperationer anges (kommandot ÖPPNA/S), öppnas databanken för endast en användare, dvs ett felmeddelande erhålls om en annan användare vill använda databanken och den redan är öppen.
- Om en användare med ställning X försöker logga in i ett enanvändarsystem som redan används av flera andra användare med en gemensam SYSDB, kommer inloggningen att misslyckas, eftersom det inte går att öppna SYSDB för skrivoperationer, se ovan.
- Detsamma gäller för en användare med ställning S som försöker logga in i ett enanvändarsystem, som redan används av en användare med ställning X. Den senare har redan öppnat SYSDB för skrivoperationer, vilket medför att alla andra användare är utestängda.

16.4 Att överföra databanker mellan enanvändar- och fleranvändarsystem

Så länge databanken är öppen för användarna i ett fleranvändarsystem, kan den inte öppnas av en användare i ett enanvändarsystem och vice versa. Eftersom det inte är någon fysisk skillnad mellan databankerna i ett enanvändar- respektive fleranvändarsystem, finns det inga andra begränsningar för användningen av databankerna i respektive typ av system.

Om databanken ändras när den tillfälligt används i enanvändarsystem är det inte säkert att innehållet i LOGDB därefter stämmer med verkligheten, såvida inte samma LOGDB också finns i enanvändarsystemet. I detta fall bör man ta en kopia av databanken innan man använder den i fleranvändarsystemet.

17 Databasadministratörens uppgifter

De viktigaste uppgifterna för *databasadministratören* (DBA) är att skapa och underhålla databaserna, dvs att definiera och ta bort databanker och användare samt att ta säkerhetskopior av databankerna. Uppgifterna kan delas mellan flera användare, som alla har DBA-ställning.

Flera av DBAs uppgifter behandlas detaljerat i andra kapitel i denna handbok. Det här kapitlet innehåller därför endast en översiktlig summering.

17.1 Att skapa databaser

DBAs första uppgift är att skapa SYSDB med programmet SYSDBGEN (se bilaga F, början). I SYSDB definieras databasen och SYSDB innehåller vidare biblioteket över användare, databanker och användarbehörigheter.

Med SYSDBGEN kan också databankerna TRANSDB, LOGDB och SYSQL skapas. När väl SYSDB är skapad kan DBA definiera användarna, databankerna och användarbehörigheterna. Detta görs med hjälp av programmet DBADM eller med kommandot DBA. Hur detta går till beskrivs i kapitel 12.

I QL-miljön finns det tre databanker – SYSQL, PROCDB och ARBDB – som DBA är ansvarig för och som kräver några kommentarer:

- I SYSQL finns hjälptexter, felmeddelanden, systemprocedurer och systeminformation. SYSQL har den generella behörigheten R (läsning). DBA kan skapa ytterligare systemprocedurer eller systeminformation med hjälp av procedureditorn (avsnitt 17.3). För att DBA skall kunna göra detta, måste han tilldela sig själv S- eller X-behörighet till SYSQL.
- PROCDB är den databank, som utnyttjas för att lagra procedurbiblioteken. Den generella behörigheten till denna databank kan vara P, R, S eller X. Om P används kan inte användare utan personlig behörighet S komma åt innehållet i PROCDB. Varje användare som skall ges behörighet till PROCDB måste även tilldelas en personlig behörighet. Om R används kan procedurerna i PROCDB utföras men inte editeras. Om S används kan procedurerna även editeras. Om X används kan användaren även skapa och ta bort procedurbibliotek.

  DBA anger lämplig generell behörighet för PROCDB och tilldelar användarna personlig behörighet om inte den generella behörigheten är tillräcklig. DBA måste tilldela sig själv X-behörighet till PROCDB.

För de användare som skall skapa procedurer med procedureditorn måste DBA skapa ett procedurbibliotek. Namnet på detta bibliotek överensstämmer normalt med användarens namn i systemet (kapitel 14, variabeln PROCBIB). DBA skapar biblioteket om inte användaren har X-behörighet till PROCDB och kan skapa biblioteket själv. Det finns en systemprocedur för att skapa procedurbibliotek, DEFPRBIB.

- Databanken ARBDB används för temporär lagring av data. Generell behörighet för ARBDB är normalt X.

17.2 Att underhålla databaser

Med underhåll menas att definiera, definiera om och ta bort användare, databanker och användarbehörighet samt att kopiera och återställa databanker. Kopiering och återställning görs med hjälpprogrammet DBBRU. Övriga funktioner utförs med hjälpprogrammet DBADM eller kommandot DBA.

Databankerna TRANSDB och LOGDB som används för transaktionshantering respektive loggning definieras, definieras om och tas bort med kommandot DBA (kapitel 15). Omdefinieringen måste utföras i två steg:

- Först tas databanken bort.
- Därefter definieras databanken på nytt.

När man tar bort databanken innebär det inte att den försvinner fysiskt från skivan. Endast referensen i SYSDB, som refererar till filen på skivan, tas bort.

Omdefiniering används för att koppla ihop TRANSDB eller LOGDB med en annan fil på skivan. En ny LOGDB kan skapas vid omdefinieringen. Den gamla sparas på det yttre minnet.

17.3 Att lägga in systemberoende information

SYSQL innehåller hjälpinformation och felmeddelanden. I SYSQL finns dessutom information om den aktuella MIMER/QL-versionen och datorsystemberoende information. Den senare informationen presenteras på skärmen med systemproceduren INFO, som anropas med kommandot:

```
QL> Utför SYSPROC(INFO);
```

För utförlig beskrivning hänvisas till bilaga A.

Informationen kan ändras med hjälp av procedureditorn. Ny information kan läggas in med hjälp av proceduren INFO.

Man kan dessutom lägga in ett meddelande, som visas på skärmen när användarna loggar in i systemet. Detta meddelande läggs också in med hjälp av procedureditorn. Meddelandet lagras i SYSQL.

17.3.1 Att uppdatera befintlig information

All information som visas med proceduren INFO lagras i biblioteket SYSINFO i SYSQL. För att man skall kunna uppdatera informationen måste man ha minst S-behörighet till SYSQL. Proceduren som innehåller den systemberoende informationen heter QLAPPA. För att uppdatera denna procedur ger man kommandot:

```
QL> Editera SYSINFO(QLAPPA);
```

Ändringarna utförs därefter med editeringskommandona, som beskrivs i bilaga D.

Editeringen avslutas med kommandot SLUTA, varvid systemet övergår till kommandoläge (QL>). Genom att utföra proceduren INFO kan man kontrollera ändringarna.

Det meddelande som visas på skärmen när användarna loggar in i systemet skapas på samma sätt. För att lägga in ett nytt meddelande eller ändra eller ta bort ett gammalt ger man kommandot:

```
QL> Editera SYSINFO(*INFO);
```

Därefter används editeringskommandona för att göra erforderliga ändringar i meddelandet. Här följer ett exempel:

```
QL> Editera SYSINFO(*INFO);
ED> !
10  ====>     Detta är version 3.3 av MIMER/QL     <====
20
30  Skriv Utför SYSPROC(INFO); för mer information!
40  !
ED> Sluta
```

Ovanstående exempel visar hur man skriver ett nytt meddelande. Om man vill ta bort ett meddelande gör man på följande sätt:

```
QL> Editera SYSINFO(*INFO);
ED> Radera
ED> Sluta
```

Man kan kontrollera det nya meddelandet genom att gå tillbaka till kommandoläge, avsluta QL och därefter åter starta systemet.

17.3.2 Att skapa nya systemprocedurer

Att skapa en ny systemprocedur innebär att man skapar en procedur i SYSINFO, utökar proceduren INFOMENY och skapar en ny procedur i procedurbiblioteket SYSPROC.

Proceduren i SYSINFO innehåller informationen, som skall presenteras på skärmen när man utför proceduren INFO. Proceduren skapas på samma sätt som proceduren *INFO ovan. DBA väljer själv vilket namn proceduren skall ha. Den får dock inte ha samma namn som en redan befintlig procedur i SYSINFO. Editeringskommandot INVENTERA används för att ta reda på vilka procedurer som finns i SYSINFO.

När proceduren i SYSINFO är klar, måste man uppdatera procedurbiblioteket SYSPROC. Proceduren INFOMENY i SYSPROC innehåller ett antal +MENY-satser. +MENY används för att skapa procedurmenyer (avsnitt 13.11). Det finns en +MENY-sats för varje alternativ i menyn. +MENY– satsen innehåller en ledtext, som skall visas i menyn, och namnet på den procedur, som skall exekveras. Om man skall lägga till en eller flera procedurer, måste man alltså skapa en eller flera nya +MENY-satser i proceduren INFOMENY, se exemplet nedan.

Man måste också skapa en ny procedur i SYSPROC, alltså den procedur som skall utföras när det nya alternativet väljs på menyn. Man kan välja vilket procedurnamn som helst utom ett som redan finns i SYSPROC. Den nya proceduren består bara av två satser (se exemplet nedan), nämligen en +TILLDELA-sats och kommandot UTFÖR. +TILLDELA-satsen används för att tilldela en

procedurvariabel det nya namnet. UTFÖR-kommandot medför att en procedur i SYSPROC utförs, som läser och presenterar raderna i proceduren, vilken finns i biblioteket SYSINFO. Värdet på procedurvariabeln bestämmer vilken procedur i SYSINFO som skall användas.

Det här exemplet visar hur man skapar en ny systemprocedur i proceduren SYSPROC(INFO). Först måste man lägga in den information som ska visas på skärmen:

```
QL> Editera SYSINFO(BACKUP);
ED> !
10                    Logg för databankskopiering
20                  ==============================
30
32              Denna information loggar datum när
34              kopiering görs och vem som utför
35              kopieringen.
36
40              Databank        Datum          Utförd av
50              ____            _____          _____
60              PROCDB          1984-OKT-05    Sven
70              DATAB1          1984-OKT-05    Sven
80              DATAB2          1984-OKT-07    Gunnar
ED> Sluta
```

Därefter skall man utöka menyn i proceduren INFOMENY.

```
QL> Editera SYSPROC(INFOMENY);
ED> Lista
12    +Meny PRESREL 'Skillnader mellan version 3.2 och 3.3';
20    +Meny QLAPPA 'Datorsystemberoende info (bilaga A)';
30    +Meny STOP Sluta (återvänd till QL-kommandoläge eller föreg. meny;
ED>   !
40    Meny DBBACK 'Logg för databankskopiering';
50    !
```

Slutligen måste man skapa den nya SYSPROC-proceduren, som definierar vilken procedur i SYSINFO som skall läsas och listas.

```
ED> Aktuell DBBACK;

Aktuell procedur: SYSPROC(DBBACK) i SYSQL
ED>  !
10    +Tilldela &KW='BACKUP';
20    Utför SYSPROC(HÄMTINFO);
30    !!
ED> Sluta
```

Nu är det klart och man kan kontrollera funktionen genom att ge kommandot:

```
QL> Utför SYSPROC(INFO);
```

Installationsberoende information

Värden på vissa hårdvaru- och mjukvaruberoende parametrar är installationsberoende, dvs beroende av det datorsystem på vilket man kör MIMER/QL. Information om dessa samt om hur man använder MIMER/QL på det aktuella datorsystemet finns i handböckerna MIMER Installation Guide och MIMER User's Guide hos din MIMER-ansvariga eller hos databasadministratören.

Den installationsberoende informationen kan också presenteras på skärmen med kommandot:

```
QL> Utför SYSPROC(INFO);
```

När kommandot ges visas en meny på skärmen. Väljer du alternativet installationsberoende information kan du få informationen lagrad i skrivfilen. Innehållet i filen kan därefter skrivas ut på skrivaren. Utskriften bör arkiveras efter det här bladet.

Vi ger ett exempel :

```
QL> Utför SYSPROC(INFO);

Välj något av följande alternativ:

 1 Skillnader mellan version 3.2 och 3.3
 2 Skillnader mellan utgåva 3.3.1 och 3.3.2
 3 Installationsberoende QL-information (bilaga A)
 4 Sluta (återvänd till QL-kommandoläge eller föregående meny)
```

Med alternativ 1 visas skillnaderna mellan programversionen 3.3 och den föregående, 3.2.

Med alternativ 2 visas skillnaderna mellan utgåva 3.3.2 och den föregående utgåvan, 3.3.1. Givetvis finns det ingen information här förrän en utgåva 3.3.2 faktiskt föreligger.

Nya *versioner* innehåller större ändringar i programmets funktion och nya funktioner inom ramen för den aktuella programgenerationen. Nya *utgåvor* innehåller ändringar inom ramen för den aktuella programversionen. I den treställiga beteckningen anger första siffran generation, andra version och tredje utgåva.

Med alternativ 3 visas installationsberoende information.

Med alternativ 4 avslutas SYSPROC och återgång sker till kommandoläge i QL eller till föregående meny.

Kommandon och variabler i MIMER/QL

Denna bilaga består av enkla förteckningar över kommandon och variabler. Engelska motsvarigheter anges.

Detaljerade beskrivningar ges på bl a följande ställen i handboken:

Bilagans innehåll	*Detaljerad information*
A. QL-kommandon i alfabetisk ordning	Bilaga C
B. QL-kommandon grupperade i funktionsgrupper	Kapitel 6
C. Procedurkommandon	Kapitel 13
D. Kommandon i procedureditorn	Bilaga D
E. Lägsta behörighet till en databank vid olika QL-kommandon	Avsnitt 2.4.1–3
F. Systemvariabler	Avsnitt 13.15
G. Tillståndsvariabler	Kapitel 14

A. QL-kommandon i alfabetisk ordning

Svenska	*Engelska*
ALIAS	ALIAS
BESKRIV ALIAS	DESCRIBE ALIAS
BESKRIV DATABANK	DESCRIBE DATABANK
BESKRIV MILJÖ	DESCRIBE AREA
BESKRIV SÄTTNINGAR	DESCRIBE SETTINGS
BESKRIV TABELL	DESCRIBE TABLE
DBA	DBA
DEFINIERA INDEX	DEFINE INDEX
DEFINIERA TABELL	DEFINE TABLE
EDITERA	EDIT
ELIMINERA INDEX	REMOVE INDEX
ELIMINERA TABELL	REMOVE TABLE
GRAFIK	GRAPHICS
HJÄLP	HELP
HÄMTA	GET
KOMMANDO	COMMAND
KOPIERA ... FRÅN	COPY ... FROM
KOPIERA ... TILL	COPY ... TO
LAGRA	INSERT
OMDEFINIERA	REDEFINE
ADDERA	APPEND
RADERA	DELETE

RADERA	DELETE
SKRIV	PRINT
SLUTA	EXIT
STATISTIK	STATISTICS
STÄNG	EXCLUDE
SÄTT	SET
UTFÖR	EXECUTE
VISA	SHOW
ÅTERSTÄLL	RESET
ÄNDRA	UPDATE
ÖPPNA	INCLUDE

B. QL-kommandon grupperade i funktionsgrupper

B1. Definiering eller beskrivning av databanker

Svenska	*Engelska*
BESKRIV DATABANK	DESCRIBE DATABANK
DBA	DBA

B2. Definiering eller beskrivning av data

Svenska	*Engelska*
BESKRIV TABELL	DESCRIBE TABLE
DEFINIERA INDEX	DEFINE INDEX
DEFINIERA TABELL	DEFINE TABLE
ELIMINERA INDEX	REMOVE INDEX
ELIMINERA TABELL	REMOVE TABLE
OMDEFINIERA	REDEFINE

B3. Sökning eller manipulering av data

Svenska	*Engelska*
HÄMTA	GET
KOPIERA ... FRÅN	COPY ... FROM
KOPIERA ... TILL	COPY ... TO
LAGRA	INSERT
RADERA	DELETE
SKRIV	PRINT
ÄNDRA	UPDATE

B4. Definiering eller beskrivning av förutsättningar för ett arbetspass i QL

Svenska	*Engelska*
ALIAS	ALIAS
BESKRIV ALIAS	DESCRIBE ALIAS
BESKRIV MILJÖ	DESCRIBE AREA
BESKRIV SÄTTNINGAR	DESCRIBE SETTINGS
STÄNG	EXCLUDE
SÄTT	SET
VISA	SHOW
ÅTERSTÄLL	RESET
ÖPPNA	INCLUDE

B5. Andra funktioner

Svenska	*Engelska*
EDITERA	EDIT
GRAFIK	GRAPHICS
HJÄLP	HELP
KOMMANDO	COMMAND
SLUTA	EXIT
STATISTIK	STATISTICS
UTFÖR	EXECUTE

C. Procedurkommandon

Svenska	*Engelska*
+ AVSLUTA	+ CLOSE
+ FRÅGA	+ PROMPT
+ HOPPA	+ GOTO
+ INLED	+ OPEN
+ KOMMENTAR	+ COMMENT
+ MENY	+ CASE
+ NÄSTA	+ NEXT
+ OM...SÅ	+ IF...THEN
+ SKRIV	+ WRITE
+ STOPP	+ STOP
+ TILLDELA	+ LET

D. Kommandon i procedureditorn

Svenska	*Engelska*
DÖP-OM	RENAME
ELIMINERA	REMOVE
ERSÄTT	REPLACE
FLYTTA	MOVE
HJÄLP	HELP
INVENTERA	DIRECTORY
KOPIERA	COPY
KORRIGERA	CORRECT
LAGRA	INSERT (or ADD)
LISTA	LIST
NUVARANDE	CURRENT
OMNUMRERA	SEQUENCE
ORDNA	FIX
RADERA	DELETE
SLUTA	EXIT
SPARA	SAVE
SUDDA	RUB
ÄNDRA	CHANGE (or SUBSTITUTE)

E. Lägsta behörighet till en databank vid olika QL-kommandon

Behörighetsgrader

R = läsbehörighet (även L godtas av systemet)
S = skriv- och läsbehörighet
X = skriv och läsbehörighet samt behörighet att ta bort och lägga till tabeller

Kommando	*Behörighet*
BESKRIV DATABANK	R
BESKRIV TABELL	R
DEFINIERA INDEX	X
DEFINIERA TABELL	X
EDITERA	S
ELIMINERA INDEX	X
ELIMINERA TABELL	X
HÄMTA	R
HÄMTA (arbetstabell)	X till ARBDB, till övriga R
KOPIERA . . . FRÅN	S
KOPIERA/LADDA . . . FRÅN	X
KOPIERA . . . TILL	R
LÄGG-TILL	S
OMDEFINIERA	X
RADERA (rader i tabell)	S
RADERA (hel tabell)	X
SKRIV	R
UTFÖR	R
ÄNDRA	S
ÖPPNA/b	b = antingen R, S eller X

F. Systemvariabler

Svenska	*Engelska*
*ANTAL	*COUNT
*DATUM	*DATE
*FEL	*ERROR
*VEM	*USER

G. Tillståndsvariabler

Standardvärde anges i förekommande fall först.

Svenska	*Engelska*	*Avser*
AEXITC1	NEXITC1	Antal felaktiga försök innan systemet bryter ett arbetspass
ARBDB	WORKDB	Namn på arbetsdatabank
BEKRÄFTA | EJBEKRÄFTA	VERIFY | NOVERIFY	Bekräftelse av begärd operation
BIBLIOTEK	PROCLIB	Standardnamn på procedurbibliotek
EJEKO | EKO	NOECHO | ECHO	Eko eller ej
EJHUVUD | SHUVUD	NOPHEAD | PHEAD	Inledande text i skrivfil
EJSPÅRA | SPÅRA	NOTRACE | TRACE	Listning av QL-kommandon i en procedur

EJTRANSAKTION | TRANSAKTION	NOTRANSACTION | TRANSACTION	Transaktionshantering eller eller ej
EJUTSKRIFT | UTSKRIFT	NOGAG | GAG	Undertryckande av utskrift
FLÄNGDm	FLEN	Antal tecken reserverade för flyttalsvärden
ILÄNGDm	ILEN	Antal tecken reserverade för heltalsvärden
LÅNG | KORT	LONG | SHORT	Längd på hjälptexter
MI	CC	Meddelandeintervall
PROCDB	PROCDB	Namn på procedurdata-banken
RB	LW	Radbredd (terminalsida)
RR	LC	Rader per terminalsida
SB	PW	Radbredd (skrivsida)
SKRIVFIL	PRINTFILE	Namn på skrivfilen
SL	PL	Rader per skrivsida
SPRÅK	LANGUAGE	Val av språk (kommandon, skärmtexter)
SRA	PLS	Radmatning i skrivfil

Kommandon i MIMER/QL

I denna bilaga ges detaljerade beskrivningar av samtliga QL-kommandon i alfabetisk ordning. Beskrivningarna innehåller:

- syntax
- funktions- och parameterbeskrivning
- exempel

Kommandoorden skrivs i kortform. Kortformen av ett kommandoord är alltid de tre första bokstäverna.

ALIAS

Syntax

ALIas tabellnamn(alias,...)<,...>;

Funktion

Definierar ett eller flera alias (oftast förkortningar) för ett tabellnamn. I vissa kommandon kan sedan alias användas istället för tabellnamnet. Ett alias gäller endast under den körning då det har definierats.

Alias används av systemet för att upprätthålla en specifik position i den aktuella tabellen. Om flera alias definieras för en tabell, kan systemet underhålla flera sinsemellan oberoende positioner. Om flera alias definieras för en tabell, kan det betraktas som att systemet tillhandahåller en kopia av tabellen för varje alias.

Ett alias får bestå av högst 2 alfanumeriska tecken.

Om samma alias anges i olika ALIAS-kommandon, gäller den senaste definitionen. En tidigare definition av samma alias tas alltså bort.

Exempel

```
QL> Alias PERSONAL(P1,P2);
QL> Alias ANSTÄLLD (AN),HOTELL (H);
```

BESKRIV ALIAS

Syntax

BESkriv ALIas <tabellnamn>;

Funktion

Med kommandot visas på skärmen en lista över samtliga alias som har definierats under det aktuella arbetspasset. Om tabellnamn specificeras listas enbart alias för den angivna tabellen. I annat fall listas samtliga alias tillsammans med tillhörande tabellnamn.

Istället för BESKRIV ALIAS kan man använda kommandot VISA ALIAS, som ger samma resultat.

Exempel

QL> **Beskriv alias**;	Samtliga alias listas.
QL> **Beskriv alias** PERSONAL;	Samtliga alias för tabellen PERSONAL listas.

BESKRIV DATABANK

Syntax

BESkriv </TI/SI/A> **DATabank** databanksnamn;

Funktion, parametrar

Samtliga tabeller som finns i databanken (databanksnamn) visas på skärmen. Dessutom visas fysiskt filnamn, generell behörighet, inloggad användares behörighet samt reserverat och utnyttjat utrymme.

Exempel

QL> **Beskriv databank** ANSTDB;	Samtliga tabeller som finns i databanken ANSTDB presenteras på skärmen.
QL> **Beskriv databank** HOTELLDB;	Samtliga tabeller som finns i databanken HOTELLDB presenteras på skärmen.

BESKRIV MILJÖ

Syntax

BESkriv MILjö <'>databanksnamn<'>;

Funktion

Om databankens namn utelämnas listas samtliga databanker och tabeller som har öppnats med ÖPPNA-kommandot under det aktuella arbetspasset. Om databanksnamn anges listas bara de tabeller som har öppnats i den angivna databanken. Dessutom visas med vilken behörighet tabellerna har öppnats (R, S eller X).

Istället för BESKRIV MILJÖ kan man använda kommandot VISA ÖPPNADE, som ger samma resultat.

Exempel

QL> **Beskriv miljö**;	Lista över samtliga databanker och tabeller som har öppnats under arbetspasset visas på skärmen.
QL> **Beskriv miljö** HOTELLDB;	Tabeller som har öppnats i databanken HOTELLDB visas på skärmen.

BESKRIV SÄTTNINGAR

Syntax

BESkriv SÄTtningar;

Funktion

Tillståndsvariablernas aktuella värden presenteras på skärmen.

Istället för BESKRIV SÄTTNINGAR kan man använda kommandot VISA SÄTTNINGAR som ger samma resultat.

Exempel

QL> **Beskriv sättningar**;

BESKRIV TABELL

Syntax

BESkriv </**TI**/**SI**/**A**></**KORtIFULIständig**> **TABell** tabellnamn;

Funktion

Den angivna tabellens definition visas på skärmen. Kolumnnamnen och kolumnernas format presenteras. Dessutom visas med en asterisk vilka kolumner, som är indexerade.

Om parametern FULLSTÄNDIG används presenteras också radlängd och antal lagrade rader.

Exempel

QL> **Beskriv tabell** RUM;	Beskrivning av tabellen RUM skrivs ut på skärmen.
QL> **Beskriv/A tabell** RUM;	Beskrivning av tabellen RUM skrivs ut på skärm och skrivare.
QL> **Beskriv/A/Fullständig tabell** RUM;	Samma utskrift som i föregående med tillägg av uppgift om radlängd och antal lagrade rader.

DBA

Syntax

DBA;

Funktion

Kommandot används för att definiera och ta bort databanker, användare och behörigheter. Dessutom kan det användas för presentation av vilka databanker, användare och behörigheter som finns definierade i systemet.

DBA-funktionerna kan väljas på en meny, som visas på skärmen. Därefter frågar systemet efter nödvändiga uppgifter, genom att visa ledtexter. För att kunna skapa nya databanker och lägga in nya användare måste man ha ställning X (DBA). Användare med ställning S kan ändra sitt eget lösenord och använda VISA-kommandot för att ta del av systeminformation. En användare

med ställning S kan tilldela andra användare behörighet till de databanker som han själv har X-behörighet till.

Kapitel 12 innehåller exempel på DBA-kommandots användning.

DEFINIERA INDEX

Syntax

DEFiniera INDex tabellnamn(kolumnnamn,...);

Funktion

Kommandot skapar ett sekundärindex (sekundärnyckel) för de angivna kolumnerna i den angivna tabellen. Sekundärnyckeln används automatiskt vid uppdatering och sökning. Operationerna utförs snabbare och man får på så sätt svar snabbare. För att man skall kunna använda kommandot måste man ha X-behörighet till den databank där tabellen är lagrad.

Exempel

QL> **Definiera index** RUM(TYP);

När HÄMTA-kommandot används för att hämta data, presenteras resultatet i en annan sorteringsordning än om inget sekundärindex finns. Om inget sekundärindex finns, listas data i den ordning som bestäms av primärnyckeln. Finns det däremot en indexerad kolumn, listas data i den ordning som bestäms av sekundärnyckeln, om det i DÄR-satsen ställs villkor på den indexerade kolumnen.

Avsnitt 11.4 innehåller fler exempel.

DEFINIERA TABELL

Syntax

DEFiniera TABell tabellnamn (kolumnnamn **ÄR** format
<:kolumnnamn **ÄR** format>
<,kolumnnamn **ÄR** format>)
I databanksnamn;

Funktion

Kommandot används för att skapa en ny tabell (tabellnamn) i en databank (databanksnamn). Tabellen innehåller ett antal kolumner (kolumnnamn), specificerade i en kolumnlista. Kolumnnamnen får bestå av bokstäverna A-Ö och siffrorna 0–9 och skall alltid börja med en bokstav. En tabell kan bestå av högst 99 kolumner. Tabellnamnen måste vara unika, dvs det får endast finnas en tabell med ett visst namn i en databank. Värdet lagras i kolumnen enligt det angivna formatet. Formatet anges enligt följande:

Tn

där

T = lagringstyp (C – alfanumeriska data, I –- heltal eller F -flyttal)
n = lagringsutrymme

Beroende på lagringstyp och det datorsystem som används, varierar talområdet för n enligt följande:

Lagringstyp C

n = 1-min(MMRL,MMRL-summan(kolumnlängderna))

Summan(kolumnlängderna) = Summan av de redan definierade kolumnlängderna.

Lagringstyp I

n = 1-ILÄNGDB.

Lagringstyp F

n = 3-FLÄNGDB.

Aktuella värden för variablerna MMRL, ILÄNGDB och FLÄNGDB presenteras på skärmen med kommandot KÖR SYSPROC(INFO) (bilaga A). Den maximala radlängden (summan av de individuella kolumnernas längder) är begränsad till MMRL.

Den eller de kolumner, som utgör primärnyckeln, måste anges först i kolumnlistan. Ett kolon (:) används för att markera vilka kolumner som ingår i primärnyckeln. Detta kolon ersätter kommatecken (,) i kolumnlistan efter den sista primärnyckelkolumnen och före den första icke-primärnyckelkolumnen i listan. Om kolon inte skrivs kommer samtliga kolumner i listan att ingå i primärnyckel.

För att man skall kunna skapa en ny tabell måste man ha X-behörighet till den aktuella databanken.

Exempel

```
QL> Definiera tabell PRIS(PRISKOD är C4:
+>                        PRIS är I2)
+>  i HOTELLDB;
```

Avsnitt 11.1 innehåller fler exempel.

EDITERA

Syntax

EDItera <procedurbibliotek><(procedur)>;

Funktion

Kommandot startar procedureditorn som används för att skapa och ändra procedurer.

Procedurerna lagras i den databank, som specificeras i tillståndsvariabeln PROCDB (kapitel 14). Om procedurbibliotek utelämnas används det som specificeras av tillståndsvariabeln BIBLIOTEK. Standardvärden för dessa tillståndsvariabler är PROCDB respektive användarens namn. Om procedur utelämnas, används TEMP. Detta procedurnamn är reserverat genom att man inte kan skapa procedurer, som inleds med tecknen TEMP samt består av ytterligare tecken. TEMP1 och TEMPER kan alltså inte användas medan TEMP och TEM accepteras. Det finns två andra namn, som är reserverade på motsvarande sätt: SPAR och SIST.

För att man skall kunna använda editorn krävs minst S-behörighet till PROCDB. För att skapa procedurbibliotek, krävs X-behörighet.

När man ger EDITERA-kommandot startas editorn och man befinner sig i editeringsläge: symbolen ED> visas på skärmen. Följande editeringskommandon kan nu användas:

- LAGRA, KORRIGERA, ORDNA, SUDDA och ÄNDRA för operationer på strängar inom en rad.
- LAGRA, ERSÄTT och RADERA för operationer på en hel rad.
- KOPIERA och FLYTTA för operationer på flera rader.
- NUVARANDE, ELIMINERA, DÖP-OM, SPARA och OMNUMRERA för operationer på hela proceduren.
- INVENTERA för operationer på biblioteket.
- HJÄLP eller ? för att visa hjälpinformation.

Om man skriver ett utropstecken (!) när ED> visas på skärmen, övergår editorn till inmatningsläge. I inmatningsläge frågar systemet efter en ny rad genom att radens nummer visas på skärmen. Om man skriver två utropstecken (!!) eller ett utropstecken följt av SLUTA (!SLUTA) övergår systemet till kommandoläge: kommandosymbolen QL> visas på skärmen.

Om man skriver ett utropstecken eller ett utropstecken direkt följt av ett uppdateringskommando, övergår systemet till editeringsläge.

Exempel

Se resp kommando i bilaga D.

ELIMINERA INDEX

Syntax

ELIminera INDex tabellnamn(kolumnnamn,...);

Funktion

Kommandot tar bort det index som finns för kolumnen kolumnnamn i tabellen tabellnamn. För att kunna använda kommandot måste man ha X-behörighet till den aktuella databanken.

Exempel

QL> **Eliminera index** PERSONAL(ANSTNR);	Index för kolumnen ANSTNR i tabellen PERSONAL tas bort.

ELIMINERA TABELL

Syntax

ELIminera TABell tabellnamn;

Funktion

Kommandot medför att alla data och definitionen av tabellen tas bort. Man måste ha X-behörighet till den aktuella databanken.

Exempel

QL> **Eliminera tabell** NÖJEN;	Tabellen NÖJEN tas bort. Samtliga data i tabellen samt tabelldefinitionen försvinner.

GRAFIK

Syntax

GRAfik;

Funktion

Medför att körningen lämnar MIMER/QL och går över till grafiksystemet MIMER/GR. I MIMER/GR kan kurvor, stapeldiagram och cirkeldiagram framställas i färg.

HJÄLP

Syntax

HJÄlp <kommando I felkod>;

Funktion

Kommandot visar syntaxen för det angivna kommandot eller en beskrivning av den angivna felkoden.

Om man bara anger HJÄLP visas en förteckning över samtliga möjliga kommandon. Om man vill ha syntaxen för ett speciellt kommando eller en grupp av kommandon, anges kommandot eller begynnelsebokstäverna för kommandogruppen.

Om man anger felkod visas en förklaring till koden på skärmen. Ett felaktigt kommando resulterar i ett felmeddelande och en felkod. I de fall felmeddelandet inte klargör felorsaken bör hjälptexten göra det.

Exempel

Om man exempelvis vill veta syntaxen för kommandot DEFINIERA TABELL skriver man:

```
QL> Hjälp Definiera tabell;
```

Om man vill veta syntaxen för alla DEFINIERA-kommandon skriver man:

```
QL> Hjälp Definiera;
```

Om man vill veta syntaxen för alla kommandon, som börjar på D skriver man:

```
QL> Hjälp D;
```

Om man vill ha en utförligare förklaring till en speciell felkod skriver man tex så här:

```
QL> Hjälp -1242;
QL> Hjälp 69;
```

Det första exemplet är en felkod i MIMER/DB, där alla felkoder inleds med ett minustecken (-).

HÄMTA

Syntax

HÄMta </TI/SI/A> kolumnlista <**DÄR** villkorssats>;

eller

HÄMta tabellnamn(kolumnlista) <**DÄR** villkorssats>;

Funktion

HÄMTA-kommandot hämtar data från en eller flera tabeller. Med data menas värden för de kolumner som anges i kolumnlistan. Genom att ange en villkorssats kan man begränsa antalet rader från vilka data skall hämtas.

Kolumnlistan kan betraktas som en tabell, vilken innehåller de angivna kolumnerna. Om tabellnamn anges laddas denna tabell med de data som HÄMTA-kommandot hämtar. Tabellen från vilka data hämtas måste givetvis vara definierad i systemet. Om inte tabellnamn anges visas raderna på skärmen.

Om den tabell som anges i HÄMTA-kommandot inte är knuten till någon speciell databank skapas tabellen i arbetsdatabanken, ARBDB. Denna tabell kan därefter användas som vilken annan tabell som helst, dvs man kan hämta, uppdatera och ta bort data i tabellen. När tabellen senare används som arbetstabell i ett HÄMTA-kommando, kommer den att ersättas av den nya tabell som HÄMTA-kommandot skapar. Detta händer inte om arbetstabellen finns i en databank som inte är en arbetsdatabank. I det senare fallet uppdaterar HÄMTA-kommandot tabellen, dvs lägger in raderna, förutsatt att de kolumnnamn som specificeras i kolumnlistan överensstämmer med kolumnerna i tabellen. Om arbetstabellen definieras i ARBDB får tabellens namn inte bestå av mer än 3 tecken.

Om tabellen skall ligga i ARBDB måste arbetstabellen definieras innan HÄMTA-kommandot används första gången. Om fleranvändarsystem används, delar alla användare på ARBDB.

Kolumnlistan består av en eller flera "tabref.kolumnnamn" åtskilda med komma (,) eller kolon (:). Kolumnerna som föregår kolon definierar primärnyckeln för den tabell, som HÄMTA-kommandot eventuellt skapar. Tecknet * (asterisk) kan ersätta kolumnnamnen. HÄMTA-kommandot ersätter * med de kolumner som ingår i den tabell till vilken refereras med alias eller tabellnamn.

Användaren måste specificera kopplingen mellan tabellerna i villkorssatsen (kapitel 8).

Exempel

För att förkorta kommandona har lätt uttydbara alias använts för tabellerna.

QL> **Hämta** P1.*; — Listar samtliga rader i tabellen PERSONAL (P1 är alias) på skärmen.

QL> **Hämta** H1.HNAMN **där** H1.POSTNR = '75147';	Listar alla hotell som har postnummer 75147 och presenterar dem på skärmen.
QL> **Hämta** H1.HNAMN,R1.RUMSNR **där** H1.HID = R1.HID;	Listar samtliga rumsnummer på samtliga hotell.
QL> **Hämta** H1.HNAMN,A1.ANAMN **där** A1.ANSTNR = P1.ANSTNR **och** P1.HID = H1.HID;	Hämtar namnen på alla hotell samt namnen på deras anställda. Tabellen PERSONAL (P1) innehåller kopplingen mellan tabellerna HOTELL (H1) och ANSTÄLLDA (A1).
QL> **Hämta** AT1(H1.HNAMN, R1.TYP,P2.PRIS) **där** H1.HNAMN **in** 'UPP' **och** R1.PRISKOD = P2.PRISKOD;	Listar alla hotellnamn, rumstyper och rumspriser för de hotell vilkas namn börjar med 'UPP'. För ändamålet skapas en arbetstabell med kolumnerna HNAMN,RUM och PRIS (AT1). Tabellen RUM (R1) används som kopplingstabell mellan tabellerna HOTELL (H1) och PRIS (P2).

KOMMANDO

Syntax

KOMmando 'stränguttryck';

Funktion

Funktionen hos KOMMANDO är systemberoende. Kommandot överför 'stränguttryck' till en definierad anpassningsrutin. Med hjälp av denna rutin kan man lägga in skräddarsydda kommandon speciellt anpassade för en viss tillämpning. Kommandot gör det möjligt att utnyttja funktioner som är tillgängliga endast på vissa datorsystem. KOMMANDO medger alltså tillägg av funktioner som inte finns i MIMER/QL.

PUSH-funktionen i DEC-20/TOPS-20 eller SPAWN-kommandot i VAX/VMS kan tjäna som exempel. Dessa avbryter utförandet av ett program eller kommando och aktiverar ett nytt program/kommando. När det senare är utfört, fortsätter utförandet av det avbrutna programmet/kommandot. Det kan vara önskvärt att nå denna funktion från MIMER/QL. På detta sätt kan körningen av MIMER/QL tillfälligt avbrytas och andra program komma in emellan. Därefter kan man återuppta arbetet med MIMER/QL utan att behöva göra om alla initieringar, dvs öppna databanker/-tabeller etc.

Exempel

```
QL> Kommando 'EDIT FIL.DAT';      Avbryt körningen och starta editorn.
<Editering av filen FIL.DAT>
      .
      .
      .
      * Sluta%                    Avsluta editeringen.
QL>
```

Den exakta användningen av KOMMANDO finns beskriven i den installationsberoende informationen (bilaga A).

KOPIERA ... FRÅN

Syntax

KOPiera</LÄGg-till\|/LADda\|</DUPlikatlogg\|/EJDuplikatlogg>
tabref **FRÅn** <'>filnamn<'> <(<nX,>kolumnnamn **ÄR** format,...)>;

Funktion

Kommandot läser data från en sekvensiell fil och lagrar det i en tabell. Postspecifikationen – kolumnnamn är format, ... – anger formatet på posten i den sekvensiella filen. Om postspecifikationen utelämnas, förutsätts att posten i den sekvensiella filen har exakt samma utseende som en rad i tabellen vad gäller format och ordning. Med parametern nX anges att n stycken tecken skall hoppas över i den sekvensiella posten.

Med parametrarna LÄGG-TILL och LADDA anger man hur data skall sorteras in i tabellen.

Vid LÄGG-TILL läggs de nya raderna in i tabellen utan hänsyn till om det redan finns rader i tabellen eller inte. Dubletter som finns i den sekvensiella filen lagras inte. Dubletter kan loggas i filen SKRIVFIL.

Vid LADDA kommer redan befintliga rader i tabellen att hämtas ut och läggas till de rader, som kopieras från den sekvensiella filen och därefter lagras samtliga rader i den "tomma" tabellen. Om antalet rader i den sekvensiella filen är mycket större än antalet rader i tabellen, tex om tabellen är tom, är det bättre att använda LADDA. Om det finns dubletter, dvs rader där innehållet i primärnyckeln är lika, kan man vid LADDA inte styra vilken rad som skall lagras i tabellen. Man kan inte heller logga de dubletter som inte lagras i tabellen.

LÄGG-TILL är standardvärde och gäller om inget annat anges.

För att kunna använda KOPIERA ... FRÅN måste användaren ha följande behörighet:

X om LADDA skall användas.
S om LÄGG-TILL skall användas.

Loggning av dubletter i skrivfilen styrs med parametrarna DUPLIKATLOGG (loggning sker) och EJDUPLIKATLOGG (ingen loggning sker). DUPLIKATLOGG är standardvärde och gäller om inget annat anges.

Posternas maximala längd i en sekvensiell fil (variabeln MSRL) visas på skärmen med kommandot KÖR SYSPROC(INFO), se Bilaga A.

Exempel

```
QL> Kopiera/Lagra/Ejduplikatlogg HOTELL
    från HOTFIL (HNAMN är C20,1X, TELE är C15);
```

Data kopieras från den sekvensiella filen HOTFIL till tabellen HOTELL. Ett mellanslag läggs in mellan kolumnerna HNAMN och TELE. Raderna läggs in i tabellen oavsett om det finns data i tabellen eller inte. Dubletter loggas inte.

```
QL> Kopiera/Ladda ANSTÄLLD från 'ANSTFIL';
```

Innehållet i den sekvensiella filen ANSTFIL läggs in i tabellen ANSTÄLLD. LADDA medför att de rader som redan finns i tabellen hämtas ut och läggs till de rader som kopieras från den sekvensiella filen.

Kapitel 9 innehåller fler exempel.

KOPIERA ... TILL

Syntax

KOPiera tabref **TILl** <'>filnamn<'> <(<nX,>kolumnnamn **ÄR** format,...)>;

Funktion

Kommandot läser rader från en tabell och lagrar dessa i en sekvensiell fil. Postspecifikationen – kolumnnamn är format,... – anger formatet för posten i den sekvensiella filen. Om postspecifikationen utelämnas, antas att den sekvensiella posten har samma utseende som en rad i tabellen vad avser format och ordning. Med parametern nX anges att n stycken tecken skall hoppas över i posten. Det överhoppade fältet fylls med blanktecken.

Postens maximala längd (variabeln MSRL) visas på skärmen med kommandot KÖR SYSPROC(INFO) (bilaga A).

Exempel

```
QL> Kopiera NÖJEN till 'NÖJFIL';
```

Den sekvensiella filen NÖJFIL har samma format som tabellen NÖJEN.

```
QL> Kopiera PERSONAL till 'PERSFIL' (3X, ANSTNR är C5);
```

Kolumnen HID kopieras inte till PERSFIL.

Kapitel 9 innehåller fler exempel.

LAGRA

Syntax

LAGra tabref; (1)

eller

LAGra tabref (kolumnnamn='data',...); (2)

Funktion

LAGRA-kommandot kan användas på två sätt. I fall (1) frågar systemet efter de värden som skall läggas in i tabellen. Man behöver inte känna till vilka kolumner som finns i tabellen, eftersom kolumnnamnen visas på skärmen. I fall (2) måste man specificera vilka kolumner som man vill tilldela data.

I båda fallen skapas en ny rad i tabellen. Raden sorteras in på rätt plats med hjälp av värdet i primärnyckelkolumnerna (se kommandot DEFINIERA TABELL). Om det redan finns en rad i tabellen med samma värde i primärnyckelkolumnerna, accepteras inte den nya raden.

För att man skall kunna använda LAGRA-kommandot måste man ha minst S-behöriget till den aktuella databanken.

Alternativ 1: Systemet frågar efter data

Kolumnernas namn visas på skärmen och systemet förväntar sig att operatören anger data följt av vagnretur (RETURN eller CR).

Även kolumnernas typ (C, I eller F) och längd (antal inmatningspositioner) visas på skärmen, tex C30. När data har matats in i alla kolumner sorteras raden in i tabellen. Därefter frågar systemet efter nästa rad genom att visa den första kolumnen.

Inmatningen avslutas genom att man skriver ett eller två utropstecken (! eller !!). Ett utropstecken medför att inmatningen av den aktuella raden avbryts och att nästa rad kan matas in. Två utropstecken medför att inmatningsproceduren avslutas och att systemet övergår till kommandoläge (QL>). Om man skriver fel och vill göra om inmatningen, skriver man ett utropstecken i stället för data, varvid man kan mata in raden på nytt. Om man skriver ett utropstecken i den första kolumnen, har det samma effekt som två utropstecken, dvs systemet övergår till kommandoläge (QL>).

Om man bara trycker på vagnretur utan att ge något värde, händer följande:

- Om datatypen är C: blanktecken lagras i kolumnen. Därefter visas nästa kolumn.
- Om datatypen är I eller F: inmatningen accepteras inte – odefinierat värde. Systemet ber operatören mata in ett nytt värde. Om man vill lämna kolumnen tom, dvs hoppa över den, kan man ange två apostrofer (' '). Observera att man alltid måste ange värden i primärnyckelkolumnerna.

Om det inmatade värdet skall lagras i en C-kolumn och värdet är mindre än kolumnens längd, lagras värdet med avslutande blanktecken (mellanslag).

Inledande och efterföljande blanktecken i en I- eller F-kolumn är inte signifikanta, när värdet lagras i tabellen.

Alternativ 2: Operatören specificerar kolumnerna

Här tilldelas kolumnerna värden med en rad per kommando. Man måste känna till vilka kolumner, som finns i tabellen. Om inte kan man använda kommandot BESKRIV TABELL för att få en beskrivning av tabellen.

Ordningen på kolumnerna i kommandot har ingen betydelse. Inledande och efterföljande blanktecken är inte signifikanta för värden som lagras i I- eller F-kolumner, se även ovan.

Observera att alla värden som innehåller specialtecken, tex komma, parentes, snedstreck, kolon och citationstecken, måste inledas och avslutas med apostrof (') eller citationstecken (").

Kolumner som inte specificeras i kommandot tilldelas ett odefinierat värde. Ett sådant värde visas med punkter (...) på skärmen. Observera att primärnyckelkolumnerna alltid måste tilldelas värden.

Exempel

Systemet frågar efter värden:

```
QL> Lagra HOTELL;
HID              (C3): H07<VR>
HNAMN           (C20): VIKINGEN<VR>
.                (. ): .
HID              (C3): !!
QL>
```

En ny rad matas in i tabellen HOTELL. Genom att man anger två utropstecken (!!), avslutas inmatningen. I detta läge hade det dock räckt med bara ett utropstecken.

Om raden i stället skall matas in genom att operatören anger värdena direkt i kommandot, ser kommandot ut så här:

```
QL> Lagra HOTELL(HID=H07,HNAMN=VIKINGEN);
```

OMDEFINIERA

Syntax

```
OMDefiniera tabellnamn
        ADDera (kolumnnamn ÄR format,...);
        RADera (kolumnnamn,...);
```

Funktion

Kommandot används när man skall ändra en tabelldefinition. Definitionen kan bara ändras när tabellen inte innehåller några data. Dessutom måste använda-

ren ha X-behörighet till den aktuella databanken. Om man vill ändra en tabell som innehåller data, måste man först tömma tabellen med kommandona KOPIERA ... TILL och RADERA. När man har ändrat tabelldefinitionen återlagras data med kommandot KOPIERA ... FRÅN.

Med OMDEFINIERA-kommandot kan man:

- lägga till nya kolumner, OMDEFINIERA ADDERA,
- ta bort bort kolumner, OMDEFINIERA RADERA

Med OMDEFINIERA ADDERA läggs de nya kolumnerna in efter dem som redan finns i tabellen. Om man vill lägga in en eller flera nya kolumner i mitten, tex för att utöka antalet primärnyckelkolumner, måste man först ta bort alla kolumner "till höger" om de nya kolumnerna. Därefter definierar man om kolumnerna från och med de nya kolumnerna.

Med OMDEFINIERA RADERA tar man bort en eller flera kolumner från tabellen. Även primärnyckelkolumner kan tas bort, men minst en måste lämnas kvar. Det måste alltid finnas en primärnyckelkolumn i en tabell.

Exempel

QL> **Omdefiniera** RUM **Addera** (YTA **är** C3);	Kolumnen YTA läggs till tabellen RUM.
QL> **Omdefiniera** RUM **Radera** (YTA);	Kolumnen YTA tas bort från tabellen RUM.

RADERA

Syntax

RADera tabref <**DÄR** villkorssats>;

Funktion

Kommandot medför att en eller flera rader i en tabell tas bort. Med villkorssatsen anger man vilka rader som skall tas bort. Om villkorssatsen utelämnas, töms hela tabellen. Om BEKRÄFTA-läge är satt (se SÄTT– kommandot), ställs dock följande fråga innan tabellen töms:

Alla rader tas bort ? (J/N):

Man kan ångra sig genom att svara N (nej), varvid operationen avbryts. I annat fall svarar man J (ja).

För ytterligare information om villkorssatsen, se HÄMTA-kommandot.

Om villkorssatsen utelämnas, erfordras X-behörighet för den aktuella databanken. I annat fall räcker S-behörighet.

Exempel

QL> **Radera** NÖJEN;	Hela tabellen NÖJEN töms.
QL> **Radera** NÖJEN **där** NÖJEN.HID = 'H03';	Endast rader som uppfyller angivet villkor tas bort i tabellen.

SKRIV

Syntax

SKRiv tabellnamn <'rubriktext'>;

Funktion

Kommandot medför att den angivna tabellen överförs till en speciell fil på skivan. Tabellen lagras formaterad. Namnet på filen specificeras i tillståndsvariabeln SKRIVFIL.

Filen öppnas av det första SKRIV-kommandot eller HÄMTA/SI/A-kommandot och är öppen ända tills körningen avslutas med SLUTA-kommandot. Filen kan också stängas med något av kommandona SÄTT SKRIVFIL, ÅTERSTÄLL och ÅTERSTÄLL SKRIVFIL. Nästa SKRIV-kommando kommer i så fall att öppna filen på nytt.

Med formaterad menas att utskriften är redigerad med sidnummer och kolumnrubriker på varje sida. Det är också möjligt att ange en rubrik för hela tabellen (rubriktext). Tillståndsvariabeln SL anger antalet rader per sida. SL = 0 (noll) medför att inga sidnummer och kolumnrubriker skrivs ut. Radlängd i antal tecken styrs med tillståndsvariabeln SB. Rubriken för hela tabellen kan bestå av högst SB-2 tecken. Rubriken måste vara omsluten av apostrofer (').

Exempel

QL> **Sätt** SL='50'; QL> **Skriv** PERSONAL 'Förteckning över anställda';	Raderna i tabellen PERSONAL lagras i skrivfilen under rubriken 'Förteckning över anställda'. Sidlängden sätts till 50 rader.

SLUTA

Syntax

SLUta;

Funktion

Kommandot medför att alla tabeller stängs och körningen avslutas.

STATISTIK

Syntax

STAtistik;

Funktion

Medför att körningen lämnar MIMER/QL och går över till statistiksystemet MIMER/ST. I MIMER/ST kan ett 40-tal olika statistiska analyser göras.

STÄNG

Syntax

STÄng databanksnamn<(tabellnamn,...) <,...>;

Funktion

Kommandot stänger angivna tabeller och databanker. STÄNG är alltså motsatsen till ÖPPNA.

Exempel

QL> **Stäng** ANSTDB,HOTELLDB(NÖJEN);

Stänger samtliga tabeller i databanken ANSTDB men enbart tabellen NÖJEN i databanken HOTELLDB.

SÄTT

Syntax

SÄTt <EJUtskrift | UTSkrift> <,RR=RR | n> <,RB=RB | n>
<,MI=100 | n> <,EJEko | EKO> <,AEXitcl=50 | n>
<,ILÄngdm=ILÄNGDCm | n> <,FLÄngdm=FLÄNGDCm | n>
<,LÅNg | KORt> <,SKRivfil=SKRIVF | filnamn> <,EJSpåra | SPÅra>
<,PROCDB=PROCDB | databanksnamn> <,SL=SL | n> <,SB=SB | n>
<,BIBLIOTEK=BIBLIOTEK | procedurbibliotek>
<,EJTransaktion | TRAnsaktion> <,BEKräfta | EJBekräfta>
<,ARBetsdatabank=ARBETSDATABANK | databanksnamn>;

Funktion

Med SÄTT-kommandot kan man förändra tillståndsvariablernas värden. När systemet startas tilldelas samtliga tillståndsvariabler ett standardvärde.

Värdena för följande systemberoende tillståndsvariabler presenteras på skärmen med kommandot KÖR SYSPROC(INFO) (även bilaga A):

RB, RR, ILÄNGDC, FLÄNGDC, SKRIVF, PROCDB, ARBDB, SL, SB och BIBLIOTEK

SÄTT SKRIVFIL medför att utskriftsfilen definieras och att en redan öppnad utskriftsfil stängs.

Exempel

QL> **Sätt Ejutskrift**	Ingen information, som skall visas på skärmen, skall undertryckas.
QL> **Sätt** RR = '50';	Sidornas storlek = 50 rader per sida.
QL> **Sätt** RB = '40';	Radernas längd = 40 tecken per rad.
QL> **Sätt** MI = '100';	Värdet anger hur ofta meddelande om antal hittills utförda operationer skall skickas till terminalen. Detta kan tex användas när man har ett stort antal rader. Operationerna kan ta lång tid och man vill gärna veta att någonting händer. Detta gäller kommandona KOPIERA, SKRIV, ÄNDRA, RADERA (med villkorssats) och HÄMTA (vid läsning till arbetstabellen). Inget meddelande skickas om MI = 0 (noll) eller om UTSKRIFT är satt.
QL> **Sätt** FLÄNGD4 = '10';	Värdet anger antalet tecken som skall reserveras för kolumnerna av typ F4 när de presenteras på skärmen.
QL> **Sätt** SL='20';	Anger sidstorleken (antal rader per sida) i skrivfilen, SKRIVFIL. Överst på varje sida skrivs kolumnernas namn och sidans nummer. Om SL = 0 (noll), skrivs alla rader ut i en följd.
QL> **Sätt** SB='40';	Anger radlängden (antal tecken per rad) i skrivfilen, SKRIVFIL.

UTFÖR

Syntax

UTFör <procedurbibliotek><(procedurnamn)>;

Funktion

Kommandot startar en procedur (procedurnamn), som finns lagrad i ett procedurbibliotek. Biblioteket lagras som en vanlig MIMER-tabell i en speciell procedurdatabank.

Procedurdatabanken specificeras i tillståndsvariabeln PROCDB (kapitel 14). Om procedurbibliotek utelämnas används det bibliotek som specificeras i tillståndsvariabeln BIBLIOTEK. Standardvärdena för dessa tillståndsvariabler är PROCDB respektive användarens namn. Om procedurens namn utelämnas startas proceduren TEMP. TEMP är reserverat och får inte användas som inledande tecken i namn på procedurer som användaren skapar om fler tecken följer. TEMP1 och TEMPER är alltså inte tillåtna, medan TEMP och TEM accepteras. Även namnen SPAR och SIST är på motsvarande sätt reserverade.

MIMER-procedurer innehåller en meny- och frågefunktion. Med det senare menas att proceduren frågar efter information genom att visa ledtexter på skärmen. Om man vill lämna frågeläge skriver man ett eller två utropstecken (! eller !!). Ett utropstecken medför att man kommer tillbaka till föregående nivå i ett menyträd (avsnitt 13.11.1). Om den aktuella nivån är den högsta eller om proceduren inte innehåller menyer, övergår systemet till kommandoläge (QL>). Två utropstecken medför också att systemet övergår i kommandoläge (QL>).

Exempel

QL> **Utför** PROC (HOTPROC);	Proceduren HOTPROC i biblioteket PROC startas.

Kapitel 13 innehåller fler exempel.

VISA

Syntax

VISa ALIas <tabellnamn> **|BUFfertar|ÖPPnade** <databanksnamn>
|SÄTtningar |TIDen|VEM|VERsion;

Funktion

Kommandot presenterar information på skärmen om det pågående arbetspasset.

- VISA ALIAS presenterar samtliga alias för den angivna tabellen eller samtliga tabeller (se även BESKRIV ALIAS).
- VISA BUFFERTAR presenterar status och statistik för vissa interna databuffertar.
- VISA ÖPPNADE presenterar vilka databanker och tabeller som är öppna samt med vilken tillgänglighet de är öppnade (se även BESKRIV MILJÖ).

- VISA SÄTTNINGAR presenterar tillståndsvariablernas värden (se även BESKRIV SÄTTNINGAR).
- VISA TIDEN presenterar datum och tid.
- VISA VEM presenterar information om aktuell användare (namn, ställning och internt användarnummer).
- VISA VERSION presenterar information om den aktuella programversionen enligt följande:

Version	QL	DB:S	MDR
3.3.1	84–12–01	84–01–05	84–01–22

Version	Generation, version och utgåva. Versionsbeteckningen måste uppges om tex ett fel skall rapporteras.
QL, DB:S och MDR:	Modulernas versionsdatum. DB:S anger att det är ett enanvändarsystem. DB:M indikerar ett fleranvändarsystem. MDR syftar på de datorberoende rutinerna.

ÅTERSTÄLL

Syntax

ÅTErställ
<MI><,EKO><,FLÄngdm><,UTSkrift><,BEKräfta>
<,ILÄngdm><,RR><,LÅNg><,RB><,AEXitc1><,SKRivfil>
<,PROCDB><,SL><,BIBliotek><,SB><,SPÅra>
<,ARBetsdatabank><,TRAnsaktion>;

Funktion

Kommandot återställer tillståndsvariablerna till standardvärden. Om ingen tillståndsvariabel specificeras, återställs samtliga tillståndsvariabler och alla öppna databanker stängs. Dessutom stängs utskriftsfilen och alla alias tas bort.

Kommandot ÅTERSTÄLL SKRIFIL medför att utskriftsfilen stängs.

Exempel

QL> **Återställ**;	Samtliga tillståndsvariabler återställs till sina standardvärden.
QL> **Återställ** PROCDB;	Tillståndsvariablen PROCDB återställs till standardvärdet.

ÄNDRA

Syntax

ÄNDra tabref (kolumnnamn = data,...) <**DÄR** villkorssats>;

Funktion

Kommandot tilldelar kolumnerna värden på en eller flera rader i tabellen. Kolumnnamn anger vilka kolumner som skall uppdateras och villkorssatsen anger vilka rader som skall uppdateras. Värdet som skall läggas in i kolumnen ges av 'data'. Alfanumeriska data som innehåller skiljetecken måste omges med apostrofer (' ') eller citationstecken(" "). Värdet i primärnyckelkolumnen kan inte uppdateras. För information om villkorssatsen, se kommandot HÄMTA. För att kunna använda kommandot måste man ha S-behörighet för den aktuella databanken.

Exempel

```
QL> Ändra HOTELL (HNAMN='VIKING')
+>  där HOTELL.HID=H07;
```

Kolumnen HNAMN på raden med HID=H07 ändras.

ÖPPNA

Syntax

ÖPPna </RI/SI/X> databanksnamn <(tabellnamn,...)><,...>;

Funktion

Med ÖPPNA-kommandot anger man vilka databanker och tabeller, som man vill arbeta med samt deras tillgänglighet. Om man vill ändra innehållet i databanken, måste man ange tillgängligheten som /S eller /X, som gör det möjligt att skriva i databanken. Det förutsätter dock att användaren har endera av dessa behörigheter till databanken.

Viss försiktighet måste iakttas beträffande användningen av /X. /X medför att övriga användare i systemet inte kan använda den aktuella databanken. Om tillgänglighet inte anges, gäller /R, som endast gör det möjligt att läsa i databanken. Om tabellerna inte specificeras blir samtliga tabeller i databanken tillgängliga.

Om en tabell (eller databank) har öppnats för läsning och man därefter vill skriva i tabellen, måste man först stänga tabellen (eller databanken) med

kommandot ÅTERSTÄLL eller STÄNG innan man kan öppna den igen för skrivning. Tänk på att ÅTERSTÄLL stänger alla öppna databanker.

Om databankerna eller tabellerna redan är öppna när ÖPPNA-kommandot ges, visas ett felmeddelande på skärmen för de tabeller som redan är öppna. Dessa meddelanden kan emellertid undertryckas med kommandot SÄTT UTSKRIFTER. I så fall visas de inte.

Exempel

QL> **Öppna** HOTELLDB (HOTELL,RUM);	Tabellerna HOTELL och RUM i databanken HOTELLDB öppnas för skrivning.
QL> **Öppna** ANSTDB;	Samtliga tabeller i databanken ANSTDB öppnas för läsning.

Editeringskommandon i procedureditorn

I denna bilaga ges detaljerade beskrivningar av samtliga editeringskommandon i alfabetisk ordning. Beskrivningarna innehåller normalt:

- syntax
- funktionsbeskrivning
- parameterbeskrivning
- exempel

Kommandoorden som används i editeringskommandona kan skrivas i kortform i likhet med alla andra kommandoord i MIMER/QL-systemet. Kortformen av ett kommandoord är alltid de tre första bokstäverna.

I kommandon som arbetar på strängar används skiljetecknet I. Tecknet / kan bytas ut mot något av tecknen " ' eller ? (avsnitt 13.14.3).

DÖP-OM

Syntax

DÖP-om procedur

Funktion

Kommandot medför att aktuell procedur döps om. Det kan också användas för att flytta en procedur till ett annat bibliotek. I det senare fallet skall man ange ett annat bibliotek än det aktuella (även avsnitt 13.14.3).

Parametrar

Parameter	*Förklaring*
Procedur	Nytt procedurnamn på aktuell procedur.

Exempel

Döp-om NBIBL.NYPROC	Aktuell procedur döps om till NYPROC och flyttas till biblioteket NLIB.
Döp-om SVEN	Aktuell procedur döps om till SVEN.

ELIMINERA

Syntax

ELIminera <procedur>

Funktion

Kommandot medför att angiven procedur tas bort från biblioteket. Om procedur utelämnas, tas aktuell procedur bort från biblioteket (även avsnitt 13.14.3).

Exempel

Eliminera NBIBL.PROC	Proceduren PROC tas bort från biblioteket NLIB.
Eliminera	Aktuell procedur tas bort.

ERSÄTT

Syntax

ERSätt <radnummerintervall> (1)

eller

ERSätt <radnummerintervall>/sträng1/sträng2/ (2)

Funktion

Kommandot medför att angivna rader ersätts i den aktuella proceduren (1 ovan). Om kommandot skrivs enligt 2 ovan har det samma funktion som ÄNDRA-kommandot.

FLYTTA

Syntax

FLYtta <mål,><procedur,><radnummerintervall>

Funktion

Kommandot flyttar en eller flera rader från en annan eller aktuell procedur till den aktuella proceduren. Raderna, som flyttas, numreras om i steg om 1 med början på angivet värde – <mål>.

Parametrar

Parameter	*Förklaring*
Mål	Den första raden som flyttas tilldelas detta radnummer. Numret får bestå av upp till fyra siffror. Om numret utelämnas, placeras de angivna raderna sist i den aktuella proceduren.
Procedur	Namnet på den procedur från vilken raderna skall flyttas (avsnitt 13.14.3). Om procedur utelämnas, flyttas de angivna raderna i den aktuella proceduren.
Radnummer	Sekvensnummer. Raderna som skall flyttas (avsnitt 13.14.3).

Exempel

Flytta 115,70–90	Raderna 70—90 flyttas om inom den aktuella proceduren. Första raden (nr 70) får det nya numret 115.

HJÄLP

Syntax

HJÄlp <kommando>

Funktion

Med kommandot erhålls information på skärmen om editeringskommandon (syntax och beskrivning) och andra begrepp. Även begränsad information om procedurkommandona kan erhållas. Namnet på kommando etc om vilket information önskas kan anges i kortform (tre första bokstäverna enligt översikten nedan).

Parametrar

Editeringskommandon

DÖP-om	KOPiera	ORDna
ELIminera	KORrigera	RADera
ERSätt	LISta	SLUta
FLYtta	LAGRA	SPAra
HJÄlp	NUVarande	SUDda
INVentera	OMNumrera	ÄNDra

Procedurkommandon

AVSluta	KOMentar	SKRiv
FRÅga	MENy	STOpp
HOPpa	NÄSta	TILldela
INLed	OM	

Diverse begrepp

ARItmetiskt uttryck	INTervall	PROcedur
EDItering	PRN (procedurnamn)	STRäng
INMatning		

INVENTERA

Syntax

INVentera <<databank.>bibliotek>

Funktion

Samtliga procedurnamn i angivet bibliotek visas på skärmen. Om bibliotek utelämnas, visas procedurnamnen i det bibliotek som aktuell procedur ligger i.

Parametrar

Parameter	*Förklaring*
Databank	1–8 tecken (avsnitt 13.14.3).
Bibliotek	1–8 tecken.

KOPIERA

Syntax

KOPiera <radnummer,><procedurnamn,><radnummerintervall>

Funktion

Kommandot kopierar flera rader från aktuell procedur eller en annan procedur till den aktuella proceduren. Kopierade rader numreras om i steg om 1 med början på värdet som anges av <radnummer>.

Parametrar

Parameter	*Förklaring*
Radnummer	Radnummer med max 4 siffror. De kopierade raderna läggs in i den aktuella proceduren med början på angivet nummer. Om numret utelämnas, läggs raderna in i slutet av proceduren.
Procedurnamn	Namnet på den procedur från vilken raderna skall kopieras (avsnitt 13.14.3). Om namnet utelämnas, kopieras angivna rader i den aktuella proceduren.
Radnummerintervall	Raderna som skall kopieras (avsnitt 13.14.3).

Exempel

Kopiera MENY	Proceduren MENY kopieras och läggs in i slutet av den aktuella proceduren.
Kopiera 115,NBANK.NBIBL.TEXT,20–100	Raderna 20–100 i proceduren TEXT i biblioteket NBIBL i databanken NBANK kopieras in i den aktuella proceduren. Raderna tilldelas numren 115, 116 osv.

KORRIGERA

Syntax

KORrigera <radnummerintervall> /sträng1/sträng2/ (1)

eller

KORrigera <radnummerintervall> #kolumn#sträng2# (2)

Funktion

Kommandot enligt (1) medför att sträng2 skrivs över sträng1 i hela radnummerintervallet från och med första förekomst. Kommandot enligt (2) medför att sträng2 skrivs över sträng1 i radnummerintervallet med start i angiven kolumn.

Parametrar

Parameter	*Förklaring*
Radnummerintervall	Raderna som skall bearbetas.
Sträng1 och sträng2	1–68 tecken (avsnitt 13.14.3).
#	Indikerar en bestämd kolumnposition.
Kolumn	Kolumnens nummer på tabellraden. Maximalt två siffror. Första kolumnen har nummer 1.

Exempel

Korrigera 50 ' IPN ' INPUT '	Strängen IPN på rad 50 skrivs över med INPUT.
Korrigera 20–100#2#**#	Innehållet i kolumn 2 och 3 på raderna 20–100 skrivs över med tecknen **.

LAGRA

Syntax

LAGra <radnummerintervall> (1)

eller

LAGra <radnummerintervall>/sträng1/sträng2/ (2)

eller

LAGra <radnummerintervall> #kolumn#sträng2# (3)

Funktion

Kommandot enligt (1) medför att en eller flera rader läggs in i den aktuella proceduren. Kommandot enligt (2) medför att sträng2 läggs in omedelbart framför första förekomst av sträng1, medan kommandot enligt (3) medför att sträng2 läggs till omedelbart framför angiven kolumn (3). Bearbetningen sker på angivna rader.

Parametrar

Parameter	*Förklaring*
Radnummerintervall	Raderna som skall bearbetas (avsnitt 13.14.3).
Sträng1 och sträng2	1–68 tecken (avsnitt 13.14.3).
#	Indikerar en bestämd kolumnposition.

Kolumn	Kolumnens nummer i tabellen. Numret får bestå av upp till två siffror. Första kolumnen är nummer 1.

Vid kommandovariant (1) efterfrågas varje ny rad genom att systemet visar motsvarande radnummer. Detta pågår ända tills något av följande inträffar:

- Sista raden har matats in.
- Två utropstecken (!!) matas in (tillbaka till editeringsläge).
- Ett utropstecken (!) matas in (tillbaka till inmatningsläge).
- Ett utropstecken direkt följt av UPDATE-kommando matas in.

Exempel

Lagra 14–16	Tre nya rader skall matas in. Dessa får radnumren 14, 15 och 16.
Lagra 21-	Nya rader skall matas in. Dessa får radnumren 21, 22 osv.
Lagra 50/MODELL/NY/	Strängen NY läggs in omedelbart framför strängen MODELL på rad 50.

LISTA

Syntax

LISta <procedur,><radnummerintervall></sträng1/>

Funktion

Kommandot medför att de rader, som innehåller sträng1 i angiven procedur, visas på skärmen. Om procedur utelämnas, listas raderna i den aktuella proceduren. Om radnummer utelämnas, listas samtliga rader, som innehåller sträng1, i proceduren. Om sträng1 utelämnas, listas samtliga rader.

Parametrar

Parameter	*Förklaring*
Procedur	1–8 tecken (avsnitt 13.14.3).
Radnummerintervall	Raderna som skall bearbetas (avsnitt 13.14.3).
Sträng1	1–68 tecken (avsnitt 13.14.3).

Exempel

Lista 'PETER'	Samtliga rader, som innehåller 'PETER', i aktuell procedur, listas på skärmen.
Lista	Hela proceduren listas.
Lista NBANK.NBIBL.TEXT,20–100	Raderna 20–100 i proceduren TEXT i biblioteket NLIB i databank NBANK, listas på skärmen.

NUVARANDE

Syntax

NUVarande (1)

eller

NUVarande <<databank.>bibliotek.>procedur (2)

Funktion

Kommandot enligt (1) visar namnet på aktuell procedur enligt följande:

Aktuell procedur: Bibliotek(procedur) i databank.

Kommandot enligt (2) skiftar aktuell procedur.

Parametrar

Parameter	*Förklaring*
Databank	1–8 tecken (avsnitt 13.14.3).
Bibliotek	1–8 tecken.
Procedur	–8 tecken.

Exempel

Nuvarande NBIBL.NPROC	Byter aktuell procedur till NPROC i biblioteket NBIBL.

OMNUMRERA

Syntax

OMNumrera

Funktion

Kommandot medför att raderna i den aktuella proceduren numreras om i steg om 10 med början på 10. **Observera**: Radnummer för eventuella +HOPPA-satser ändras inte — sådana radnummer måste korrigeras manuellt.

ORDNA

Syntax

ORDna <radnummerintervall>/sträng1/sträng2/

Funktion

Första förekomst av sträng1 byts ut mot innehållet i sträng2 på angivna rader (se även kommandot ÄNDRA).

Parametrar

Parameter	*Förklaring*
Radnummerintervall	Raderna som skall bearbetas (avsnitt 13.14.3).
Sträng1 och sträng2	1–68 tecken.

Exempel

Ordna 30–90/GAMMAL/NY/	Första förekomst av strängen GAMMAL på raderna 30–90 byts ut mot NY.
Ordna 'KARL'SVEN'	Första förekomst av strängen KARL på samtliga rader i proceduren byts ut mot SVEN.

RADERA

Syntax

RADera radnummerintervall

Funktion

Kommandot raderar en eller flera rader från den aktuella proceduren.

Parametrar

Parameter	*Förklaring*
Radnummerintervall	Anger vilka rader som skall tas bort (avsnitt 13.14.3).

Exempel

Radera 20	Rad 20 raderas.
Radera 200-	Samtliga rader från och med rad 200 raderas.
Radera -50	Samtliga rader till och med rad 50 raderas.
Radera 30–60	Raderna från och med rad 30 till och med rad 60 raderas.
Radera *-150	Samtliga rader från och med aktuell rad (*) till och med rad 150 raderas.

SLUTA

Syntax

SLUta

Funktion

Editeringen avslutas och systemet går över till kommandoläge (kommandosymbolen QL> visas på skärmen). Kommandot har samma verkan som inmatning av två utropstecken (!!) i inmatnings- eller editeringsläge.

SPARA

Syntax

SPAra procedur

Funktion

Kommandot medför att en kopia av aktuell procedur sparas under ett annat namn.

Parametrar

Parameter	*Förklaring*
Procedur	1–8 tecken (avsnitt 13.14.3).

Exempel

Spara NBIBL.INMATA	En kopia av den aktuella proceduren sparas under namnet INMATA i biblioteket NBIBL.

SUDDA

Syntax

SUDda <radnummerintervall>/sträng1/sträng2/ (1)

eller

SUDda <radnumerintervall><radnummer>#kolumn#sträng2# (2)

Funktion

Kommandot enligt 1 medför att det antal tecken som finns i sträng2, raderas från de angivna raderna. Raderingen börjar från och med första förekomst av sträng1. Om kommandot skrivs enligt 2 raderas tecknen med början i angiven kolumn.

Parametrar

Parameter	*Förklaring*
Radnummerintervall	De rader som skall bearbetas (avsnitt 13.14.3).
Kolumn	Kolumnens nummer. Numret får bestå av upp till två siffror. Första kolumnen på raden har nummer 1.

Exempel

Sudda 50'INM'.....'	Radera fem tecken med början på tecknen INM på rad 50.
Sudda 20–100#2#**#	Radera två tecken med början i kolumn 2 på raderna 20–100.

ÄNDRA

Syntax

ÄNDra <radnummerintervall>/sträng1/sträng2/

Funktion

ÄNDRA-kommandot medför att sträng1 byts ut mot sträng2 i de angivna raderna.

Parametrar

Parameter	*Förklaring*
Radnummerintervall	Rader där ändring skall ske (avsnitt 13.14.3).
Sträng1 och sträng2	1–68 tecken. Alla tecken utom tecknet / kan användas.

Exempel

Ändra 30–90/GAMMAL/NY/	Strängen GAMMAL byts ut mot strängen NY på raderna 30–90.
Ändra 'KARL'SVEN'	Namnet KARL byts ut mot namnet SVEN på samtliga rader i proceduren.

Felmeddelanden i MIMER/QL

Nedanstående felmeddelanden kan erhållas på skärmen under ett arbetspass i MIMER/QL. Ett felmeddelande består av dels en sifferkod, dels en förklarande text.

Den förklarande texten finns i både kort form (markerat med fet stil) och mer utförlig form. Användaren väljer vilkendera som skall visas på skärmen med kommandot

Sätt Kort | Lång

Kod Meddelandetext

Felmeddelanden vid QL-kommandon

2 Syntax — Felplacerat skiljetecken
Kommandot innehåller ett felplacerat skiljetecken.

4 Syntax — Otillåten ordningsföljd mellan skiljetecken
Kommandot innehåller en följd av skiljetecken som kommer i fel ordning.

6 Konvertering — Tecken ej numeriskt
Kommandot innefattar en parameter som skall vara numerisk (en sifferföljd). Aktuellt parametervärde är ej numeriskt.

8 För långt kommando — buffertarea överfylld
Kommandot är för långt. Tillräckligt utrymme i en intern buffertarea saknas. Försök utföra samma operation med ett kortare kommando eller flera kommandon efter varandra. Går inte det — kontakta din MIMER-ansvariga och fråga om QL-installationen kan modifieras med större buffertarea.

10 För långt kommando — översättningstabell överfylld
Kommandot är för långt. Tillräckligt utrymme i en intern buffertarea saknas. Försök utföra samma operation med ett kortare kommando eller flera kommandon efter varandra. Går inte det — kontakta din MIMER-ansvariga och fråga om QL-installationen kan modifieras med större buffertarea.

11 Syntax — Parentes- eller apostroftecken saknas
Kommandot innehåller en eller flera vänsterparenteser som inte motsvaras av högerparenteser eller också fattas apostrofteckнet till höger om en teckensträng.

25 Hämta — tabell och kolumnlista passar inte ihop
Vid ett HÄMTA-kommando som använder en permanent tabell som ar-

betstabell passar inte arbetstabellens kolumndefinitioner ihop med kolumnlistan (söklistan). Ett kolumnnamn i listan kan då vara felstavat eller också innehåller listan ett kolumnnamn som inte definierats i samband med arbetstabellen.

26 Överföringsfel — mata in sista raden på nytt
Sista raden gick inte att läsa — mata in på nytt!

27 För långt databanksnamn
Namnet på en databank kan inte var mer än 8 tecken. Längden anges med variabeln DBNMXL (se handboken, bilaga A).

28 För långt filnamn
Filnamnet är för långt. Tillåten längd anges med variabeln LFN (se handboken, bilaga A).

29 Namn på databanken saknas
Databanksparametern saknas.

30 Tabellen är redan definierad — använd OMDEFINIERA-kommandot
En redan definierad tabell kan inte definieras på nytt om den inte först tas bort med kommandot ELIMINERA TABELL. Omdefiniering skall utföras med OMDEFINIERA RADERA och OMDEFINIERA LÄGG-TILL.

31 Omdefiniering av tabell som innehåller data är otillåten
När en tabell innehåller data kan den inte omdefinieras förrän data flyttats undan. Börja med att kopiera tabellen med KOPIERA. Ta därefter bort data med RADERA och omdefiniera tabellen med OMDEFINIERA RADERA och/eller OMDEFINIERA LÄGG-TILL. Lägg så tillbaka data med kopiera.

32 Otillåten datatyp
En formatangivelse innehåller en otillåten datatyp. Tillåtna datatyper är C (alfanumeriska tecken), I (heltal) och F (flyttal). En del datorer godtar endast C vid kommandot KOPIERA. Se bilaga A.

33 Systemtabell kan ej uppdateras
I samband med definiering eller omdefiniering av en tabell har ett fel uppstått. Felet kan vara allvarligt och åtföljas av MIMER/DB-fel.

34 Tabell med angivet namn har ej öppnats
Tabellen har inte deklarerats med ÖPPNA-kommandot.

35 Kan ej öppna tabellpekare på nytt
I samband med definiering eller omdefiniering av en tabell har ett fel uppstått. Felet kan vara allvarligt och åtföljas av databasfel. Underrätta din MIMER-ansvariga.

36 Otillåten relationsoperator
Ett villkorsuttryck i en DÄR-sats innehåller en otillåten eller odefinierad relationsoperator.

37 Tabellen finns inte i angiven databank
En angiven tabell saknas i den öppnade databanken.

38 Otillåten logisk operator
Ett villkorsuttryck i en DÄR-sats innehåller en otillåten eller odefinierad logisk operator. Definierade logiska operatorer är OCH och ELLER.

39 Otillåtet (eller otillåten kombination av) kommandotillägg
Kommandot innehåller ett obekant kommandotillägg eller flera kommandotillägg som är oförenliga.

40 Odefinierad databank
Databanken är odefinierad. Definiera den med DBA-kommandot, förutsatt att du är databasadministratör.

41 Redan öppnad för läsning av procedurtolken
En procedur innehåller ett kommando som kräver behörighet S eller X till en databank (t ex ÖPPNA-kommandot). Databanken har dock redan öppnats med behörighet R av procedurtolken och måste därför öppnas med önskad behörighet innan proceduren verkställs.

42 Databanken redan öppnad med annan och oförenlig behörighet
Du försöker öppna en redan öppnad databank men din behörighet är oförenlig med den redan använda. Om en tabell (eller databank) har öppnats med läsbehörighet (R) kan den inte senare öppnas med skrivbehörighet (S eller X) utan att först stängas med kommandona STÄNG eller ÅTERSTÄLL.

43 Databanken ej öppnad
Kommandot kan inte utföras med en ej öppnad databank.

44 Formellt fel — otillåten behörighet
Tillåtna behörigheter är R, S och X.

45 Operationen kräver skrivbehörighet (S eller X)
Kommandot avser skrivning i databanken. Denna har dock öppnats med endast läsbehörighet (R). Stäng databanken och öppna den på nytt med behörighet S eller X.

46 För lång post
En post som skall kopieras är för lång för en intern buffertarea. Försök minska postlängden, t ex genom att i kommandot hänvisa till färre kolumner. Går inte det — kontakta din MIMER-ansvariga och fråga om QL-installationen kan modifieras med större buffertarea.

47 Använd relationsoperator kan ej förekomma vid aktuell datatyp
Vissa relationsoperatorer är tillåtna endast i samband med datatyp C, ej I eller F. Dessa operatorer är %,/%, * och /*.

48 Operationen kräver behörighet X
Vid utförandet av kommandot krävs behörighet X. Sådan innehas av databasadministratören eller ägarna till databanken.

49 Odefinierat kommando
Du har gett ett kommando som inte hör till QL. QL-kommandon visas med HJÄLP-kommandot.

50 Odefinierad tillståndsvariabel
Du har angett en odefinierad variabel i ett SÄTT- eller ÅTERSTÄLL-kommando.

51 Skrivfilen i QL gick ej att öppna
Du har försökt öppna skrivfilen i QL men det har inte gått.

52 En eller flera parametrar saknas
Kommandot kräver tillägg av en eller flera parametrar. Använd HJÄLP-kommandot.

53 Otillåtet värde på tillståndsvariabel
Du har försökt sätta ett otillåtet värde på en tillståndsvariabel.

54 QL:s dynamiska arbetsarea full
En intern arbetsarea är för liten vid tolkningen av ett kommando. Kontakta din MIMER-ansvariga och fråga om QL-installationen kan modifieras med större area.

55 Otillåten längd på kolumner
Du har angett ett format med otillåten längd.

56 Primärnycklar kan ej uppdateras
Värden på primärnycklar kan ej ändras. Ta istället först bort raden och skjut därefter in en ny rad med det nya värdet som primärnyckel.

57 Primärnyckelkolumn(er) saknas
Värden måste finnas i alla primärnyckelkolumner.

58 Behörighet ej tillräcklig
Du har inte den behörighet som krävs för operationen.

59 Odefinierad omdefiniering
Angiven kommandooperation vid omdefiniering är odefinierad. Giltiga operatorer är LÄGG-TILL och RADERA.

60 Odefinierat värde ej godtagbart i primärnyckelkolumn
Primärnyckelkolumner får ej sakna värde.

61 Kolumn som skall läggas till finns redan
Du har försökt lägga till en kolumn som redan finns definierad.

62 Kolumn som ska tas bort finns inte
Du har försökt ta bort en kolumn som inte finns definierad.

63 Databanken blockerad av andra. Försök igen!
Du har försökt öppna en databank men inte lyckats. Antingen har du begärt X-behörighet sedan andra användare redan öppnat databanken. Eller också har du inte begärt X-behörighet men någon annan har gått in med sådan före dig.

64 För många databanker öppnade
En intern buffertarea är full p g a att alltför många databanker öppnats. Stäng dem som du kan avvara.

65 Formellt fel — DÄR-sats felaktigt angiven
Du har överträtt de formella reglerna för DÄR-satser.

66 Formellt fel — okänt skiljetecken mellan kommandoord
Formellt fel som inte närmare kan preciseras.

67 Odefinierad tabellangivelse — ALIAS- eller ÖPPNA-kommando saknas
Kommandot innehåller en tabellangivelse (tabref, dvs ett alias eller ett tabellnamn) som är odefinierad. I fallet tabellnamn är det kommandot ÖPPNA som utelämnats, i fallet alias kommandot ALIAS.

68 Ändring eller borttagning av tabellangivelse saknas i DÄR-sats
Tabellangivelse som motsvarar den tabell i vilken ändring eller borttagning skall ske måste finnas med i åtminstone ett villkor.

69 Odefinierat kolumnnamn
Kommandot innehåller ett okänt kolumnnamn.

71 För många MIMER-tabeller öppnade
Du har öppnat så många tabeller att en intern arbetsarea är full. Stäng ovidkommande tabeller med STÄNG. Om det inte hjälper — kontakta din MIMER-ansvariga och fråga om QL-installationen kan modifieras med en större area.

72 För många alias använda
Du har angett så många alias (pekare) att en intern arbetsarea är full. Rensa i arean genom att ta bort ovidkommande tabeller med STÄNG-komman-

dot. Om du ändå får problem när du fortsätter att ställa frågor — kontakta din MIMER-ansvariga och fråga om QL-installationen kan modifieras med en större area.

73 Cirkulär kopplingsväg
En TILL-tabref får inte avse en tabell som redan tidigare passerats på kopplingsvägen.

74 För långt kolumnnamn
Ett kolumnnamn får vara högst 8 tecken långt.

75 För många kolumner i söklistan
Söklistan innehåller så många kolumner att en intern arbetsarea är full. Gör om kommandot med en kortare söklista, eller kontakta din MIMER-ansvariga och fråga om QL kan modifieras med en större area.

76 Allvarligt MIMER/DB-fel

77 Tabell blockerad av annan användare. Försök igen!
Du har försökt öppna en tabell men inte lyckats. Antingen har du begärt X-behörighet sedan andra användare redan öppnat tabellen. Eller också har du inte begärt X-behörighet men någon annan har gått in med sådan före dig.

78 Alias kan ej definieras — tabellen har öppnats med X-behörighet
En tabell som öppnats med X-behörighet kan inte ges något alias.

80 Raden ej inlagd, då den redan finns
Du försöker lägga in en rad som har samma värde på primärnyckeln som en redan inlagd rad. Den nya raden tas inte emot.

81 För långt namn på en MIMER-tabell
Ett tabellnamn får vara högst 8 tecken långt.

82 Kan ej definiera index
Fel vid uppdatering av en systemtabell. Felet kan vara allvarligt. Underrätta din MIMER-ansvariga!

83 Transaktion misslyckad — försök igen!
Kommandot har misslyckats p g a att genererad transaktion ej utförts!

84 Kan ej definiera tabellangivelser
Ospecificerat fel i samband med definiering av tabref. Kan indikera ett allvarligt fel i MIMER. Underrätta din MIMER-ansvariga!

86 Borttagning av ett index har misslyckats
Fel vid uppdatering av en systemtabell. Felet kan vara allvarligt. Underrätta din MIMER-ansvariga!

88 Villkor kopplade med ELLER kräver samma vänster-tabell-angivelser
Otillåten kopplingsväg, tabellangivelser felaktigt hopkopplade. Den kopplingsväg som följer av villkoren i en DÄR-sats är otillåten eller tvetydig.

90 Söklistan innehåller tabellangivelse som ej finns med i DÄR-sats
Söklistan innehåller en tabellangivelse som inte finns med i den kopplingsväg som anges i DÄR-villkoren

95 Den sekvensiella filen kunde ej öppnas
Ditt försök att öppna en sekvensiell fil i samband med kommandot KOPIERA har misslyckats. Det är sannolikt att orsaken ligger i det datorsystem du använder. Se bilaga A för ytterligare information.

96 Kan ej öppna hjälptabell
Ditt försök att öppna QL:s hjälptabell har misslyckats. Underrätta din MIMER-ansvariga!

97 Formellt fel — TILL/FRÅN saknas i KOPIERA-kommando
Formellt fel i ett KOPIERA-kommando. Parametern som anger kopieringsriktning saknas.

98 Formellt fel — för många parametrar i HJÄLP-kommando
Högst två parametrar kan förekomma i ett HJÄLP-kommando, t ex DEFINIERA TABELL eller BESKRIV TABELL.

99 Arbetsdatabank odefinierad
Arbetsdatabanken (ARBDB) är odefinierad och access är därför omöjlig. Om du använder QL i ett fleranvändarsystem — underrätta din MIMER-ansvariga.

101 Felaktig språkangivelse
Du har angivit ett språk som ej finns definierat i systemdatabanken SYSQL.

102 Formellt fel i filnamnsangivelse.

103 Filen finns inte

104 Du försöker bryta igenom filskyddet

105 För många filer har öppnats

106 Diskutrymmet har tagit slut

108 Felaktig tabell för språkdefinition i SYSQL
Den tabell som skall innehålla alternativa språk-definitioner finns ej definierad i systemdatabanken SYSQL.

Felmeddelanden vid procedurer

150 Kan ej öppna procedurdatabanken
Procedurdatabanken kan inte öppnas. Undersök om angivet namn på databanken är riktigt.

151 Odefinierat procedurbibliotek i UTFÖR-kommando

152 Odefinierad procedur (UTFÖR)
Procedur finns inte i angivet procedurbibliotek.

153 Maximalt antal nivåer vid procedurkapsling överskridet
Gränsen för antalet möjliga nivåer i en procedur (dvs maximiantalet samtidigt aktiva procedurer) överskriden.

154 Reserverat procedurnamn i UTFÖR-kommando
Procedurnamn får inte börja med ARB, SIST och SPAR.

160 Formellt fel — otillåtet variabelnamn
Variabelnamn består av ett specialtecken (&&) samt 1–8 valfria tecken, följt av ett skiljetecken.

161 Formellt fel — tilldelningsoperator saknas (TILLDELA)
Variabelnamnet måste åtföljas av en tilldelningsoperator (=).

162 Formellt fel — otillåtet uttryck till höger om TILLDELA-kommando

163 Formellt fel — strängskiljetecken saknas
Strängskiljetecken skall finnas med.

164 Formellt fel — för långt kommando
Fullt i en buffertarea, försök dela upp kommandot.

165 Formellt fel — variabel skall finnas med i OM-SÅ-kommando.
Endast en variabel kan förekomma i vänstra delen av ett logiskt uttryck i ett OM-SÅ-kommando.

166 Formellt fel — otillåten relationsoperator i OM-SÅ-kommando

167 Formellt fel — SÅ saknas i OM-SÅ-kommando
Ett logiskt uttryck måste följas av SÅ.

168 Formellt fel — procedurnamn skall finnas med i MENY-kommando
MENY måste följas av ett procedurnamn.

169 För långt namn på identifierare för radvis bearbetning
Namn på HÄMTA-kommando för radvis bearbetning består av 1–8 tecken.

170 Odefinierad variabel
Du har använt en variabel som inte definierats.

171 Otillåtet variabelvärde i numeriskt uttryck
Fel har uppstått vid försök att omvandla värdet till heltal. Värdet måste vara en följd av numeriska tecken.

172 För stora heltalsvärden i numeriskt uttryck
Din dator klarar inte så stora heltalsvärden.

173 Otillåten ordningsföljd mellan aritmetiska operatorer

174 Stacken full vid beräkning av numeriskt uttryck
Internt minne fullt, dela upp uttrycket.

175 Parentes utan motsvarighet förekommer i TILLDELA-kommando

180 Odefinierad procedur i MENY-kommando
Angiven procedur finns inte.

181 Otillåtet radnummer i HOPPA-kommando
Radnumret måste vara ett heltal.

182 Ledtext i kommando som tar mer än en rad
Ledtexten måste avslutas inom en rad.

183 Tabell redan öppnad med X-behörighet i INLED-kommando
Tabellen skulle inte ha öppnats med X-behörighet, eftersom den senare används i en operation för radvis åtkomst.

184 Identifierare för radvis bearbetning upptagen (INLED)

185 Odefinierad identifierare för radvis bearbetning (NÄSTA,AVSLUTA)

186 Otillåten avslutning av procedur
Ofullständigt procedurkommando avslutar en procedur.

187 QL:s dynamiska minne för procedurtolk fullt
En intern arbetsarea räcker inte till för angiven procedur. Areans storlek kan dock lätt ökas av din MIMER-ansvariga.

188 Det finns inga fler rader att läsa för detta NÄSTA-kommando
När inga fler rader finns att läsa för ett NÄSTA-kommando, så signaleras detta med att systemvariabeln &*ANTAL sätts till 0. Om man därefter ändå försöker att läsa rader betraktas detta som ett fel och denna felkod signaleras.

189 Formellt fel funnet i INLED-kommando

190 Ofullständig procedursats

191 Otillåten frågevariant
Enda giltiga kommandotillägg är /VERSALER.

198 Överföringsfel — mata in sista raden på nytt
Den rad du senast försökte mata in kom inte med. Mata in den igen!

199 Programfel i QL:s procedurhanterare
Fel som inte får förekomma — var snäll och rapportera det på MIMER:s felrapport!

Varningar

201 Kolumnen längre än RL — visst innehåll stympat
Längden hos en kolumn är större än värdet i variabeln RL. Endast det kolumninnehåll som motsvarar av RL angiven längd visas.

204 Kolumnlängd större än SB — utskrivet värde stympat
Längden på en tabell utskriven med SKRIV-kommandot är större än vad som motsvarar värdet på tillståndsvariabeln SB. Endast det kolumninnehåll som motsvarar av SB angiven längd skrivs ut. Värdet på SB kan ändras med SÄTT-kommandot.

205 Samma kolumnnamn två eller flera gånger
Namnet på en kolumn upprepas i kolumnlistan i ett DEFINIERA TABELL-kommando. Den första kolumndefinitionen kommer att gälla medan övriga lämnas utan avseende.

207 Dubletter kan inte loggas vid KOPIERA LADDA
Dublettrader (som avser primärnycklar) kan inte upptäckas och loggas vid KOPIERA LADDA.

208 Kommandotillägg för utmatningsenhet lämnad utan avseende
Kommandot ger utmatning endast på terminal.

209 "Skenbar fleranvändarsituation" slår av transaktionshanteringen
"Skenbar fleranvändarsituation" föreligger när användare med S-ställning kör med enanvändare. Den gemensamma SYSDB öppnas då för samtidig läsning. Även om TRANSDB (och LOGDB) är definierade kan ingen transaktionshantering eller loggning ske.

210 Kolumnlängd större än RL
Värdet förkortas. Längden av en kolumn i en tabell är längre än värdet på tillståndsvariabeln RL. Rader som visas förkortas till RL-tecken. Värdet på RL kan ändras med SÄTT-kommando.

211 Extra data i filen för fördefinierade kommandon ignorerades
Ett ofullständigt QL-kommando påträffades i filen för fördefinierade kommandon (semikolon saknas).

Felmeddelanden i procedureditorn

301 Odefinierat procedurbibliotek

302 PROCDB redan öppnad för läsning

303 Odefinierad editeringsfunktion

304 Ogiltigt intervall

305 Bibliotek eller procedur får inte anges

306 Sträng 2 saknas i ORDNA/ÄNDRA/LÄGG TILL/KORRIGERA/ SUDDA

307 Eftersom du vill ta bort samtliga rader — använd ELIMINERA

308 Nytt procedurnamn utelämnat

309 Minst en rad har fått nytt nummer /varning/

310 Position kan inte anges vid ÄNDRA eller ORDNA

311 Ogiltig position

312 Procedurnamnet finns redan

313 Odefinierad procedurdatabank

314 Du har inte skrivbehörighet till denna PROCDB

315 Databanksfel

316 Tillfälligt systemfel. Försök igen!

317 Formellt fel (skiljetecken)

318 Reserverat procedurnamn (TEMP, SIST, SPAR)

MIMER HOTELL AB — vårt övningsexempel

Övningsexemplet MIMER HOTELL AB följer med vid leverans av MIMER/QL. Varje ny användare kan alltså utnyttja det vid inlärning.

I denna bilaga beskrivs

- hur exemplet görs tillgängligt
- ett fullständigt arbetspass, byggt på vissa förutsättningar

Hur exemplet görs tillgängligt

Det är lämpligast att definiera exemplets databanker i enanvändarmiljö så att varje användare kan ha sina egna databanker. De kommandon som används för att köra olika program kan variera mellan olika datorsystem. Om du är osäker på vilket kommando du skall använda så kan du läsa i den USER-guide som medföljer leveransen eller fråga din systemansvariga.

Hur exemplet blir tillgängligt kan variera från installation till installation. Innan det kan köras måste under alla förhållanden en SYSDB finnas upplagd. Nedan beskrivs de alternativ som kan förekomma.

Du lägger själv upp exemplet

Du börjar med att under din dators operativsystem köra hjälpprogrammet SYSDBGEN, som skapar en SYSDB. Det sker genom att du ger kommandot:

```
run SYSDBGEN
```

Du får då upp följande:

```
Define DBA-Administrator and create SYSDB
-----------------------------------------

DBA username (max. 8 chars) :
```

Här kan du antingen svara med användarnamnet DONALD eller också med ett eget användarnamn (högst 8 tecken).

När du angett användarnamn och tryckt på RETURN (motsvarande) får du upp ytterligare en fråga:

```
Define DBA-Administrator and create SYSDB
-----------------------------------------

DBA username (max. 8 chars) : DONALD
DBA password (max. 8 chars) :
```

Du svarar på motsvarande sätt, antingen med lösenordet HEMLIGT eller med ett eget lösenord (högst 8 tecken).

Efter svar och RETURN (motsvarande) kommer en tredje fråga:

```
Define DBA-Administrator and create SYSDB
-----------------------------------------

DBA username (max. 8 chars) : DONALD
DBA password (max. 8 chars) : HEMLIGT
Filename for SYSDB          :
```

Den besvaras med namnet på en operativsystemfil där SYSDB skall finnas.

Sedan får du en fråga om denna fils initiala storlek. Hur stor SYSDB behöver vara bestäms av storleken på den applikation man skall lägga upp. I detta fall räcker det med 10 sidor:

```
Define DBA-Administrator and create SYSDB
-----------------------------------------

DBA username (max. 8 chars) : DONALD
DBA password (max. 8 chars) : HEMLIGT
Filename for SYSDB          : SYSDB
Initial size of SYSDB       : 10
```

Därefter kommer en fråga om man vill använda transaktionshantering. I och med att detta är en testapplikation är det inte nödvändigt, och du besvarar därför den frågan med nej:

```
Define DBA-Administrator and create SYSDB
-----------------------------------------

DBA username (max. 8 chars) : DONALD
DBA password (max. 8 chars) : HEMLIGT
Filename for SYSDB          : SYSDB
Initial size of SYSDB       : 10
Transdb wanted! (Y/N)       : N
```

Efter detta återstår bara att definiera SYSQL:

```
Define DBA-Administrator and create SYSDB
-----------------------------------------

DBA username (max. 8 chars) : DONALD
DBA password (max. 8 chars) : HEMLIGT
Filename for SYSDB          : SYSDB
Initial size of SYSDB       : 10
Transdb wanted? (Y/N)       : N
SYSQL wanted? (Y/N)         : Y
Filename for SYSQL          : SYSQL
```

Här svarar du med filnamnet för den SYSQL som medföljer vid leveransen av MIMER/QL.

Exemplet är installerat

En SYSDB finns. HOTELLDB och ANSTDB är definierade. Du kan gå direkt med följande kommando:

```
run SQLS
```

varvid följande kommer upp:

```
**************************
*                        *
*       MIMER/QL         *
*      Version 3.3.1     *
*                        *
**************************
```

Username:

Du svarar med det användarnamn som du angav när programmet SYSDBGEN kördes. Nästa ledtext är:

Password:

som besvaras med det lösenord som du valde när programmet SYSDBGEN kördes. Du är nu inne i QL, vilket bekräftas av att du får upp kommandomarkören QL>.

Du kan nu definiera databankerna HOTELLDB och ANSTDB enligt nedanstående.

Arbetspasset inleds

run sqls

```
**************************
*                        *
*       MIMER/QL         *
*      Version 3.3.1     *
*                        *
**************************
```

Username: DONALD
Password: HEMLIGT

QL> **set** lan=swe;
QL>

Om din systemansvariga ändrat tillståndsvariabeln för språk så att svenska är standardvärde behöver du inte ge kommandot SET lan=swe.

Arbetspasset fortsätter

Definiera databanker och användare

```
QL>dba;

****************** Funktioner för databasdefinition ******************
*                                                                      *
*   1 Skapa   användare    2 Skapa   databanker    3 Skapa   access  *
*   4 Ta bort användare    5 Ta bort databanker    6 Ta bort access  *
*   7 Visa    användare    8 Visa    databanker    9 Visa    access  *
*   0 Avsluta                                                          *
************************************************************************
Ange val : 2

Skapa/omskapa databanker
- - - - - - - - - - - - - - - - -

Databank (max 8 tecken)      : HOTELLDB
Filnamn                      : HOTELLDB
Storlek (0 = befintlig fil)  : 15
Allmän access (X/D/L/P/B)    : X
<<<< Skapad >>>>

Databank (max 8 tecken)      : ANSTDB
Filnamn                      : ANSTDB
Storlek (0 = befintlig fil)  : 15
Allmän access (X/D/L/P/B)    : X
<<<< Skapad >>>>

Databank (max 8 tecken)      : PROCDB
Filnamn                      : PROCDB
Storlek (0 = befintlig fil)  : 15
Allmän access (X/D/L/P/B)    : X
<<<< Skapad >>>>

Databank (max 8 tecken)      :

****************** Funktioner för databasdefinition ******************
*                                                                      *
*   1 Skapa   användare    2 Skapa   databanker    3 Skapa   access  *
*   4 Ta bort användare    5 Ta bort databanker    6 Ta bort access  *
*   7 Visa    användare    8 Visa    databanker    9 Visa    access  *
*   0 Avsluta                                                          *
************************************************************************
Ange val : 1

Skapa/omskapa användare
- - - - - - - - - - - - - - - - -

Databank (max 8 tecken)      : SVEN
Lösenorden (max 8 tecken)    : ....
Behörighet (X/S)             : S
<<<< Skapad >>>>
Användare (max 8 tecken):
```

```
****************** Funktioner för databasdefinition ******************
*                                                                    *
*   1 Skapa   användare    2 Skapa   databanker    3 Skapa   access  *
*   4 Ta bort användare    5 Ta bort databanker    6 Ta bort access  *
*   7 Visa    användare    8 Visa    databanker    9 Visa    access  *
*   0 Avsluta                                                        *
**********************************************************************
Ange val : 0
```

Definiera tabeller

```
QL>definiera tabell HOTELL (HID är C3 : HNAMN är C20, ADRESS är C20,
+>      POSTNR är C6, POSTADR är C15, TELE är C15, STANDARD är C5,
+>      CHEF är C5, BÄDDAR är C4) i HOTELLDB;

QL>beskriv tabell HOTELL;

HOTELL    1   HID          *C   3
          2   HNAMN         C  20
          3   ADRESS        C  20
          4   POSTNR        C   6
          5   POSTADR       C  15
          6   TELE          C  15
          7   STANDARD      C   5
          8   CHEF          C   5
          9   BÄDDAR        C   4

QL>definiera tabell NÖJEN (HID är C3, VERKSAMH är C20 : STARTMÅN är C2,
+>                         SLUTMÅN är C2) i HOTELLDB;

QL>beskriv tabell NÖJEN;

NÖJEN     1   HID          *C   3
          2   VERKSAMH     *C  20
          3   STARTMÅN      C   2
          4   SLUTMÅN       C   2
          5   ÖVRIGT        C  30

QL>definiera tabell PERSONAL (HID är C3, ANSTNR är C5) i HOTELLDB;

QL>beskriv tabell PERSONAL;

PERSONAL  1   HID          *C   3
          2   ANSTNR       *C   5

QL>definiera tabell PRIS (PRISKOD är C4 : PRIS är 12) i HOTELLDB;

QL>beskriv tabell PRIS;

PRIS      1   PRISKOD      *C   4
          2   PRIS          I   2

QL>definiera tabell RUM (HID är C3, RUMSNR är C6 : TYP är C15,
+>                       PRISKOD är C4, INFO är C40) i HOTELLDB;

QL>beskriv tabell RUM;
```

```
RUM         1   HID          *C   3
            2   RUMSNR       *C   6
            3   TYP           C  15
            4   PRISKOD       C   4
            5   INFO          C  40
```

QL>**definiera tabell** ANSTÄLLD (ANSTNR **är** C5 : ANAMN **är** C20, ADRESS **är** C20,
+> POSTNR **är** C6, POSTADR **är** C15, ANKN **är** I2, ANSTDAT **är** C6,
+> BEFATTN **är** C20) **i** ANSTDB;

QL>**beskriv tabell** ANSTÄLLD;

```
ANSTÄLLD    1   ANSTNR       *C   5
            2   ANAMN         C  20
            3   ADRESS        C  20
            4   POSTNR        C   6
            5   POSTADR       C  15
            6   ANKN          C   6
            7   ANSTDAT       C   6
            8   BEFATTN       C  20
```

Radlängd : 94 *Antal rader : 0*

QL>**hjälp;**

Hjälp <kommando ! felkod>;

Visar hjälpinformation i form av förklaringar och skrivregler gällande QL-kommandon eller felkoder för DB och QL. Observera dock att kommandot "HJÄLP" inte kan användas.

Felkod : : = heltal

```
Kommando : : =  ALIAS      ! BESKRIV      ! DBA        ! DEFINIERA
                EDITERA    ! ELIMINERA    ! HJÄLP      ! HÄMTA
                KOPIERA    ! KOMMANDO     ! LAGRA      ! OMDEFINIERA
                RADERA     ! SKRIV        ! SLUTA      ! STÄNG
                SÄTT       ! UTFÖR        ! VISA       ! ÅTERSTÄLL
                ÄNDRA      ! ÖPPNA
```

Mer information om MIMER kan du få om du kör systemprocedurmenyn eller info med kommando UTFÖR SYSPROC (MENY) eller UTFÖR SYSPROC-(INFO).

QL>**hjälp** OMDEFINIERA;

OMDEFINIERA tabellnamn
LÄGG-TILL (kolumnnamn ÄR format, . . .)!
RADERA (kolumnnamn är format, . . .);

Omdefinierar en tabell. Med LÄGG-TILL läggs nya kolumndefinitioner in, med RADERA avlägsnas gamla kolumndefinitioner. Vid omdefiniering krävs att inga data finns lagrade i tabellen.

```
1
tabellnamn : : = 1–8 tecken
kolumnnamn : : = 1–8 tecken
format : : = Tn
  Tn : : = Cn,  n = 1–MIN(504,–SUM(KOLUMNLÄNGD)
           In,  n= 1–4
           Fn,  n= 3–8
```

Omdefiniera tabeller (tabellen får ej innehålla data vid OMDEFINIERA-kommando)

QL>**omdefiniera** ANSTÄLLD **lägg-till** (LÖNKOD **är** C3);

QL>**definiera** INDEX ANSTÄLLD(ANAMN);

QL>**beskriv tabell** ANSTÄLLD;

```
ANSTÄLLD 1  ANSTNR      *C  5
         2  ANAMN        C  20* Indexerad *
         3  ADRESS       C  20
         4  POSTNR       C  6
         5  POSTADR      C  15
         6  ANKN         I  2
         7  ANSTDAT      C  6
         8  BEFATTN      C  20
         9  LÖNKOD       C  3
```

QL>**definiera tabell** LÖN (LÖNKOD **är** C3 : LÖN **är** I2) **i** ANSTDB;

QL>**beskriv tabell** LÖN;

```
LÖN      1  LÖNKOD      *C  3
         2  LÖN          I  2
```

Öppna databanker för skrivning

QL>beskriv databank ANSTDB;

Databank : ANSTDB *Fil : ANSTDB*

Ställning		Storlek	
Allmän	*Användare*	*Begärd*	*Använd*
X	*X*	*15*	*2*

Tabeller

ANSTÄLLD LÖN

QL>**öppna/s** ANSTDB,HOTELLDB;

QL>**visa öppnade**;

```
HOTELLDB : S  HOTELL     NÖJEN     PERSONAL   PRIS     RUM
ANSTDV   : S  ANSTÄLLD   LÖN
```

Kopiera data från sekvensiella filer

```
QL>kopiera RUM från RUMFIL;

   14 rader lagrade i RUM

QL>kopiera/ladda NÖJEN från NÖJEN
                (2X, HID är C3, 5X, VERKSAMHET är C20, 1X, STARTMÅN är C2,
                4X, SLUTMÅN är C2, 8X, ÖVRIGT är C30);

   12 rader laddade i NÖJEN

...

QL>kopiera ANSTÄLLD från ANST;
```

Mata in data från terminalen

```
QL>lagra RUM;

HID          (C3 )  : H01
RUMSNR       (C6 )  : R101
TYP          (C15)  : ENKEL
PRISKOD      (C4 )  : P40
INFO         (C40)  : FÄRG-TV, BAD

HID          (C3 )  : !

QL>lagra NÖJEN (HID = H01, VERKSAMHET = GOLF, STARTMÅN = 05,
                SLUTMÅN = 10, ÖVRIGT = 'NIO HÅL');
```

Sökning i tabell (ta fram alla rader och kolumner i en tabell)

```
QL>hämta RUM.*;
```

HID	RUMSNR	TYP	PRISKOD	INFO
H01	R101	ENKEL	P40	FÄRG-TV, BAD
H01	R102	DUBBEL	P60	FÄRG-TV, BAD
H02	R101	DUBBEL	P60	DUSCH
H02	R102	DUBBEL	P70	FÄRG-TV, BAD, KYLSKÅP
H02	R103	DUBBEL	P80	FÄRG-TV, DUSCH, MINIBAR
H03	R101	ENKEL	P70	FÄRG-TV, DUSCH, HAVSUTSIKT
H03	R102	DUBBEL	P90	VIDEO, DUSCH, MINIBAR
H03	R103	SVIT	P90	FÄRG-TV, DUSCH, TRE RUM
H03	R104	SVIT	P90	FÄRG-TV, POOL, FYRA RUM
H04	R101	ENKEL	P10	DUSCH I KORRIDOREN
H04	R102	ENKEL	P20	DUSCH I KÄLLAREN
H05	R101	DUBBEL	P80	FÄRG-TV, MINIBAR, HAVSUTSIKT
H05	R102	SVIT	P90	TRE RUM, MINIBAR, HAVSUTSIKT
H06	R101	ENKEL	P30	DUSCH
H06	R102	ENKEL	P50	FÄRG-TV

15 rader hittade

```
QL>hämta/a RUM.PRISKOD där RUM.HID = H05 OCH
                           RUM.TYP = SVIT;
```

PRISKOD
P90

1 rad hittad

```
QL>hämta/p NÖJEN.HID,RUM.PRISKOD där NÖJEN.HID = RUM.HID
                                  och NÖJEN.VERKSAMHET = DANS
                                  eller NÖJEN.VERKSAMHET = GOLF;
```

6 rader hittade

Radera och ändra innehåll i tabell

```
QL>ändra ANSTÄLLD (ANKN = 125) där ANSTÄLLD.ANSTNR = A122;
```

1 rad ändrad

```
QL>radera RUM där RUM.TYP = SVIT;
```

3 rader raderade

```
QL>radera NÖJEN;
```

Alla rader tas bort OK? (J/N) : n

** Kommandot avbryts **

Att skapa arbetstabeller (skapa en arbetstabell med alla hotellchefer)

```
QL>hämta W1 (HOTELL.HNAMN, ANSTÄLLD.ANAMN)
            där HOTELL.CHEF = ANSTÄLLD.ANSTNR;
```

6 rader lagrade

```
QL>hämta W1.*;
```

HNAMN	*ANAMN*
DE LUXE	*GUSTAVSSON HÅKAN*
HÖRNAN	*NILSSON ANNA*
LYKTAN	*HOLGERSSON HARALD*
PLAZA	*JOHANSSON GUNNAR*
REDCAP	*HOLGERSSON HARALD*
STANDARD	*KARLSSON ELIN*

6 rader hittade

Procedurhantering

```
QL>utför SYSPROC(MENU);

1. Definiera ett procedur-bibliotek
2. Eliminera ett procedur-bibliotek
3. Ladda ett procedur-bibliotek från en fil
4. Skriva ett procedur-bibliotek till en fil
5. Visa nyheter och information
6. Åter till kommandonivå

? 1

Namn på procedur-bibliotek : TEST

1. Definiera ett procedur-bibliotek
2. Eliminera ett procedur-bibliotek
3. Ladda ett procedur-bibliotek från en fil
4. Skriva ett procedur-bibliotek till en fil
5. Visa nyheter och information
6. Åter till kommandonivå

? 6

QL>editera TEST(meny);

ED>!
  10    +Skriv  'Meny för listprocedurer'
  20    +Skriv;
  30    Öppna/s  anstdb,testdb;
  40    +Om  &*fel='0' så+hoppa 100;
  50
  60    +Kommentar      Kontrollera om databankerna gick att öppna;
  70
  80    +Skriv 'Databankerna kunde ej öppnas, försök igen!';
  90    +Stopp
 100    Sätt  Sbredd=80, Slängd=40, Sra=2, ejutskrift;
 110    Alias  hotell(h),nöjen(n),rum(r),pris(p);
 120    +Kommentar      Kortnamn för tabeller
 130    +Meny  list1 'Lista över hotell i viss ort';
 140    +Meny  list2 'Lista över hotell i viss prisklass';
 150    +Meny  list3 'Lista över hotell i med viss verksamhet';
 160    !

ED>Aktuell list1
Aktuell procedur TEST(LIST1) i procdb

ED>!
  10    +Sät  skrivfil=orter;
  20    +Skriv  'Listrutin för vilka hotell som finnes på en viss ort';
  30    +Skriv
  40    +Skriv
  50    +Tilldela  &ort/Versaler=+Fråga 'Ange ort:';
  60    +Hämta  w1 (h.hnamn, h.adress)
  70                där h.postadr=&ort;
```

```
 80    +Om  &*antal=0 så hoppa 150;
 90    Skriv  W1 'Lista över hotell belägna i &ort';
100    +Tilldela  &Svar=+Fråga/Versaler
110    'Vill du se resultatet på skärmen (J/N)';
120    +Om  &Svar='J' så hämta w1.*;
130    Eliminera  W1;
140    +Hoppa  50;
150    +Skriv  'Inget MIMER-hotell finns i &ort';
160    +Hoppa  50;
170    !
```

ED>**Aktuell** list2
Aktuell procedur TEST(LIST2) i procdb

```
ED>!
 10    +Sät  skrivfil=priser;
 20    +Skriv  'Listrutin för vilka hotellpriser';
 30    +Skriv
 40    +Skriv
 50    +Tilldela  &pris=+Fråga 'Ange maxpris:';
 60    +Hämta  w1 (h.hnamn, h.adress, p.pris, r.typ)
 70                där r.priskod=p.priskod och h.hid=och p.pris<=&pris;
 80    +Om  &*antal=0 så+hoppa 150;
 90    Skriv W1 'Lista över hotell med rumspriser mindre än &pris';
100    +Tilldela  &Svar=+Fråga/Versaler
110    'Vill du se resultatet på skärmen (J/N)';
120    +Om  &Svar='J' så hämta w1.*;
130    Eliminiera  w1;
140    +Stopp;
150    +Skriv  'Inget hotell i angiven prisklass finns';
160    !
```

ED>**Aktuell** list3
Aktuell procedur TEST(LIST3) i procdb

```
ED>!
 10    +Sät  skrivfil=verksamhet;
 20    +Skriv  'Listrutin för vilka nöjen olika hotel erbjuder';
 30    +Skriv
 40    +Skriv;
 50    +Tilldela  &nöje=+Fråga 'Ange önskad nöjesverksamhet:';
 60    +Hämta  w1 (nöjen.*) där nöjen.verksamh='&nöjen';
 70    +Om  &*antal=0 så+Hoppa 110;
 80    +Skriv  W1 'Lista över hotell med &nöje';
 90    Eliminera  w1;
100    +Stopp!
110    +Skriv  'Inget hotell med önskad verksamhet hittades';
120    +Hoppa  50;
130    !!
```

```
QL>EDI TEST(PIPE);
ED>!
 10    +Kommentar      Procedur som ökar alla rumspris med en;
 20    +Kommentar      inläst procentsats med hjälp av;
 30    +Kommentar      radvis åtkomst;
 40    Öppna/s  anstdb;
```

```
  50    Alias pris(p);
  60    +Tilldela &ökning=+Fråga 'Ange procentuell prisökning'
  70    +Inled pris hämta (&pris=p.pris,&kod=p.priskod);
  80    +Kommentar      Hämta en rad och bearbeta den
  90    +Kommentar      Spar även primärnyckel för ändring
 100    +Nästa rum;
 110    +Om &*antal=0 så+Hoppa 160;
 120    +Tilldela &pris=&pris *(1+&ökning);
 130    +Ändra p (pris=&pris) där p.priskd=&kod;
 140    +Kommentar      Tag nästa rad i tabellen
 150    +Hoppa 100;
 160    +Avsluta rum;
 170    !!
QL>utför TEST(MENU);
```

Meny för listprocedurer

1. *Lista över hotell i viss ort*
2. *Lista över hotell i viss prisklass*
3. *Lista över hotell med viss verksamhet*

? 1

Ange ort : Uppsala

Vill du se resultatet på skärmen (J/N) : j

HNAMN	*ADRESS*
LYKTAN	*STORGATAN 45*
REDCAP	*KUNGSGATAN 20*

2 rader hittade

Ange ort : !

```
QL>sluta;

********** MIMER/QL *********
*                           *
*        Körningstid 32 min         *
*                           *
********** MIMER/DB *********
```

Litteratur

Nedanstående litteraturtips är i första hand avsedda för den som behöver skaffa sig mer grundkunskaper om ADB i allmänhet och/eller om databassystem.

Datalära. Ett studiematerial på grundläggande nivå utgivet av Riksdataförbundet, bl a bestående av ett 20-tal mindre häften i olika delämnen.

Dyrke, Roland — Gratte, Ingvar — Wrede, Rabbe: *Grunderna i ADB — en praktisk introduktion till administrativa datasystem.* Lund 1982.

Renås, Rune: *ADB för alla.* Stockholm 1982.

ADB på jobbet. Kurspaket från SIPU, statens institut för personaladministration och -utbildning. Stockholm 1983.

Sundgren, Bo: *Databaser och datamodeller.* Lund 1981. En relativt kortfattad men innehållsrik första översikt över databassystem.

Date, C J: *An Introduction to Database Systems.* Reading, Mass., 1981. Mer omfattande introduktion till databasområdet.

Några grundläggande dataord

Nedanstående ordlista är avsedd för den som har ingen eller mycket liten tidigare bekantskap med ADB och är till för att underlätta läsningen av handboken och förståelsen av det språk som talas av ADB-folk. En del bland datafolk mycket vanliga engelska termer har tagits med.

Ordlistan gör inte anspråk på fullständighet.

Access

Åtkomst. Används t ex om åtkomst av data lagrade i yttre minnen (sekundärminnen). Olika användare kan tilldelas olika accessrätt, dvs få tillgång till mer eller mindre begränsade delar av data. Accesstid eller söktid är den tid som åtgår för att söka reda på och presentera data från ett minne.

Alfanumeriska tecken

Bokstäver och siffror.

Anropa

Beordra fram och köra en viss programrutin. Man säger t ex att ett kommando anropar en bestämd programrutin eller att en användare anropar en speciell programvara.

Applikationsprogram

Program avsett för tillämpning inom något verksamhetsområde, t ex redovisning eller ordbehandling, till skillnad från t ex systemprogram eller hjälpprogram, som används för den interna hanteringen i ett datorsystem (kopiering av filer, omnumrering av programrader, administration av utskrifter osv).

Area

Minnesutrymme.

ASCII

Varje tecken har en s k ASCII-kod, ett genom konvention bestämt talvärde, som motsvarar tecknet. EBCDIC är ett annat kodsystem av samma slag.

Back-up

Vanligt ord för reserv, t ex reservkopia av datamedium, reservminne, reservutrustning.

Bandminne
Yttre minne för långtidslagring av data där lagringen sker på ett magnetband.

Bibliotek
Uppsättning. Vanlig term för en samling program eller data.

Binär
I det binära talsystemet utttrycks alla tal med hjälp av de två siffrorna 0 och 1. Talet 1 blir 1, talet 2 blir 10, talet 3 blir 11, talet 4 blir 100 osv. Binära tal används internt i datorsystem för att representera olika tal och tecken. Att det binära talsystemet används i datasammanhang beror på att dess två siffror har en enkel fysisk motsvarighet i datorns interna elektriska strömmar ("ström på", "ström av").

Bit
Förkortning av Binary Digit, binär siffra, dvs 0 eller 1.

Buffert
Ett minnesutrymme som används för tillfällig lagring av data.

Byte
Beteckning för en grupp bitar, oftast 8 stycken.

Centralenhet
Se CPU.

CPU
Datorns centralenhet, som normalt består av behandlingsenhet, styrenhet och primärminne. CPU-tid är den tid som CPU tas i anspråk under ett arbetspass.

Definiera
Vanligt uttryck med innebörden att användaren bestämmer hur något skall se ut eller fungera. Man definierar t ex en skärmbild, dvs bestämmer var olika ledtexter, rubriker, inmatningsfält etc skall placeras.

Datorsystem
Kombination av dator, kringutrustning och operativsystem.

Defaultvärde
Detsamma som standardvärde.

Disk
Detsamma som skivminne.

Diskett
En liten billig skiva för långtidslagring av data, kan raderas och återanvändas. Placeras i en diskettstation vid läsning eller skrivning.

EBCDIC-kod
Se ASCII-kod.

Editera
Redigera, dvs ändra om, i inskrivna program, programrader, texter etc. Editering utförs med en programvara kallad editor. Många olika editorer finns i marknaden.

Editor
Se editera.

Eko
Det datorsystemet skriver ut på skärmen som resultat av användarens inmatning. Man säger att systemet ekar tillbaka något.

Fil
Data som användaren bestämt skall höra ihop med varandra, t ex en fil med uppgifter om personers namn och telefonnummer. I en sekvensiell fil lagras data efter varandra.

Format
Ofta ungefär detsamma som form eller typ, ibland ungefär detsamma som skrivsätt eller uttryckssätt. Med t ex heltalsformat avses en viss datatyp, nämligen heltal. Formatera betyder oftast redigera eller utforma, t ex bestämma radlängd och sidhöjd på en utskrift.

Gränssnitt
Ingångs- eller övergångsmodul i form av hård- eller mjukvara. Ett gränssnitt kan t ex vara en uppsättning programrutiner som gör att en programvara kan nås från en annan.

Hjälpprogram
Se applikationsprogram.

Hårdvara
Se maskinvara.

Inenhet
Anordning för inmatning av data, t ex tangentbord, optisk läsare, bandstation och diskettläsare.

Inmatningsmarkör
Ett eller flera tecken som systemet visar på skärmen och som anger att användaren skall mata in något. Exempel: QL>, som anger att MIMER/QL väntar sig ett QL-kommando.

Instruktion
I allmänhet detsamma som kommando.

Integrering

Avser i datasammanhang oftast att enkla övergångar finns mellan skilda programvaror i ett system, att data lagrats på ett sådant sätt att de meningsfullt kan bearbetas av de olika programvarorna i systemet och att det inte finns några tekniska hinder för åtkomst av program och data från alla terminaler i systemet.

Interaktiv

Interaktiv databehandling äger rum när användaren direkt vid bildskärmen svarar på systemets frågor, matar in data osv och omgående får tillbaka svar, bearbetningsresultat osv, dvs för ett slags dialog med datorn.

Interface

Vanligt ord för gränssnitt.

Kilobyte

Se minne.

Kringutrustning

Yttre minnen, skrivare och andra enheter som kan anslutas till en dators centralenhet.

Kommando

En uppmaning till ett datorsystem att utföra en viss uppgift. Uppmaningen skall vara formulerad i ett språk som systemet kan tolka. Kommandot utlöser ett antal interna, förutbestämda operationer som leder till avsett resultat.

Manipulera

Att manipulera data är ett vanligt sammanfattande uttryck för operationer som innebär att data sorteras om, kompletteras, ändras, raderas o dyl.

Maskinvara

Den fysiska datorn och dess kringutrustning, själva apparaterna och dessas komponenter. En ungefärlig synonym är hårdvara. Jämför programvara.

Meny

En skärmbild som visar ett antal valmöjligheter för användaren.

Miljö

Mångtydigt ord i datasammanhang. Man talar t ex om stordatormiljö (system där en väsentlig del av ADB-arbetet bygger på en eller flera stora datorer), COBOL-miljö (system där programmeringen sker i COBOL, kan även beteckna språkbruket i COBOL) och terminalmiljö (system där arbetet i stor utsträckning sker interaktivt från terminaler). Miljö kan även avse den skräddarsydda utformning ett system får hos en viss användare, t ex valfritt definierade kommandoord, speciella skärmbilder o dyl.

Minne

Används vanligen om det fysiska lagringsmediet för data. Se vidare primärminne och sekundärminne. Minnesstorlek mäts ofta i kilobyte (1024 byte, förkortas K) eller megabyte (1000 kilobyte).

Mjukvara
Se programvara.

Odefinierat värde
Innebär att värde saknas, vilket normalt inte uppfattas som 0 eller som blankt.

Operativsystem
En övergripande programvara som svarar för att ett datorprogram utförs riktigt, att nödvändiga interna minnesutrymmen finns tillgängliga, att data flyttas mellan primär- och sekundärminne m m. En dator fungerar inte utan operativsystem.

Parameter
Ungefär detsamma som variabel. En parameter kan ofta sättas till ett visst värde av användaren, t ex QL-parametern SL som anger sidlängd vid utskrift på listor.

Portabilitet
Möjligheten att flytta programvaror och datamängder från ett datorsystem till ett annat, något som normalt inte är möjligt utan omfattande arbete (omprogrammering, ny inmatning av data). MIMER-systemet är ett exempel på portabla programvaror.

Primärminne
Det centrala minnet i en dator, det minne som används för lagring av den programvara som just används och de data som just behandlas.

Printer
Detsamma som skrivare.

Program
En serie instruktioner eller kommandon som talar om vad datorsystemet skall göra, t ex skriva ut en uppmaning till användaren att mata in ett värde, lagra värdet, utföra en beräkning där värdet ingår, skriva ut resultatet på skärmen osv. Programmet skrivs alltid i ett språk som datorsystemet är förmöget att omtolka till olika meningsfulla operationer. Det finns ett flertal olika programspråk och varianter av programspråk. Exempel: COBOL, FORTRAN, BASIC, Pascal, ADA.

Programvara
Programmet eller programmen som används i ett datorsystem, till skillnad från maskinvaran. Kallas även mjukvara till skillnad från hårdvara. Programvaran laddas in i systemet från t ex ett band eller en diskett, antingen vid varje användning eller permanent. Observera dock att viss programvara, t ex operativsystem, kan föreligga i hårdvaruform, fysiskt inbyggd i systemet.

Prompt(er)
Se inmatningsmarkör.

Sekundärminne

Ett minne i form av t ex ett skivminne, ett diskettminne eller ett magnetbandsminne, där data lagras när de inte är under direkt bearbetning. Sekundärminnet har till skillnad från primärminnet stor lagringskapacitet, men det tar längre tid att hämta data i sekundärminnet.

Session

Vanligt ord för arbetspass.

Skivminne

Ett vanligt yttre minne hos stora och medelstora datorer. Har stor lagringskapacitet och korta söktider.

Skrivare

Enhet som gör utskrifter av data.

Slutanvändare

Användare som huvudsakligen kör applikationsprogram och oftast inte behöver ha kunskaper i programmering eller ADB-teknik.

Standardvärde

Ett variabelvärde som lagts in i systemet vid leveransen och som kan ändras.

Sträng

En följd av alfanumeriska tecken.

Stänga

Se öppna.

Subrutin

En del av ett dataprogram som utför en bestämd uppgift och som kan anropas flera gånger under en programkörning från olika ställen i programmet.

Syntax

Regler för sammanfogandet av ett språks beståndsdelar till meningsfulla satser. I ett konstruerat språk som ett dataprogrammeringsspråk måste syntaxreglerna följas exakt för att satserna skall kunna förstås av systemet. Syntax kallas också en mönstersats som ger en bild av hur en viss typ av satser formuleras i ett språk.

Systemprogram

Se applikationsprogram.

Sätta, sättning

Vanligt uttryck i samband med angivande av variabelvärden. Man sätter t ex tillståndsvariabeln SL i MIMER/QL till 15 vilket ger en sidlängd på 15 rader.

Söktid

Se access.

Uppdatera
Ändra eller komplettera lagrade data.

Utenhet
Anordning för utmatning av data, t ex bildskärm, skrivare, diskettstation och bandstation.

Utility
Detsamma som hjälpprogram (se applikationsprogram).

Variabel
En del av ett kommando, ett aritmetiskt uttryck el dyl som kan anta olika värden, vilka tilldelas antingen av datorsystemet eller av användaren.

Värde
Ett tal eller en sträng.

Yttre minne
Se sekundärminne.

Öppna
För att användaren skall kunna läsa eller påverka innehållet i en fil, en MIMER-tabell eller en MIMER-databank måste den först öppnas, dvs göras tillgänglig, med ett särskilt kommando. När användaren inte längre behöver tillgång till filen etc stängs denna med ett annat kommando.

Index

Hänvisningar avser sidnummer. Ingen hänvisning förekommer till bilagorna. Kommandon och variabler skrivs med stora bokstäver. Att ett begrepp avser ett kommando framgår genom att kommandotyp anges inom parentes. Kommandotyperna betecknas QL (QL-kommandon för spontan frågeverksamhet), proc (procedurkommandon) och ed (kommandon i procedureditorn).