IDA BAZA

Uporabniški priročnik

Ident: 83073044

IDA BAZA

Uporabniški priročnik

Ident: 83073044

Prva izdaja
Oktober 1987

ISKRA DELTA

KAZALO

Poglavje 1 Uvod v IDA Bazo _____ 1-1

1.1 Pojem baze podatkov _____ 1-1

1.1.1 Prednosti sistema baze podatkov _____ 1-2

1.2 Logična struktura baze podatkov _____ 1-3

1.2.1 Shema _____ 1-3
1.2.2 Podshema _____ 1-8
1.2.3 Vrste zapisov v IDA Bazi _____ 1-8

1.3 Podatkovni prostor _____ 1-10

1.4 Dostop do podatkov _____ 1-12

1.4.1 Večuporabniški režim _____ 1-12
1.4.2 Operativno področje _____ 1-13

1.5 Beleženje, obnovitev baze _____ 1-13

1.6 Zaščita podatkov _____ 1-14

1.7 Programski pristop do baze podatkov _____ 1-14

1.8 Označeni zapisi _____ 1-14

Poglavje 2 DDC V1.1-Program za opis baze podatkov 2-1

2.1 Aktiviranje DDC prevajalnika 2-4

2.1.1 Aktiviranje DDC prevajalnika-verzija DELTA/V 2-4
2.1.2 Aktiviranje DDC prevajalnika-verzija DELTA/M 2-7
2.1.3 Poročilo o napakah pri prevajanju 2-9

2.2 Elementi DDC jezika 2-9

2.2.1 Primer opisa sheme 2-10
2.2.2 Primer opisa logične strukture 2-14
2.2.3 Primer opisa fizične strukture 2-16
2.2.4 Primer opisa operativnega področja 2-17
2.2.5 Kreiranje podsheme je sestavljeno iz 2-18
2.2.6 Primer opisa logične strukture podsheme 2-21
2.2.7 Primer opisa sekvenčne datoteke 2-21

Poglavje 3 Manipulacija podatkov v IDA Bazi 3-1

3.1 Uvod 3-1

3.1.1 Nadrejeni zapisi 3-1
3.1.2 Podrejeni zapisi 3-1
3.1.3 Zaklepanje zapisov 3-2
3.1.4 Zaščita podatkov s pomočjo gesla 3-4

3.2 Ukazi in funkcije manipulacijskega jezika IDA Baze 3-5

3.2.1 DML ukazi 3-5
3.2.2 DBMIO funkcije 3-5
3.2.3 SEQIO funkcije 3-6
3.2.4 Splošni napotki 3-7

3.3 DML klicni stavki 3-15

3.3.1 Začetek in konec dela s podatkovno bazo 3-15

3.3.1.1 HELLO 3-15
3.3.1.2 BYE 3-15

3.3.2 DML ukazi 3-16

3.3.2.1 CANCEL 3-16
3.3.2.2 COMMIT 3-17
3.3.2.3 DBMIO 3-17
3.3.2.4 LOGDAT 3-18
3.3.2.5 SEQIO 3-19

3.3.3 DBMIO funkcije ____ 3-19

3.3.3.1 DELG ____ 3-19
3.3.3.2 GETD ____ 3-20
3.3.3.3 GETG ____ 3-21
3.3.3.4 GETP ____ 3-22
3.3.3.5 GETR ____ 3-23
3.3.3.6 INSA ____ 3-25
3.3.3.7 INSB ____ 3-26
3.3.3.8 INSG ____ 3-27
3.3.3.9 RWRG ____ 3-28

3.3.4 SEQIO funkcije ____ 3-29

3.3.4.1 SCLO ____ 3-29
3.3.4.2 SGET ____ 3-29
3.3.4.3 SINS ____ 3-30
3.3.4.4 SOPE ____ 3-30
3.3.4.5 SRWD ____ 3-31
3.3.4.6 SRWR ____ 3-32

Poglavje 4 Program za formatiranje baze podatkov ____ 4-1

4.1 Uvod ____ 4-1

4.2 Izvajanje programa za formatiranje baze podatkov ____ 4-2

4.3 Primer razširitve podrejenega zapisa ____ 4-3

Poglavje 5 Program za kontrolo dela baze podatkov ____ 5-1

5.1 Uvod ____ 5-1

5.2 Izvajanje programa za kontrolo dela baze podatkov ____ 5-1

5.2.1 Aktiviranje baze podatkov ____ 5-2
5.2.2 Zaustavljanje baze podatkov ____ 5-2
5.2.3 Kontrola delovanja baze podatkov ____ 5-3

Poglavje 6 Beleženje sprememb in obnova baze podatkov ____ 6-1

6.1 Uvod ____ 6-1

6.2 Beleženje in obnova logičnih transakcij ____ 6-2

6.3 Beleženje funkcij in obnova baze podatkov ____ 6-2

6.4 Uporaba DBREStore programa ____ 6-3

Poglavje 7 Pomožni programi baze podatkov 7-1

7.1 Uvod 7-1

7.2 DBGet - prepis zapisov v sekvenčne datoteke 7-2

7.3 DBPut - polnjenje zbirke zapisov 7-3

7.4 DBDel - brisanje zapisov baze podatkov 7-5

7.5 Reorganizacija baze podatkov 7-6

Poglavje 8 Optimiziranje baze podatkov 8-1

8.1 Uvod 8-1

8.2 Oblikovanje V/I področij 8-1

8.3 Oblikovaje operativnih področij 8-2

8.4 Oblikovanje fizične strukture 8-3

8.5 Oblikovanje logičnih blokov 8-3

8.6 Oblikovanje velikosti zbirk nadrejenih zapisov 8-5

8.7 Optimizacija logičnih transakcij 8-5

Dodatek A Kaj je REFORMAT in kaj je RAZŠIRITEV ... A-1

Dodatek B Ukazi in funkcije jezika za upravljanje s podatki B-1

Dodatek C DML sporočila C-1

Dodatek D Primer programa D-1

1. UVOD V IDA BAZO

1.1. POJEM BAZE PODATKOV

Beseda **podatki** se nanaša na skupek neobdelanih dejstev in številk. **Informacija** je del, ki je izvlečen iz skupka **podatkov** in obdelan ter prikazan za določen namen. Iz istih podatkov je mogoče dobiti zelo različne **informacije.**

Na začetku uporabe računalnikov za shranjevanje in obdelavo podatkov so se kot nosilci podatkov uporabljale luknjane kartice in magnetni trakovi, kjer so bili podatki dosegljivi le zaporedno. Izgledi posameznih kartic in zapisov so bili definirani v programih. Povezave med različnimi karticami ali zapisi so bile simulirane tako, da so se podatki na večih karticah ali zapisih ponovili. Podatki so bili grupirani za določen namen (program) in če se je pojavila potreba, da se nekateri že obstoječi podatki uporabijo za drug namen, jih je bilo potrebno ponovno pripraviti (sortiranje, ponovno luknjanje, (redundanca), kar je prinašalo velike težave pri vzdrževanju. Enak način shranjevanja in vzdrževanja podatkov se je ohranil do danes, čeprav so se zmogljivosti računalnikov in zunanjih pomnilnikov skokovito povečevale. Zaradi povečanih zmogljivosti računalnikov in zunanjih pomnilnikov, se je nujno povečalo tudi število in raznolikost podatkov in odnosov med njimi. Verjetnost napak in neažurnosti podatkov se je povečevala še hitreje, kot je naraščalo število podatkov. Obstoječe tehnike za upravljanje podatkov so postale nezadostne.

Zaradi tega so se razvile podatkovne baze, ki centralizirajo skrbništvo nad podatki in do določene mere omogočajo neodvisnost programov od podatkov.

1.1.1. PREDNOSTI SISTEMA BAZE PODATKOV

Večina organizacij je spoznala, da so podatki zelo pomembni in dragoceni, če so pravilni in ažurni. Zato morajo biti skrbno organizirani in vedno razpoložljivi, dostop do njih pa mora biti hiter. IDA Baza je programsko orodje, ki to omogoča.

IDA Baza je programsko orodje za organizacijo in manipulacijo velikega števila podatkov, ki omogoča vzporedno obdelovanje istih ali različnih podatkov. Na ta način postanejo podatki skupna last cele organizacije in kot taki predstavljajo veliko vrednost, zato mora biti dostop do njih skrbno kontroliran.

IDA Baza omogoča tako zaščito z mehanizmom varovalnih GESEL in "pristopnih pravic" do nivoja posameznega podatka. Da pri vzporednem dostopu različnih uporabnikov do istih podatkov ne bi prišlo do konfliktnih situacij, ima IDA Baza vgrajen mehanizem ZAKLEPANJA in ČASOVNE OMEJITVE, ki zagotavlja, da v kolikor že pride do konflikta se le-ta po preteku ČASOVNE OMEJITVE razreši.

IDA Baza zagotavlja neodvisnost programov od podatkov na podlagi logičnih opisov podatkov, ki se nahajajo v PODSHEMI. Programer se lahko osredotoči na problem, ki ga mora rešiti in se ne ozira na to, kako in kje so podatki shranjeni. Za to je odgovoren skrbnik podatkovne baze. Novi podatki in povezave se lahko dodajajo, ne da bi to zahtevalo spremembe že obstoječih programov. Programi se lahko spreminjajo, ne da bi to vplivalo na podatkovno bazo. To omogoča organsko rast baze podatkov in vključevanje vedno novih aplikacij.

IDA Baza zagotavlja varnost podatkov v slučaju napak na aparaturni opremi z LOGIRANJEM TRANSAKCIJ in/ali LOGIRANJEM FUNKCIJ.

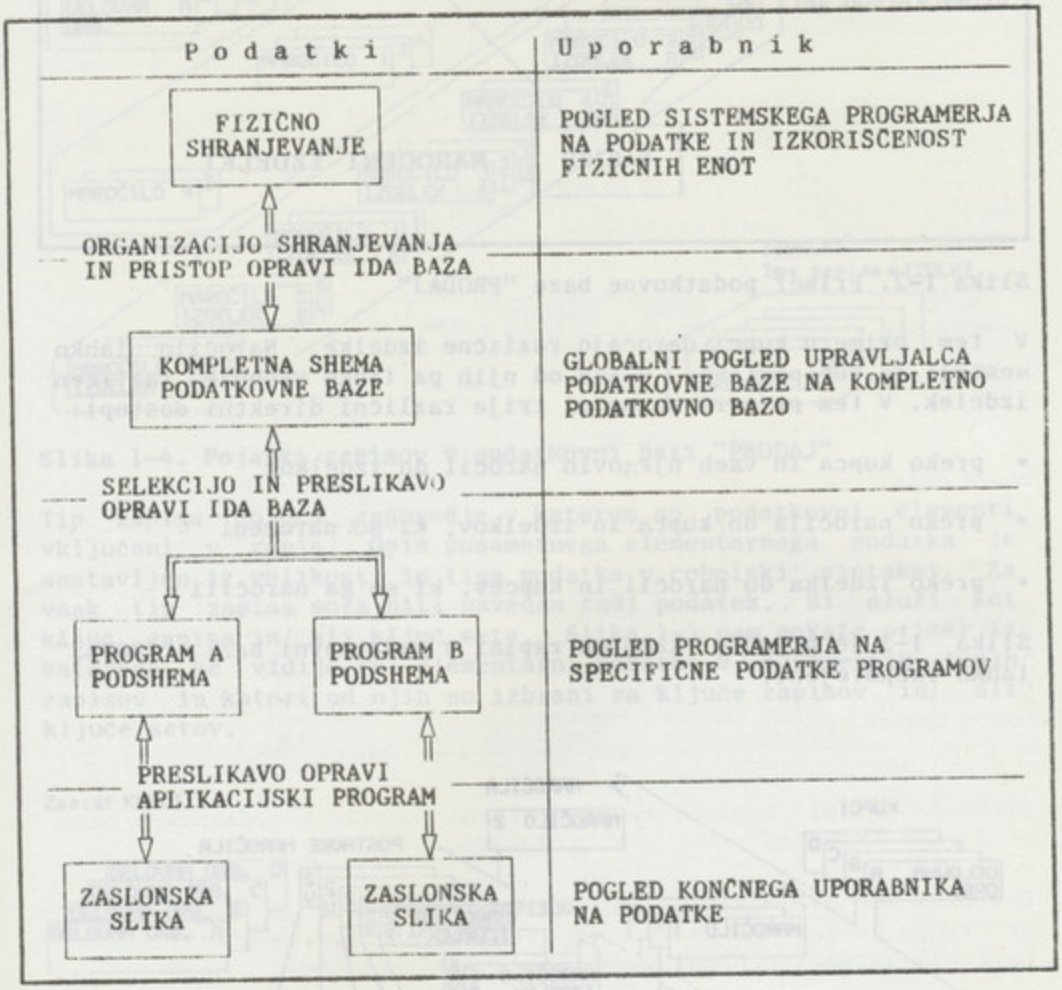
Logična zasnova IDA Baze omogoča definiranje hierarhičnih in mrežnih povezav in s tem približevanje logičnega modela podatkov realnemu svetu. Jezik za upravljanje s podatki - DML - je zasnovan na štirih osnovih funkcijah, ki so neodvisne od tipa zapisa.

Sestavni del IDA Baze so tudi servisni programi, ki omogočajo enostavno in hitro vzdrževanje podatkov.

1.2. LOGIČNA STRUKTURA PODATKOVNE BAZE

Elementarni podatki so lahko predstavljeni, shranjeni in dostopni računalniku na različne načine. Podatki morajo biti definirani tako, kot to uporabnik želi. Sistem za delo s podatkovno bazo pa jih mora prevesti v fizično strukturo, ki omogoča najefektnejši dostop.

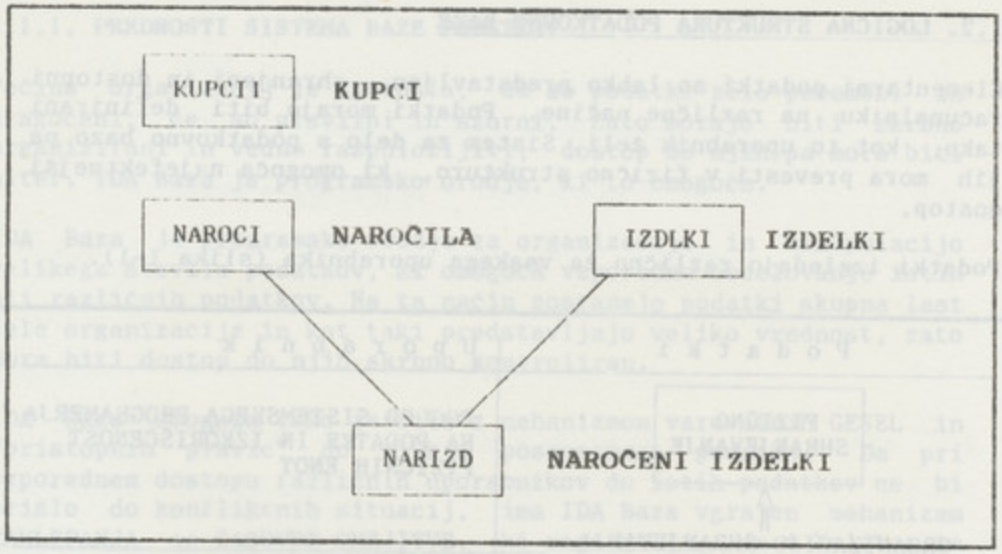
Podatki izgledajo različno za vsakega uporabnika (slika 1-1).



Slika 1-1. Različni pogledi na podatke

1.2.1. SHEMA

Shema je globalna logična definicija vseh vrst podatkov v podatkovni bazi, ne glede na njihovo fizično predstavitev. Povezave med različnimi zapisi v shemi so lahko zelo kompleksne. Za nazorno predstavitev služi primer na sliki 1-2.

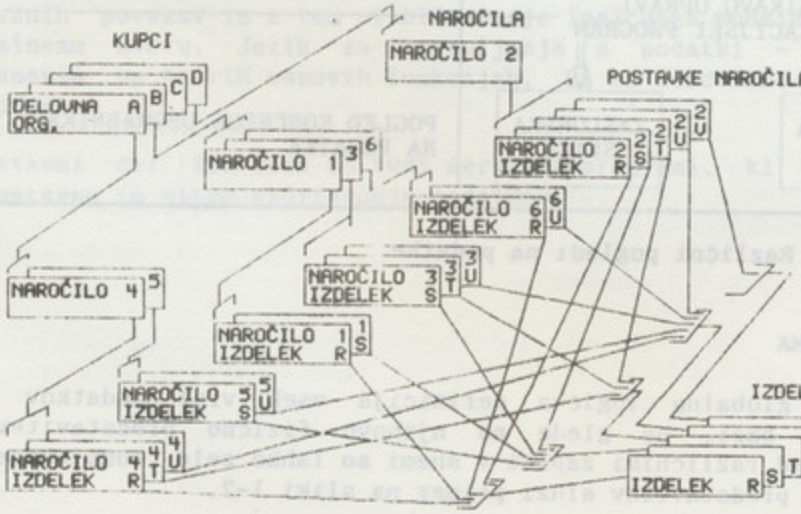


Slika 1-2. Primer podatkovne baze "PRODAJ"

V tem primeru kupci naročajo različne izdelke. Naročilo lahko sestoji iz več postavk, vsaka od njih pa lahko vsebuje različen izdelek. V tem primeru so možni trije različni direktni dostopi:

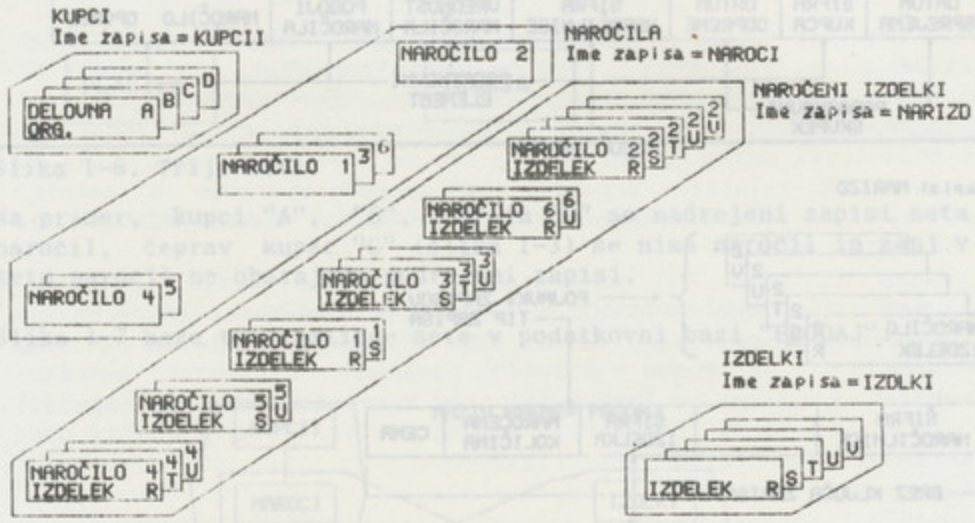
- preko kupca in vseh njegovih naročil do izdelkov
- preko naročila do kupca in izdelkov, ki so naročeni
- preko izdelka do naročil in kupcev, ki so ga naročili

Slika 1-3 je primer, kako so zapisi v podatkovni bazi "PRODAJ" lahko razporejeni.



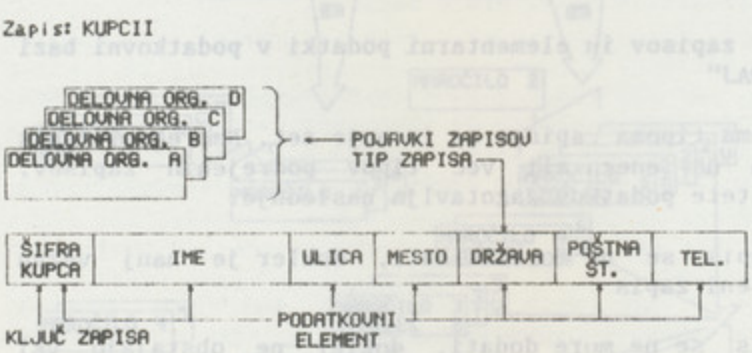
Slika 1-3. Povezave med pojavki zapisov v podatkovni bazi "PRODAJ"

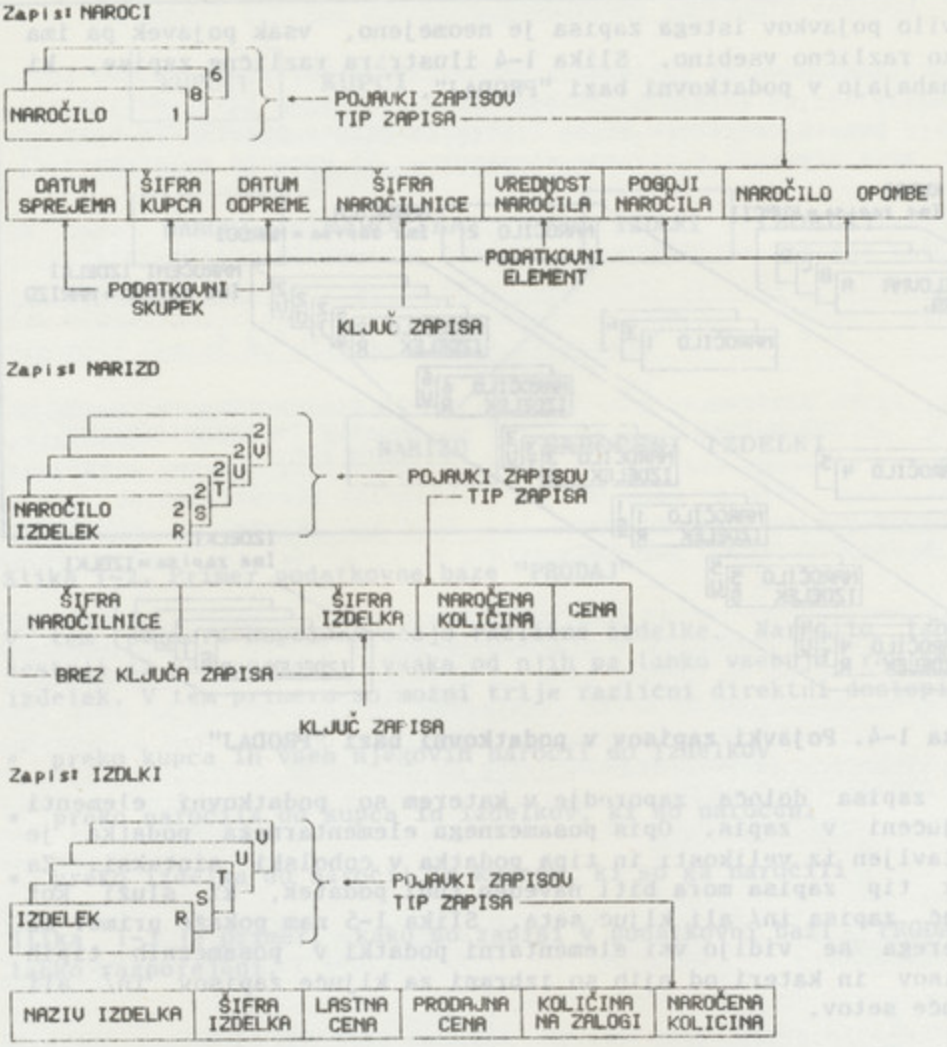
Stevilo pojavkov istega zapisa je neomejeno, vsak pojavek pa ima lahko različno vsebino. Slika 1-4 ilustrira različne zapise, ki se nahajajo v podatkovni bazi "PRODAJ".



Slika 1-4. Pojavki zapisov v podatkovni bazi "PRODAJ"

Tip zapisa določa zaporedje v katerem so podatkovni elementi vključeni v zapis. Opis posameznega elementarnega podatka je sestavljen iz velikosti in tipa podatka v cobolski sintaksi. Za vsak tip zapisa mora biti naveden tudi podatek, ki služi kot ključ zapisa in/ ali ključ seta. Slika 1-5 nam pokaže primer iz katerega se vidijo vsi elementarni podatki v posameznih tipih zapisov in kateri od njih so izbrani za ključe zapisov in/ ali ključe setov.



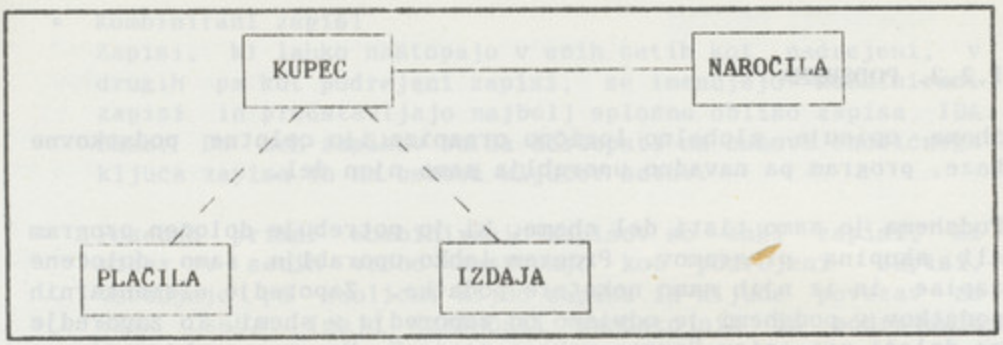


Slika 1-5. Tipi zapisov in elementarni podatki v podatkovni bazi "PRODAJ"

Povezava med dvema tipoma zapisov se imenuje set. Nadrejeni zapis je lahko vezan na enega ali več tipov podrejenih zapisov. Kontrola integritete podatkov zagotavlja naslednje:

- nadrejeni zapis se ne more brisati, dokler je nanj vezan kakšen podrejeni zapis
- odvisni zapis se ne more dodati, dokler ne obstajajo vsi njegovi nadrejeni zapisi

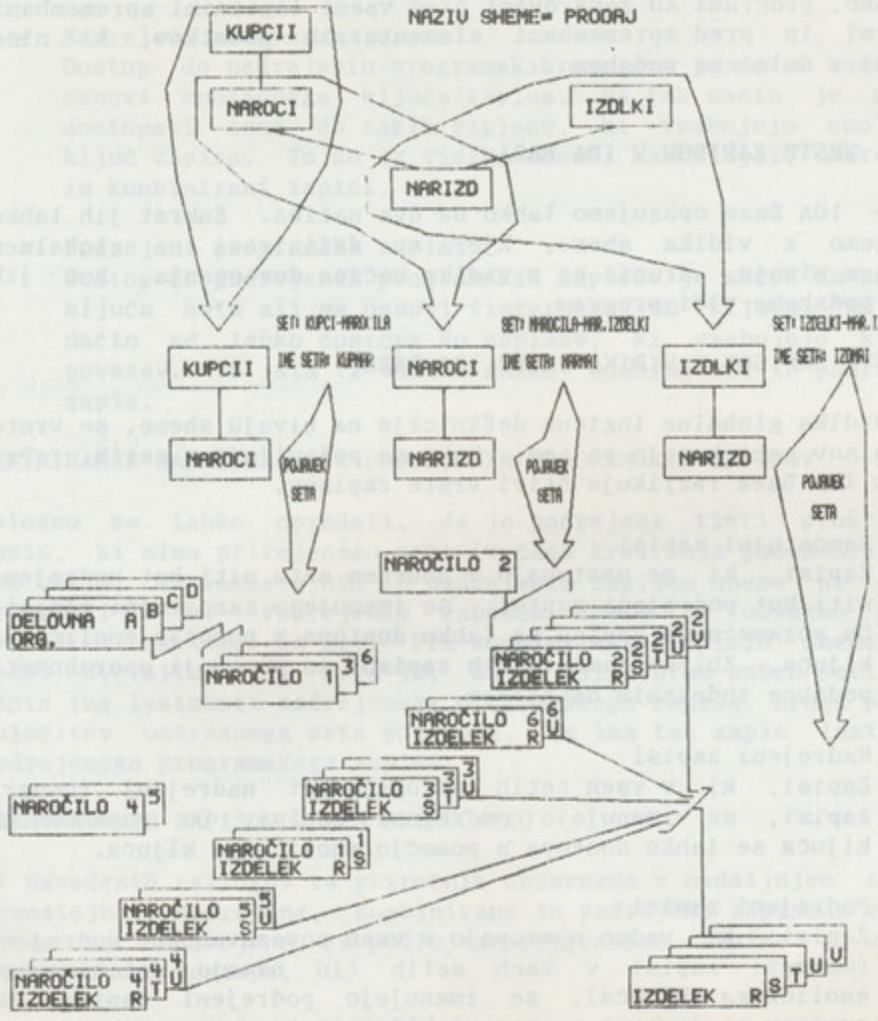
Slika 1-6 nam kaže takšno povezavo, v kateri so definirani štirje tipi zapisov (KUPEC, NAROČI, IZDAJA, PLAČILO) in trije seti.



Slika 1-6. Trije seti

Na primer, kupci "A", "B", "C" in "D" so nadrejeni zapisi seta naročil, čeprav kupec "C" (slika 1-3) še nima naročil in zanj v setu naročil ne obstajajo podrejeni zapisi.

Slika 1-7 kaže tri različne sete v podatkovni bazi "PRODAJ".



Slika 1-7. Seti in pojavki setov

1.2.2. PODSHEMA

Shema opisuje globalno logično organizacijo celotne podatkovne baze, program pa navadno uporablja samo njen del.

Podshema je samo tisti del sheme, ki jo potrebuje določen program ali skupina programov. Program lahko uporablja samo določene zapise in iz njih samo nekatere podatke. Zaporedje elementarnih podatkov v podshemi je odvisno od zaporedja v shemi. To zaporedje se določi pri opisu "programskega zapisa". Na programski zapis so vezane tudi pristopne pravice, kar omogoča kontrolo in zaščito podatkov do nivoja elementarnega podatka. Vsaka podshema je zaščitena z GESLOM.

Spremembe v shemi vplivajo na spremembe v podshemi in s tem na spremembe programov samo takrat, ko se spremeni dolžina ali tip podatka, ki je izbran v določeno podshemo. Z drugimi besedami povedano, programi so zavarovani pred vsemi logičnimi spremembami v shemi in pred spremembami elementarnih podatkov, ki niso izbrani v določeno podshemo.

1.2.3. VRSTE ZAPISOV V IDA BAZI

Zapise IDA Baze opazujemo lahko na dva načina. Enkrat jih lahko opisujemo z vidika sheme, kjer so definirani na globalnem logičnem nivoju, drugič pa z vidika načina dostopanja, kot jih skozi podsheme vidi program.

* VRSTE ZAPISOV Z VIDIKA SHEME IDA BAZE

  Z vidika globalne logične definicije na nivoju sheme, se vrste zapisov razlikujejo po tem, kako se pojavljajo v setih. Pri tem IDA Baza razlikuje štiri vrste zapisov.

  - Samostojni zapisi
    Zapisi, ki ne nastopajo v nobenem setu niti kot nadrejeni niti kot podrejeni zapisi, se imenujejo samostojni zapisi. Do posameznega zapisa se lahko dostopa s pomočjo enoličnega ključa. Zbirke samostojnih zapisov so po svoji uporabnosti podobne indeksnim datotekam.

  - Nadrejeni zapisi
    Zapisi, ki v vseh setih nastopajo kot nadrejeni (owner) zapisi, se imenujejo nadrejeni zapisi. Do posameznega ključa se lahko dostopa s pomočjo enoličnega ključa.

  - Podrejeni zapisi
    Zapisi, ki vedno nastopajo v vseh povezavah kot podrejeni (member) zapisi v vseh setih (in nimajo definiranega enoličnega ključa), se imenujejo podrejeni zapisi. Do zapisov se dostopa na osnovi ključa seta.

- Kombinirani zapisi
  Zapisi, ki lahko nastopajo v enih setih kot nadrejeni, v drugih pa kot podrejeni zapisi, se imenujejo kombinirani zapisi in predstavljajo najbolj splošno obliko zapisa IDA Baze. Do teh zapisov se da dostopati na osnovi enoličnega ključa zapisa in na osnovi ključev setov.

  Poseben primer kombiniranih zapisov so taki zapisi, ki sicer v setih vedno nastopajo kot podrejeni zapisi, vsebujejo pa enolični ključ zapisa in ključe povezav za vsak set. Imajo lastnosti samostojnih in podrejenih zapisov.

* VRSTE PROGRAMSKIH ZAPISOV Z VIDIKA PODSHEME

Programi lahko dostopajo do zapisov IDA Baze na osnovi programskih zapisov, ki so definirani v podshemi. Z vidika vrste dostop do zapisov IDA Baze razlikujemo dve vrsti zapisov:

- Nadrejeni programski zapis
  Dostop do nadrejenih programskih zapisov je možen samo na osnovi enoličnega ključa zapisa. Na tak način je možno dostopati samo do takih zapisov, ki vsebujejo enolični ključ zapisa. To so (z vidika sheme) samostojni, nadrejeni in kombinirani zapisi.

- Podrejeni programski zapisi
  Dostop do podrejenih programskih zapisov je možen na osnovi ključa seta ali na osnovi (internega) DB ključa. Na tak način se lahko dostopa do zapisov, ki vsebujejo ključe povezav. To sta (z vidika sheme) kombinirani in podrejeni zapis.

DEFINIRANJE NADREJENIH IN PODREJENIH PROGRAMSKIH ZAPISOV

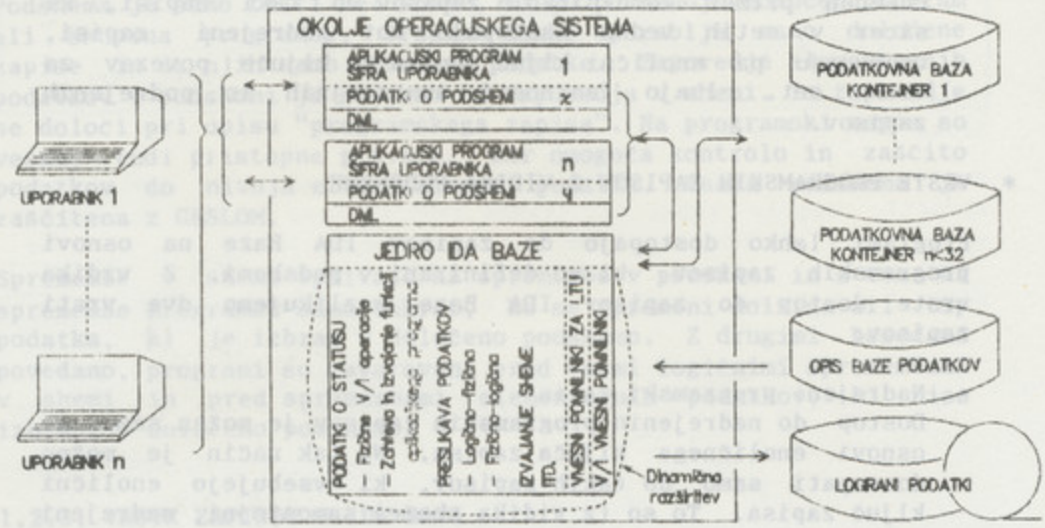
Splošno se lahko opredeli, da je nadrejeni tisti programski zapis, ki nima prirejenega seta (v času kreiranja podsheme). Pri tem velja, da samostojnim in nadrejenim zapisom sheme ni možno prirediti seta. Podrejenim zapisom sheme je obvezno treba prirediti natanko en set. Pri kombiniranih zapisih sheme sta možni dve situaciji. Prva je, da ni bil izbran noben set. Tak zapis ima lastnosti nadrejenega programskega zapisa. Druga pa, da določitev ustreznega seta povzroči, da ima tak zapis lastnosti podrejenega programskega zapisa.

OBRAVNAVANJE ZAPISOV V TEM PRIROČNIKU

Iz navedenih razlogov ta priročnik obravnava v nadaljnjem tekstu samostojne, nadrejene, kombinirane in podrejene zapise z vidika sheme ter nadrejene in podrejene programske zapise z vidika programskega dostopa.

1.3. PODATKOVNI PROSTOR

Shemo in podshemo uporablja sistem za upravljanje baze podatkov, katerega osnovna funkcija je izvajanje operacij nad podatki, ki jih zahtevajo programi. Slika 1-8 kaže, kje se posamezne informacije nahajajo in na kakšen način se uporabljajo.



Slika 1-8. Prikaz okolja podakovne baze

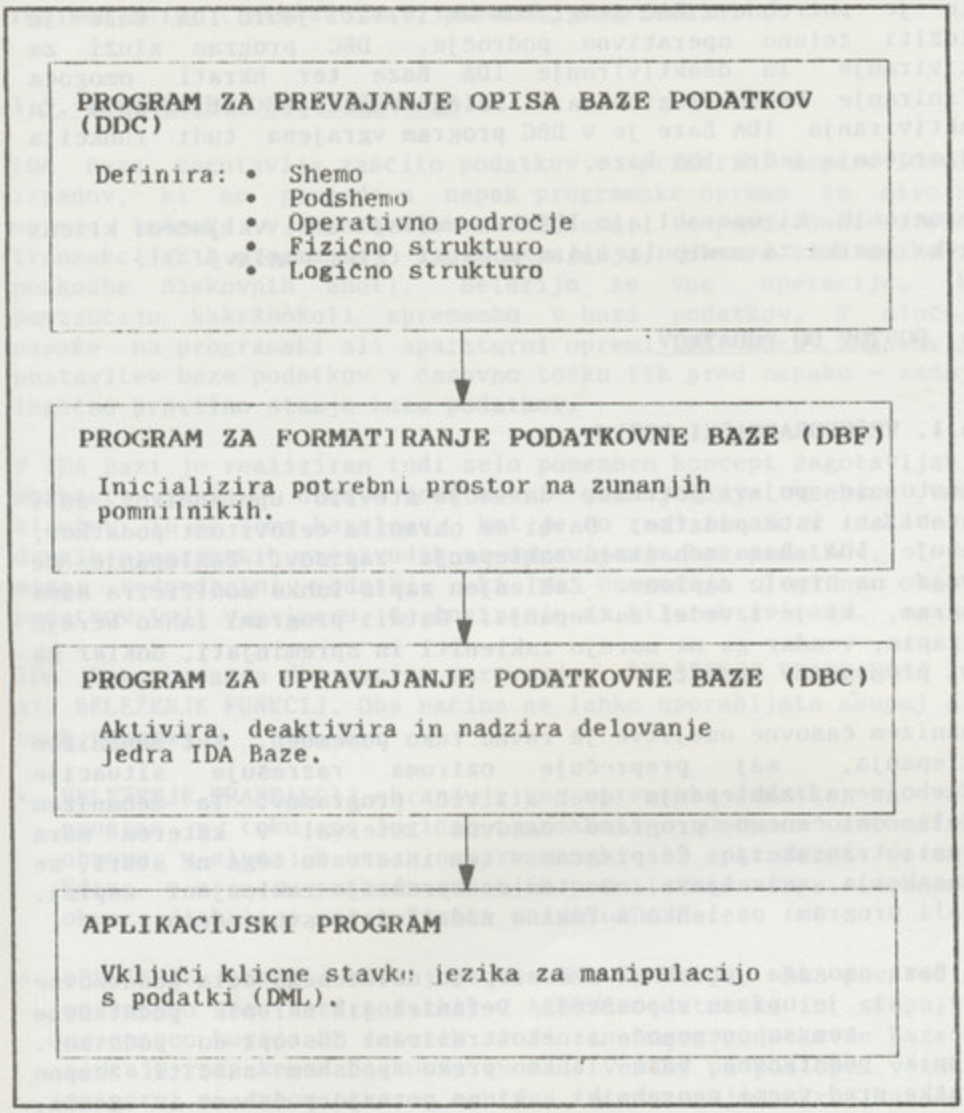
Potek operacij od programa, ki sproži zahtevo za branje določenega zapisa, preko sistema za upravljanje podatkove baze, ki zahtevo izvrši in nazaj do programa:

- Program izvede klic IDA Baze s katerim zahteva branje določenega zapisa in poda ključ.
- IDA Baza v podshemi poišče navedeni programski zapis.
- IDA Baza v shemi poišče pripadajoči tip zapisa.
- IDA Baza na podlagi informacij o fizični strukturi tipa zapisa poišče ustrezni zapis.
- IDA Baza zahteva od operacijskega sistema branje fizičnega zapisa na diskovni enoti.
- Operacijski sistem prenese zahtevane podatke iz zunanjega pomnilnika v V/I področje IDA Baze.
- IDA Baza na podlagi informacij iz podsheme izbere samo zahtevane podatke.
- IDA Baza prenese zahtevane podatke v program.
- IDA Baza zagotovi tudi informacijo o uspešnosti ali neuspešnosti zahtevane operacije.

- Program preverja uspešnost operacije in uporabi dobljene podatke.

Opisan je primer branja, vendar se tudi ostale operacije izvajajo na podoben način.

Slika 1-9 kaže zaporedje operacij od definicije logične strukture do programa.



Slika 1-9. Od definicije do programa

Najprej je potrebno z DDC programom definirati shemo, podsheme, operativno področje, fizično in logično strukturo (DDC program se uporablja tudi kasneje za modifikacije in izboljšave že obstoječih struktur).

Ko so definirane vse potrebne strukture, je potrebno inicializirati potrebni prostor na zunanjih pomnilnikih s programom za formatiranje fizične strukture (DBF). DBF program služi tudi za ponovno formatiranje že obstoječe fizične strukture v primerih reorganizacije podatkov ali za povečanje obstoječe fizične strukture. DBF program poizkuša pri inicializaciji zagotoviti zvezen prostor na zunanjih pomnilnikih zaradi boljših zmogljivosti, vendar IDA Baza pravilno deluje tudi če ta prostor ni zvezen.

Nato je potrebno z DBC programom aktivirati jedro IDA Baze in naložiti želeno operativno področje. DBC program služi za aktiviranje in deaktiviranje IDA Baze ter hkrati omogoča definiranje vrste logiranja. Poleg funkcije aktiviranja in deaktiviranja IDA Baze je v DBC program vgrajena tudi funkcija nadzorovanja jedra IDA Baze.

V programih, ki uporabljajo IDA Bazo morajo biti vključeni klicni stavki jezika za manipulacijo s podatki (glej poglavje 3).

1.4. DOSTOP DO PODATKOV

1.4.1. VEČUPORABNIŠKI REŽIM

Pogosto se pojavi potreba, da večje število uporabnikov želi uporabljati iste podatke. Da bi se ohranila celovitost podatkov, vsebuje IDA Baza mehanizem zaklepanja zapisov. Zaklepanje se izvaja na nivoju zapisov. Zaklenjen zapis lahko modificira samo program, ki je izvedel zaklepanje. Ostali programi lahko berejo ta zapis, vendar ga ne morejo zakleniti in spreminjati, dokler ga prvi program ne sprosti.

Mehanizem časovne omejitve je ravno tako pomemben, kot mehanizem zaklepanja, saj preprečuje oziroma razrešuje situacije medsebojnega zaklepanja dveh ali več programov. Ta mehanizem dodeli določenemu programu časovni interval v katerem mora izvesti transakcijo. Če program v tem intervalu tega ne stori, se transakcija razveljavi. S tem se sprostijo zaklenjeni zapisi, ostali programi pa lahko normalno nadaljujejo svoje delo.

IDA Baza omogoča uporabnikom dostop do določenega dela podatkovne baze, ki je opisan v poshemi. Definira jih skrbnik podatkovne baze. S tem so onemogočeni nekontrolirani dostopi do podatkov. Skrbnik podatkovne baze lahko preko podshem zaščiti zaupne podatke pred vsemi uporabniki, ki ne poznajo podsheme in gesla, ki omogočajo dostop do njih.

1.4.2. OPERATIVNO PODROČJE

Operativno področje se uporablja za izboljšanje učinkovitosti dela pod različnimi pogoji, ne da bi se spremenila logična struktura podatkov ali njihova fizična lokacija.

Možno je zmanjšati čas iskanja podatkov s tem, da se definira več V/I področij, da se izključi del podatkovne baze, ki se trenutno ne uporablja itd. Operativno področje je orodje skrbnika podatkovne baze za optimalno prilagajanje IDA Baze različnim režimom dela (interaktivni, paketni, vzdrževalni).

1.5. BELEŽENJE, OBNOVITEV BAZE

IDA Baza zagotavlja zaščito podatkov v primerih nepredvidenih izpadov, ki so posledica napak programske opreme in strojne opreme (napake v operacijskem sistemu, nepredvideni izpadi transakcijskih programov, izpad napajanja, okvare elekronike in poškodbe diskovnih enot). Beležijo se vse operacije, ki povzročijo kakršnokoli spremembo v bazi podatkov. V slučaju napake na programski ali aparaturni opremi IDA Baza zagotavlja postavitev baze podatkov v časovno točko tik pred napako - zadnje logično pravilno stanje baze podatkov.

V IDA Bazi je realiziran tudi zelo pomemben koncept zagotavljanja možnosti logične obnove podatkov - shranjevanje vsebine vseh ključev in ne samo kazalcev, kot je to realizirano v nakaterih drugih programskih proizvodih za upravljanje baz podatkov. To so sicer redundantni podatki, ki pa omogočajo logično obnovo podatkov tudi v primeru, če logiranje ni bilo aktivirano.

IDA Baza omogoča dva načina varovanja: BELEŽENJE TRANSAKCIJ in/ali BELEŽENJE FUNKCIJ. Oba načina se lahko uporabljata skupaj ali vsak posebej.

- BELEŽENJE TRANSAKCIJ shranjuje vse spremembe podatkov za vsak program v toku ene logične transakcije. Omogoča avtomatsko obnovo v slučajih napak na programski ali aparaturni opremi. Edino v primeru poškodb zunanjih pomnilnikov (npr. mehanska okvara diska) ni možen "vroči start" IDA Baze.

- BELEŽENJE FUNKCIJ shranjuje vse DML funkcije, ki povzročijo spremembo na bazi podatkov v ARHIVSKO datoteko in skupaj z varnostno kopijo DB datotek služi za obnovo podatkovne baze v slučaju napake na diskovni enoti. Vendar pa postopek v takem primeru ni avtomatski in zahteva intervencijo skrbnika baze podatkov. Zaščita podatkov je najzanesljivejša, če se uporabljata oba načina varovanja.

Transakcijski programi, ki delujejo v režimu beleženja logičnih transakcij, morajo v svoji programski logiki upoštevati celovitost logične transakcije. Zato se je treba že pred programsko realizacijo odločiti, ali bo izbrano beleženje transakcij kot varovalni mehanizem. To pa seveda ne velja za programe, ki podatke iz IDA Baze samo berejo. Prav tako to ne velja za servisne programe, ki so vključeni v programski paket IDA. Ti programi delujejo enako, ne glede na režim varovanja podatkov. Zato je skrbniku baze podatkov dana možnost, da se odloča o tipu varovanja podatkov v trenutku zagona baze podatkov. Izbrani režim pa potem velja za vse programe, ki uporabljajo isto bazo podatkov. Režim varovanja podatkov lahko precej vpliva na propustnost. Zaradi tega se je treba odločati med varnostjo podatkov in propustnostjo sistema.

1.6. ZAŠČITA PODATKOV

Podatki, ki so shranjeni v bazi podatkov predstavljajo veliko vrednost, določeni podatki pa so lahko tudi tajni, zato mora biti dostop do njih skrbno kontroliran. IDA Baza zagotavlja dva načina zaščite podatkov. Prvi način je zaščita z geslom vseh opisov baze podatkov (SHEMA, PODSHEMA, PODROČJE in FIZIČNA STRUKTURA), drugi način pa je zaščita samih podatkov z geslom.

1.7. PROGRAMSKI DOSTOP DO BAZE PODATKOV

Jezik za upravljanje s podatki - DML - služi kot vmesnik med aplikacijskim programom in IDA Bazo. DML "CALL" stavki se uporabljajo v programih kadarkoli je potreben dostop do baze podatkov.

DML je zasnovan na štirih osnovnih funkcijah :

Funkcija	Akcija
GET	branje zapisa
REWRITE	modifikacija zapisa
INSERT	dodajanje zapisa
DELETE	brisanje zapisa

Glej poglavje 3 za podrobnejše informacije o jeziku za upravljanje s podatki.

1.8. OZNAČENI ZAPISI

IDA Baza omogoča uporabo označenih zapisov, primarno zaradi konverzije podatkov iz sistemov, ki uporabljajo tak format podatkov. Glej poglavje 2 za dodatne informacije v zvezi z označenimi zapisi.

2. DDC V1.1—PROGRAM ZA OPIS BAZE PODATKOV

Program za opise podatkovnih struktur baze upravlja naslednje sestavne dele baze podatkov:

- SHEME
- LOGIČNE STRUKTURE
- FIZIČNE STRUKTURE
- OPERATIVNA PODROČJA
- PODSHEME
- LOGIČNE STRUKTURE PODSHEM
- SEKVENČNE DATOTEKE

SHEMA

Shema opisuje celotno bazo podatkov in vsebuje opise vseh zapisov in njihovih podatkovnih polj. Shema je identificirana s svojim imenom in zaščitena z geslom. Zapisi so definirani z imeni, prav tako podatkovna polja. Dolžine podatkovnih polj se opisujejo podobno, kot v programskem jeziku COBOL. Shemi je mogoče definirati eno fizično strukturo, eno logično strukturo, več operativnih področij in več podshem s pripadajočimi logičnimi strukturami in opisi sekvenčnih datotek.

LOGIČNA STRUKTURA

Logična struktura vsebuje logičen opis celotne baze podatkov s seti. To omogoča določitev vseh tipov zapisov, ključev in povezav med zapisi. Logična struktura je identificirana s svojim imenom in zaščitena z geslom. Zapisi so definirani v shemi, prav tako ključi za povezave. Shemi je mogoče definirati eno logično strukturo.

FIZIČNA STRUKTURA

Fizična struktura vsebuje opis velikosti zbirk nizov in specifikacije kontejnerskih datotek. Formalno so kontejnerji standardne datoteke na diskih. Možno je specificirati do 32 kontejnerskih datotek. Vsak kontejner lahko vsebuje do 32 različnih zapisov. Vsak zapis je lahko opisan do 16 krat v istem ali različnih kontejnerjih.

Velikost kontejnerja je:

- do 1.073.741.822 blokov po 512 bytov (DELTA/V) ali
- 32.766 blokov po 512 bytov (DELTA/M).

Fizična struktura je od logične strukture popolnoma neodvisna in se lahko neodvisno od logične strukture tudi spreminja. Velikosti zbirk posameznih zapisov se lahko povečujejo ali manjšajo, prav tako faktor blokiranja. Te spremembe nimajo vpliva na programe in podsheme.

OPERATIVNO PODROČJE

Operativno področje zajema del sheme ali celo shemo in predstavlja aktivni (operativni) del sheme. Poleg zapisov v področju definiramo še naslednje operativne parametre:

- število aktivnih programov
- čas dostopa do zaklenjenih zapisov
- število vseh zaklenjenih zapisov
- vhodno / izhodna področja (I/O buffers) za zapise
- število ponovitev vhodno / izhodnih področij

Področje je identificirano z imenom in zaščiteno z geslom.

PODSHEMA

Podshema opisuje zapise, ki so na voljo posameznim programom ali skupinam programov. Opisuje delne zapise za katere je značilno, da vsebujejo definicijo nekaterih (ali vseh) podatkovnih polj, programske pravice dostopa in način pristopa. Ti delni zapisi se imenujejo programski zapisi.

Ker je pogosto potrebno definirati v okviru iste podsheme več kot eno obliko zapisa (zaradi različnih podatkovnih polj, povezav ali programskih pravic), se lahko definira do 16 programskih zapisov, ki se nanašajo na isti zapis iz SHEME.

IDA BAZA predvideva naslednjo zaščito na nivoju programskih zapisov:

- GETP - dovoljeno je branje po fizični sekvenci;
- GET - dovoljeno je branje zapisa;
- INS - dovoljeno je dodajanje zapisa;
- DEL - dovoljeno je brisanje zapisa;
- RWR - dovoljeno je spreminjanje zapisa.

Podshema je identificirana z imenom in zaščitena z geslom.

LOGIČNA STRUKTURA PODSHEME

Logična struktura definira ime SETa po katerem pristopamo k podrejenemu ali kombiniranemu programskemu zapisu. Kadar ne definiramo SETa za pristop do podrejenega ali kombiniranega programskega zapisa, so programske pravice dostopa in način pristopa samo GETP - dovoljeno je branje po fizični sekvenci.

SEKVENČNE DATOTEKE

Podshemi lahko prikljućimo tudi specifikacije sekvenčnih datotek. Te so sestavljene iz specifikacije datoteke (standardna specifikacija), tipa datoteke (vhodna, vhodno/izhodna, izhodna, izhodna z atributom CR/LF) in maksimalne dolžine zapisa.

OPOMBA:

- DELTA/M

Pogoj za uporabo DDC programa je, da je uporabnik priviligiran.

- DELTA/V

Pogoj za uporabo DDC programa je, da so definirani v grupi naslednji simboli - DBV_SHEMA, DBV_PODROCJE, DBV_PODSHEMA in DBV_DINOS.

2.1 AKTIVIRANJE DDC PREVAJALNIKA

2.1.1 AKTIVIRANJE DDC PREVAJALNIKA - VERZIJA DELTA/V

• Format ukaza: DDC [/stikala] ime-datoteke

kjer pomeni:

DDC	Poziv prevajalnika opisa baze podatkov.
/stikalo	Izbor funkcije prevajalnika.
ime-datoteke	Specifikacija datoteke, ki vsebuje opis baze podatkov.

• Stikala:

Stikalo	Nenaglašeno
/[NO]CONVERT	/NOCONVERT
/[NO]DESCRIPTOR	/DESCRIPTOR
/[NO]INIT	/NOINIT
/[NO]LOAD	/NOLOAD

/HELP

/OOl=Ime-prevedene-sheme

/[NO] REPORT /NOREPORT

/MULTI= ime

◦ Stikalo CONVERT

CONVERT prevede DDP datoteke v DDC datoteke. Prevedejo se vse DDP datoteke, ki pripadajo shemi (shema, področja, fizična struktura in podsheme). Zato morajo obstajati ustrezne datoteke z ekstenzijo .DBL, .SAV, .EXT in .LIB datoteke. DDC javlja napake, ki jih ugotavlja v fazi prevajanja v DDC datoteke.

◦ Stikalo DESCRIPTOR

DESCRIPTOR kreira datoteke binarnih opisov (interne tabele, ki jih uporablja jedro baze podatkov) na ustreznih diskovnih seznamih. Nenaglašeno je /DESCRIPTOR.

◦ Stikalo INIT

INIT inicializira ustrezne entitete sistema IDA Leksikon. To stikalo se lahko uporablja samo, če je na sistemu instaliran IDA Leksikon. Nenaglašeno je /NOINIT.

◦ Stikalo LOAD

LOAD polni ustrezne entitete v IDA Leksikonu po prevajanju. Nenaglašeno je /NOLOAD.

◦ Stikalo HELP

Kratka informacija o uporabi programa DDC.

◦ Stikalo /001=Ime-prevedene-sheme

lahko uporabimo za dodajanje in prevajanje novih ali novih verzij že obstoječih SUBSHEM ali RUN-TIME-SHEM v že prevedeno SHEMO.

• Primeri DDC ukazov:

◦ $DDC SHEMAX
(=/NOCONVERT/DESCRIPTOR/NOLOAD/NOINIT)

Kreiranje datoteke DBV-SHEMA:SHEMAX.001, DBV_SHEMA:SHEMAX.REP in DBV_SHEMA:SHEMAX.LIS na osnovi opisa baze DBV_SHEMA:SHEMAX.DDC. Če .DDC vsebuje RUN-TIME-SCHEMA or SUBSCHEMA opis, se kreirata deskriptorja *.EXE na kazalih DBV_PODROCJE oziroma DBV_PODSHEMA.

◦ $DDC /NODESCRIPTOR SHEMAX
(=/NOCONVERT/NODESCRIPTOR/NOLOAD/NOINIT)

Kreiranje datoteke DBV_SHEMA:SHEMAX.001 in poročila DBV_SHEMA: SHEMAX.LIS na osnovi opisa DBV_SHEMA:SHEMAX.DDC.

◦ $DDC /CONVERT /NODESCRIPTOR SCHEMA000
(=/CONVERT/NODESCRIPTOR/NOLOAD/NOINIT)

Prevod DBV_SHEMA:SCHEMA000.SAV (DDP datoteke) in vse pripadajoče podrejene strukture (RT SHEME, PODSHEME) v datoteko opisa DBV_SHEMA: SCHEMA000.DDC. Kreira delovno datoteko DBV_SHEMA: SCHEMA000.001 in poročilo DBV_SHEMA:SCHEMA000.LIS.

◦ $DDC /LOAD /NODESCRIPTOR SCHEMA
(=/NOCONVERT/NODESCRIPTOR/LOAD/NOINIT)

Kreiranje delovne datoteke DBV_SHEMA:SCHEMA.001 in poročilo DBV_SHEMA:SCHEMA.LIS na osnovi definicije DBV_SHEMA:SCHEMA.DDC. Kreira datoteko DBV_SHEMA:SCHEMA.002 s podatki za ustezne entitete IDA Leksikona.

- $DDC /CONVERT /DESCRIPTOR /LOAD SCHEMA000
  (=/CONVERT/DESCRIPTOR/LOAD/NOINIT)

Prevod DBV_SHEMA:SCHEMA000.SAV (DDP datoteke) in pripadajoče podrejene datoteke (RT SHEME in PODSHEME) v datoteko DBV_SHEMA: SCHEMA000.DDC. Kreira datoteko DBV_SHEMA:SCHEMA000.001 in poročilo DBV_SHEMA:SCHEMA000.LIS. Kreira datoteko DBV_SHEMA: SCHEMA000.002 s podatki za ustrezne entitete IDA Leksikona.
Če .DDC vsebuje RUN-TIME-SCHEMA or SUBSCHEMA opis, se kreirata deskriptorja *.EXE na kazalih DBV_PODROCJE oziroma DBV_PODSHEMA.

- $DDC /001=SHEMAX SUBSCHEMA
  (=/NOCONVERT/DESCRIPTOR/NOLOAD/NOINIT)

SHEMAX je datoteka prevedene sheme DBV_SHEMA:SHEMAX.001. SUBSCHEMA je opis nove podsheme (ali že obstoječe - nova verzija) za shemo SHEMAX v datoteki DBV_SHEMA:SUBSCHEMA.DDC. Stikalo lahko uporabimo za prevajanje posameznih PODSCHEM. Deskriptorji SHEME, LOGIČNE STRUKTURE in FIZIČNE STRUKTURE pa morajo biti že prevedeni v datoteki DBV_SHEMA:SHEMAX.001. DDC kreira datoteko DBV_SHEMA:SUBSCHEMA.REP (poročilo), DBV_SHEMA:SUBSCHEMA.LIS in novi deskriptor SUBSCHEMA.SUB na kazalu DBV_PODSHEMA.

- $DDC /001=SHEMAX R-T-SCHEMA
  (=/NOCONVERT/DESCRIPTOR/NOLOAD/NOINIT)

SHEMAX je datoteka prevedene sheme DBV_SHEMA:SHEMAX.001. R-T-SCHEMA je opis nove run-time sheme (ali že obstoječe-nova verzija) za shemo SHEMAX v datoteki DBV_SHEMA:R-T-SCHEMA.DDC. Stikalo lahko uporabimo za prevajanje posameznih RUN-TIME-SCHEM. Deskriptorji SHEME, LOGIČNE STRUKTURE in FIZIČNE STRUKTURE pa morajo biti že prevedeni v datoteki DBV_SHEMA:SHEMAX.001. DDC kreira datoteko DBV_SHEMA:R-T-SCHEMA.REP (poročilo), DBV_SHEMA: R-T-SCHEMA.LIS in novi deskriptor R-T-SCHEMA.EXE na kazalu DBV_PODROCJE.

- $DDC /001=SHEMAX /NODESCRIPTOR R-T-SCHEMA
  (=/NOCONVERT/NODESCRIPTOR/NOLOAD/NOINIT)

SHEMAX je datoteka prevedene sheme DBV_SHEMA:SHEMAX.001. R-T-SCHEMA je opis nove run-time sheme (ali že obstoječe-nova verzija) za shemo SHEMAX v datoteki DBV_SHEMA: R-T-SCHEMA.DDC. Stikalo lahko uporabimo za prevajanje posameznih RUN-TIME-SCHEM. Deskriptorji SHEME, LOGIČNE STRUKTURE in FIZIČNE STRUKTURE pa morajo biti že prevedeni v datoteki DBV_SHEMA: SHEMAX.001. DDC kreira datoteko DBV_SHEMA:R-T-SCHEMA.LIS.

2.1.2 AKTIVIRANJE DDC PREVAJALNIKA - VERZIJA DELTA/M

• Format ukaza: DDC [/stikala] ime-datoteke

kjer pomeni:

DDC	Poziv prevajalnika opisa baze podatkov.
/stikalo	Izbor funkcije prevajalnika.
ime datoteke	Specifikacija datoteke, ki vsebuje opis baze podatkov.

• Stikala:

Stikalo	Nenaglašeno
/[NO]CONVERT	/NOCONVERT
/[NO]DESCRIPTOR	/DESCRIPTOR
/HELP	
/001=Ime-prevedene-sheme	

◦ Stikalo CONVERT

CONVERT prevede DDP datoteke v DDC datoteke. Prevedejo se vse DDP datoteke, ki pripadajo shemi (shema, področja, fizična struktura in podsheme). Zato morajo obstajati ustrezne datoteke z ekstenzijo .DBL, .SAV, .EXT in .LIB datoteke. DDC javlja napake, ki jih ugotavlja v fazi prevajanja v DDC datoteke.

◦ Stikalo DESCRIPTOR

DESCRIPTOR kreira datoteke binarnih opisov (interne tabele, ki jih uporablja jedro baze podatkov) na ustreznih diskovnih seznamih. Nenaglašeno je /DESCRIPTOR.

◦ Stikalo HELP

Kratka informacija o uporabi programa DDC.

◦ Stikalo /001=Ime-prevedene-sheme

lahko uporabimo za dodajanje in prevajanje novih ali novih verzij že obstoječih SUBSHEM ali RUN-TIME-SHEM v že prevedeno SHEMO.

• Primeri DDC ukazov:

◦ >DDC SHEMAX
(=/NOCONVERT/DESCRIPTOR)

Kreiranje datoteke [1,63]:SHEMAX.001 prevod, [1,62]:SHEMAX.REP poročilo in [1,62]:SHEMAX.LST na osnovi opisa baze [1,62]: SHEMAX.DDC. Če SHEMAX.DDC vsebuje RUN-TIME-SCHEMA ali SUBSCHEMA opis, se kreirata deskriptorja *.EXE na kazalih [1,61] oziroma *.SUB na [1,60].

◦ >DDC /NODESCRIPTOR SHEMAX
(=/NOCONVERT/NODESCRIPTOR)

Kreiranje datoteke [1,63]:SHEMAX.001 prevod in poročila [1,62]:SHEMAX.LST na osnovi opisa [1,62]:SHEMAX.DDC.

◦ >DDC /CONVERT /NODESCRIPTOR SCHEMA000
(=/CONVERT/NODESCRIPTOR)

Prevod [1,62]:SCHEMA000.SAV (DDP datoteke) in vse pripadajoče podrejene strukture (RT sheme, podsheme) v datoteko opisa [1,62]: SCHEMA000.DDC. Kreira delovno datoteko [1,63]:SCHEMA000.001 ali poročilo [1,62]:SCHEMA000.LST.

◦ >DDC /001=SHEMAX SUBSCHEMA
(=/NOCONVERT/DESCRIPTOR)

SHEMAX je datoteka prevedene sheme [1,63]:SHEMAX.001. SUBSCHEMA je opis nove subsheme (ali že obstoječe nova verzija) za shemo SHEMAX v datoteki [1,62]: SUBSCHEMA.DDC. Stikalo lahko uporabimo za prevajanje posameznih SUBSCHEM. Deskriptorji SHEME, LOGIČNE STRUKTURE in FIZIČNE STRUKTURE pa morajo biti že prevedeni v datoteki [1,63]:SHEMAX.001. DDC kreira datoteko [1,62]: SUBSCHEMA.REP (poročilo), [1,62]:SUBSCHEMA.LST in novi deskriptor SUBSCHEMA.SUB na kazalu [1,60.

◦ >DDC /001=SHEMAX R-T-SCHEMA
(=/NOCONVERT/DESCRIPTOR)

SHEMAX je datoteka prevedene sheme [1,62]:SHEMAX.001. R-T-SCHEMA je opis nove run-time sheme (ali že obstoječe nova verzija) za shemo SHEMAX v datoteki [1,62]: R-T-SCHEMA.DDC. Stikalo lahko uporabimo za prevajanje posameznih RUN-TIME-SCHEM. Deskriptorji SHEME, LOGIČNE STRUKTURE in FIZIČNE STRUKTURE pa morajo biti že prevedeni v datoteki [1,63]:SHEMAX.001. DDC kreira datoteko [1,62]:R-T-SCHEMA.REP (poročilo), [1,62]:R-T-SCHEMA.LST in novi deskriptor R-T-SCHEMA.EXE na kazalu [1,61].

```
>DDC /001 SHEMAX /NODESCRIPTOR R-T-SCHEMA
(=/NOCONVERT/NODESCRIPTOR)
```

SHEMAX je datoteka prevedene sheme [1,63]:SHEMAX.001. R-T-SCHEMA je opis nove run-time sheme (ali že obstoječe - nova verzija) za shemo SHEMAX v datoteki [1,62]: R-T-SCHEMA.DDC. Stikalo lahko uporabimo za prevajanje posameznih RUN-TIME-SCHEM. Deskriptorji SHEME, LOGIČNE STRUKTURE in FIZIČNE STRUKTURE pa morajo biti že prevedeni v datoteki [1,63]:SHEMAX.001. DDC kreira datoteko [1,62]:R-T-SCHEMA.LST.

2.1.3 POROČILA O NAPAKAH PRI PREVAJANJU

Oblika poročil:

```
*DDC--)F-Usodna (fatalna) napaka
*DDC--)I-Informacija o odkriti napaki
*DDC--)W-Opozorilo (warning)

DDC -- FATALS nn, INFORMATIONALS nn, WARNINGS nn
```

F-Severe error:
Prevajalnik ni poskusal odpraviti napake. Zato morate popraviti napako in ponovno prevesti opis baze podatkov.

I-Informational error:
Ta napaka pogosto pomeni nestandardno stavčno strukturo.

W-Warning error:
Prevajalnik je sam odpravil napako. Preglejte ta popravek in se prepričajte ali je tisto, kar ste želeli, sicer so možne nepredvidljive posledice.

2.2 ELEMENTI DDC JEZIKA

DDC opis je razdeljen v 7 delov:

- Schema-description - Shema
- Logical-structure-description - Logična struktura
- Physical-structure-description- Fizična struktura
- Run-time-schema-description - Operativno področje
- Subschema-description - Podshema
- Subschema-logical-description - Logična struktura podsheme
- Sequential-record-description - Sekvenčne datoteke

Oblika stavka:

Vsak stavek je sestavljen iz ključnih (obveznih) besed, dopolnilnih (neoveznih) besed, spremenljivk (uporabnikovih imen podatkov) in konstant. Vsak stavek mora biti napisan v svoji in v eni sami vrstici.

V primeru so ključne besede, spremenljivke in konstante napisane z velikimi črkami, dopolnilne besede pa z malimi črkami. Tekst, ki sledi * (zvezdici), je komentar.

Primeri DDC stavkov:

```
RECORD name is CSTOMR     * Za zvezdico je vedno komentar

RECORD                    = ključna beseda
name                      = dopolnilna beseda
is                        = dopolnilna beseda
CSTOMR                    = spremenljivka

ITEM description is 05 OWNKEY PIC X(6)

ITEM                      = ključna beseda
description               = dopolnilna beseda
is                        = dopolnilna beseda
05 OWNKEY PIC X(6)        = spremenljivka
COBOLski stavek picture

BLOCK contains 8 SECTORS

BLOCK                     = ključna beseda
contains                  = dopolnilna beseda
8                         = numerična konstanta
SECTORS                   = selektor
```

2.2.1 PRIMER OPISA SHEME

Vsak opis sheme se začne s stavkom:

```
SCHEMA-DESCRIPTION
```

- Ime in geslo sheme

```
SCHEMA  name is SALESA
PASSWORD    is SALESA
```

Shema opisuje obliko vseh zapisov baze podatkov. Vsebuje ime, geslo in imena vseh zapisov s priadajočim opisom podatkovnih polj. Ime sheme sestavlja 6 alfanumeričnih znakov. Znak nič (0) ne sme biti prvi znak imena. Prirejeno ime sheme je "NONAME". Geslo sestavlja 6 alfanumeričnih znakov. Prirejeno geslo je "NOAME".

• Ime zapisa

```
RECORD name is CSTOMR
```

Ime zapisa v shemi sestavlja 6 alfanumeričnih znakov. Nič (0) ne sme biti prvi znak imena. Privzeta vrednost za ime zapisa je "NONAME". Zapis definira vsa pripadajoča podatkovna polja. Zapis kupca lahko vsebuje npr. podatkovne polja ime kupca, naslov, telefonska stevilka itd.

• Ime podatkovnega polja

```
ITEM description is 05 OWNKEY PIC X(6)
```

Ime podatkovnega polja sestavlja 6 alfanumeričnih znakov; prvi znak ne sme biti nič (0). Največje število podatkovnih polj za zapis je 256. Vsak zapis vsebuje najmanj eno definicijo podatkovnega polja, to je ključa. Teh sta dve vrsti (ključ zapisa, ključ seta). Rezervirana imena so REQUAL, [redacted], SYNKEY, FILLER. Imena podatkovnih polj za posamezni zapis morajo biti enoznačna.

OWNKEY

V zgornjem primeru je oblika podatkovnega polja OWNKEY je 6 A/N znakov (X(6)). "05" je nivojsko število in predstavlja mesto podatka v strukturi podatkov v zapisu. Uporabljajo se lahko nivojska števila od 05 do 45. Na voljo so naslednji elementi COBOLske sintakse:

Picture (PIC). Pomen:

A	Alfa podatek
X	Alfanumerični podatki
9	Numerični podatki
S9	Numerični podatki s predznakom (Obvezno za COMP-3 podatkovna polja)
9V9	Decimalna vejica (pika) v numeričnih podatkih
9 COMP	Numerični podatki (interna oblika je integer)
S9 COMP-3	Numerični podatki (pakirana decimalna oblika)

S PRAVILNO RAZVRSTITVIJO PODATKOVNIH POLJ SE DA IZOGNITI VRIVANJU PRAZNEGA MESTA ("FILLER" VRIVA DDC) ZARADI PORAVNAVE POMNILNIŠKIH REZERVACIJ.

Tabela rezervacij pomnilniških lokacij za COMP polja:

S9	do	S9(4)	1 beseda (2 byta)
S9(5)	do	S9(9)	2 besedi (4 byti)
S9(10)	do	S9(18)	4 besede (8 bytov)

"COMP-3" je pakirani decimalni format. Dve stevilki sta shranjeni v enem bytu, predznak pa v skrajnem desnem bytu pakiranega polja. (Programsko je poravnan na mejo besede v pomnilniku).

```
ITEM description is 05 CSTNAM PIC X(50)
ITEM description is 05 ADRESS PIC X(60)
ITEM description is 05 TELEPH PIC 9(9)
ITEM description is 05 REMARK PIC X(6)

RECORD name is PRDUCT

ITEM description is 05     OWNKEY PIC X(12)
```

Podatkovno polje PRDUCT predstavlja grupni podatek.

```
ITEM description is 05     PRDUCT
```

Navodilo REDEFINES omogoča različne opise istega dela pomnilnika. Podatkovni element PARTAL opisuje isti pomnilnik kot PARTNO.

```
ITEM description is   11   PARTNO PIC X(60)
ITEM description is   11   PARTAL REDEFINES PARTNO
```

Grupni podatek PARTAL je opisan s podatkovnimi elementi PART01 in PART03.

```
ITEM description is     22 PART01 PIC A(10)
ITEM description is     22 PART02 PIC X(20)
ITEM description is     22 PART03 PIC X(30)
ITEM description is   11   PRICE1 PIC 9(7)V999
ITEM description is   11   QUANT1 PIC 9(7)V999

RECORD name is ORDERS
```

Zapis, ki ima INDEX name se tretira kot kombinirani tip zapisa (ki je dostopen po ključu zapisa in po ključu povezave). To se pove v logični strukturi.

```
INDEX  name is $A0000

ITEM description is 05 OWNKEY PIC X(5)
ITEM description is 05 CUSTNO PIC X(6)
ITEM description is 05 ALDATE OCCURS 2 times

ITEM description is   10 DATEAL PIC 9(6)

ITEM description is 05 DATEOT REDEFINES ALDATE
ITEM description is   10 ORDDAT PIC 9(6)
ITEM description is   10 TRNDAT PIC 9(6)
```

Definicija zapisa brez neobveznih besed:

```
RECORD ORDITM
ITEM 05 ORDRNO PIC X(5)
ITEM 05 DATAAL PIC X(28)
ITEM 05 DATAAL-05 REDEFINES DATAAL
ITEM 10 PARTNO PIC X(12)
ITEM 10 QUORDR PIC 9(5)V999
ITEM 10 QUDELI PIC 9(5)V999
```

```
RECORD name is SEQREC

ITEM description is 05     SEQDAT PIC X(120)

RECORD name is SYSREC

ITEM description is 05     OWNKEY PIC X(12)
ITEM description is 05     SYSDAT PIC X(108)
```

Uporaba navodila "REQUAL=??" za kodirani zapis.

```
RECORD name is REQREC
```

"REQUAL=??" je navodilo samo za podrejene ali kombinirane zapise, kar omogoča večkratno strukturiranje (redefiniranje) enega podatkovnega polja. Oznako sestavljata dva alfanumerična znaka. Kombinacije oznak morajo biti enoznačne. Za tem navodilom lahko vnašamo podatkovna polja samo na nivoju večjem od navolila "REQUAL=??". Navodilo "REQUAL=??" se lahko uporabi večkrat.

```
ITEM description is 05 REQUAL PIC XX    * Vedno prvo polje v kodiranem zapisu
ITEM description is 05 OWNKEY PIC X(12)
ITEM description is 05 VSEBIV PIC X(80)
```

Polje "VSEBIV" bo redefinirano dvakrat. Prvič uporabimo navodilo "REQUAL=SB" - za njim moramo pisati elemente redefinicije z nivojskimi številkami.

```
* REQUAL-SB Stalno bivališče

ITEM description is 05 REQUAL-SB REDEFINES VSEBIV
ITEM description is 10 PSTESB PIC 9(5)  * Poštna štev.
ITEM description is 10 KRAJSB PIC X(30) * Bivališče
ITEM description is 10 ULICSB PIC X(41) * Ulica
ITEM description is 10 STEVSB PIC 9(4)  * Številka
```

Drugo redefinicijo začnemo z navodilom "REQUAL=ZB" in za njim z novo strukturo elementov z nivojskimi številkami.

```
* REQUAL-ZB Začasno bivališče

ITEM description is 05 REQUAL-ZB REDEFINES VSEBIV
ITEM description is 10 PSTEZB PIC 9(5)  * Poštna štev.
ITEM description is 10 KRAJZB PIC X(30) * Bivališče
ITEM description is 10 ULICZB PIC X(41) * Ulica
ITEM description is 10 STEVZB PIC 9(4)  * Številka

END-OF-DESCRIPTION * To je zadnji stavek vsakega opisa
```

2.2.2 PRIMER OPISA LOGIČNE STRUKTURE

Opis logične strukture sheme je opis vseh povezav med zapisi (odnos nadrejeni-podrejeni zapis). Za vsako povezavo (set) morata biti definirana nadrejeni in podrejeni zapis ter pripadajoči ključ (nadrejenega) zapisa in ključ povezave (podrejenega) zapisa. Dva zapisa povezuje lahko več setov.

- Nadrejeni zapis se lahko naslovi na osnovi ključa zapisa. Tak zapis je lahko nadrejen povezavi (setu) podrejenih zapisov. Zapis vsebuje direktni ključ (OWNKEY). Direktni ključ je argument za direktni dostop do enega samega zapisa v zbirki.

- Podrejeni zapis se lahko naslovi s ključem povezave. Ti zapisi so podrejeni v vseh setih. Ključ povezave se uporablja za vzpostavljanje povezave med nadrejenimi in podrejenimi zapisi in za oblikovanje verig podrejenih zapisov. Služi tudi kot argument za dostop na začetek ali konec verige podrejenih zapisov. Dolžina vnešenega imena podatkovnega polja za ključ povezave iz podrejenega zapisa mora biti enaka dolžini direktnega ključa iz nadrejenega zapisa. Vsak podatek je lahko SAMO ENKRAT izbran za ključ povezave. Potrebna je redefinicija tega podatka, da lahko nastopa kot ključ povezave v več setih. Če v podrejenem zapisu ni podatkovnega polja z dolžino enako dolžini ključa nadrejenega zapisa, ni mogoče kreirati povezave med tema dvema zapisoma.

- Kombinirani zapis se lahko dostopa s ključem zapisa ali s ključem povezave. V nekaterih setih nastopa kot podrejeni zapis, v nekaterih pa lahko kot nadrejeni zapis. Zapis vsebuje direktni ključ in ključe povezave. Kombinirani zapisi so v nekaterih povezavah nadrejeni, v nekaterih povezavah pa so lahko podrejeni.

- Nekoliko posebna oblika nadrejenega zapisa je sistemski zapis. Dostopa se s ključem zapisa. Zanj je značilno da se ne pojavlja v nobeni povezavi z drugimi zapisi. V bistvu predstavlja pravo indeksno (ne ideksnosekvenčno) datoteko.

Shemi pripada en sam opis logične strukture.

```
LOGICAL-STRUCTURE-DESCRIPTION
LOGICAL-STRUCTURE name is SALESA

SET          name is CSTORD * Ime seta
OWNER  record  name is CSTOWR * Nadrejeni zapis
KEY    item    name is OWNKEY * Ključ nadrejenega = Direktni ključ
MEMBER  record name is ORDERS * Podrejeni zapis
KEY     item   name is CUSTNO * Ključ podrejenega = Ključ povezave

SET          name is PRDORD
OWNER  record name is PRDUCT
KEY    item   name is OWNKEY
MEMBER record name is ORDITM
KEY    item   name is PARTNO
```

Za kombinirane zapise uporabi INDEX name za SET name. MEMBER record name in KEY item name morata biti opisana kot NULL.

```
SET name is $A0000
OWNER  record name is ORDERS
KEY    item   name is OWNKEY
MEMBER record name is NULL
KEY    item   name is NULL
```

Opis seta brez dopolnilnih besed:

```
SET    ORDORD
OWNER  ORDERS
KEY    OWNKEY
MEMBER ORDITM
KEY    ORDRNO
```

Pri opisu sistemskega zapisa morata biti MEMBER record name in KEY item name opisana kot NONE.

```
SET name is SYSSET

OWNER  record name is SYSREC
KEY    item   name is OWNKEY
MEMBER record name is NONE
KEY    item   name is NONE

END-OF-DESCRIPTION
```

2.2.3 PRIMER OPISA FIZIČNE STRUKTURE

Opis fizične strukture je opis kontejnerskih datotek, kjer so shranjeni tipi zapisov. Specifikacija kontejnerja je standardna DELTA/M ali DELTA/V specifikacija. Shemi pripada ena sama fizična struktura.

```
PHYSICAL-STRUCTURE-DESCRIPTION
PHYSICAL-STRUCTURE name is SALESA
LOGICAL CONTAINER name is SALES
CONTAINER file name is dev:[directory]SALES.CON
```

Kontejner SALES vsebuje štiri različne tipe zapisov.

```
CONNECT record      CSTOMR
```

Zapise v kontejner izbiramo z njihovimi imeni. Več kot 32 različnih tipov zapisov ni mogoče izbrati v kontejner. Kombinirani zapisi zavzamejo dve prosti mesti v kontejnerju, eden je za sam zapis, drugi pa za njegovo indeksno področje. EN ZAPIS JE LAHKO IZBRAN V RAZLIČNIH KONTEJNERJIH NAJVEČ 16-KRAT.

Zbirka zapisov CSTOMR vsebuje 150 zapisov.

```
OCCURENCY number    is 150
```

Načini opisa dolžine bloka:

```
BLOCK contains      7   RECORDS
BLOCK contains      2   SECTORS
BLOCK contains      2   * (Nenaglašeno je RECORDS)
```

Blok faktor pove koliko zapisov bo v bloku. Dolžina bloka je mnogokratnik 512 (512, 1024...do 8192 znakov). Število zapisov v bloku definiramo samo pri prvem vpisu zapisa v kontejner.

```
CONNECT record      PRDUCT
OCCURENCY number    is 200
```

V tem primeru je dolžina bloka dva sektorja; ker je dolžina sektorja 512, je blok dolg 1024 znakov. DDC izračuna, koliko zapisov gre v blok.

```
BLOCK contains      2   SECTORS
```

```
CONNECT record          ORDERS
OCCURENCY number    is 1000
BLOCK contains          3
CONNECT record          ORDITM
OCCURENCY number    is 2000
BLOCK contains          12   records
CONTAINER name is SYSREC
CONTAINER file name is DBV$HEMA:SYSREC.CON
CONNECT record          SYSREC
OCCURENCY number    is 100
BLOCK contains          11   records
END-OF-DESCRIPTION
```

2.2.4 PRIMER OPISA OPERATIVNEGA PODROČJA

Kreiranje OPERATIVNEGA PODROČJA je sestavljeno iz:

- Imena in gesla operativnega področja
- Izbora tipa zapisov za operativno področje
- Operativnih opcij
- Vhodno/izhodnih področij

Ime operativnega področja sestavlja sedem alfanumeričnih znakov. Prvih šest znakov je ime sheme, sedmi znak pa označuje področje v shemi.

```
RUN-TIME-SCHEMA-DESCRIPTION
RUN-TIME-SCHEMA name is SALESA1OD
```

Geslo sestavlja 6 alfanumeričnih znakov.

```
PASSWORD            is SALESA:
```

Stevilo programov, ki so istočasno aktivni v področju je med "2" in "52(DELTA/M), 99(DELTA/V)".

```
ACTIVE      programs    is  10
```

Največje število "sočasno" zaklenjenih zapisov v aktivnem operativnem področju je lahko med 0 in 999.

```
LOCKED      records     is  500
```

Čas dostopa do zaklenjenih zapisov je v sekundah izražena velikost časovnega intervala, v katerem si lahko posamezen program zagotovi ekskluzivno pravico do uporabe teh zapisov. V takem primeru drugi programi ne morejo zaklepati ali spreminjati teh zapisov. Po preteku časov intervala lahko baza podatkov programu ta zapis odvzame in razveljavi logično transakcijo. Velikost časovnega intervala je lahko od 0 do 999 sekund.

```
ACCESS      time        is  60
```

Vsakemu zapisu izbranemu v področje je potrebno definirati vhodno/izhodno področje (I/O-AREA). Ime V/I področja sestavlja šest alfanumeričnih znakov. Imena V/I področij morajo biti različna za različne tipe zapisov.

```
I/O-AREA      name       is  IOCSTO
COPY          number     is  1
CONNECT       record         CSTOMR
```

- Podrejeni zapisi imajo lahko skupna V/I področja.

- Nadrejeni zapisi imajo lahko skupna V/I področja.

- Kombinirani zapisi in podrejeni zapisi imajo lahko skupna V/I področja.

Za kombinirani zapis DDC doda še posebno V/I področje za indeksni del tega zapisa.

Stevilo ponovitev posameznega V/I področja je med "1" in "32". V/I področja so vmesni pomnilniki, ki se uporabljajo za izvajanje vhodno/izhodnih operacij na diskovnih enotah. Posamezen zapis ima lahko en vmesni pomnilnik, ki pa je lahko ponovljen od 1 do 32 krat. Vseh kopij vseh vmesnih pomnilnikov je lahko največ 512.

```
I/O-AREA      name       is  IOPRDU
COPY          number     is  12
CONNECT       record         PRDUCT
I/O-AREA      name       is  IOORDE
COPY          number     is  1
CONNECT       record         ORDERS
I/O-AREA      name       is  IOORDI
COPY          number     is  8
CONNECT       record         ORDITM
END-OF-DESCRIPTION
```

2.2.5 KREIRANJE PODSHEME JE SESTAVLJENO IZ:

- Imena in gesla podsheme
- Imena projekta in načina pristopa v podshemo
- Izbora programskih zapisov za podshemo
- Izbora pristopa v programske zapise
- Izbora podatkovnih polj

Primer opisa podsheme:

```
SUBSCHEMA-DESCRIPTION
```

Ime podsheme sestavlja devet (9) alfanumeričnih znakov. Prvih šest znakov je ime sheme. Sedmi znak enolično označuje področje. Osmi in deveti znak enolično označujeta podshemo.

```
SUBSCHEMA name is SALESA101
```

Geslo sestavlja 6 alfanumeričnih znakov:

```
PASSWORD     is SALESA
```

Ime projekta sestavlja največ osem (8) alfanumeričnih znakov.

```
PROCESS name is DEMODB
```

Pristopni pravici za dostop v bazo podatkov sta UPDATE ali READONLY. Nenaglašena pravica je READONLY.

```
ACCESS-RIGHTS  is UPDATE
```

UPDATE pomeni, da se podatke lahko dodaja, ažurira, briše in bere.

READONLY pomeni, da se podatke lahko samo bere.

Ime programskega zapisa je sestavljeno iz imena zapisa in še treh dodatnih znakov, na primer 001 ali ADL. V eni podshemi lahko izbranemu zapisu priredimo do 16 programskih zapisov. Programski zapis je tipično samo en del zapisa sheme. Vsebuje samo tista podatkovna polja, ki jih program potrebuje. V okviru iste podsheme je lahko zapis definiran v obliki več programskih zapisov. Ti se med seboj lahko razlikujejo glede na podatkovna polja, pristopne pravice itd.

```
CONNECT subschema record CSTOMR001 from record CSTOMR
```

Zaščita zapisa (proti ostalim uporabnikom) je SHARED ali PRIVILEGED. SHARED dovoljuje vsem uporabikom delitev zapisov določene zbirke, PRIVILEGED pa zagotavlja ekskluzivno pravico dostopanja. Nenaglešena zaščita je SHARED.

```
RECORD-PROTECTION  is SHARED
```

Pristopne pravice do programskega zapisa:

```
RECORD-ACCESS  is GETP GET
```

- GETP - Branje po fizičnem vrstnem redu
- GET - Branje po ključu - dovoljeno je branje zapisov s katerokoli funkcijo razen GETP
- ~~ADD~~ - Dodajanje zapisa
- DEL - Brisanje zapisa
- RWR - Spreminjanje zapisa

Nenaglašena pristopna pravica je GETP. Izbran način dela s programskim zapisom je lahko poljuben. Na primer: GETP ali GET ali GETP, GET ali GET RWR ...

```
SELECT item      OWNKEY   * PIC X(6)
SELECT item      CSTNAM   * PIC X(50)
SELECT item      ADRESS   * PIC X(60)
SELECT item      TELEPH   * PIC 9(9)
SELECT item      REMARK   * PIC X(6)

CONNECT subschema record CSTOMR002 from record CSTOMR
RECORD-PROTECTION is SHARED/RECORD-ACCESS is GET INS RWR

SELECT item      OWNKEY   * PIC X(6)
SELECT item      CSTNAM   * PIC X(50)
SELECT item      ADRESS   * PIC X(60)
SELECT item      TELEPH   * PIC 9(9)
SELECT item      REMARK   * PIC X(6)

CONNECT subschema record ORDITM001 from record ORDITM
RECORD-PROTECTION is SHARED/RECORD-ACCESS is GETP GET

SELECT item      ORDRNO   * PIC X(5)
SELECT item      DATAAL   * PIC X(28)

CONNECT subschema record ORDITM002 from record ORDITM
RECORD-PROTECTION is SHARED/RECORD-ACCESS is GET INS RWR

SELECT item      ORDRNO   * PIC X(5)
SELECT item      DATAAL   * PIC X(28)

CONNECT subschema record ORDITM003 from record ORDITM
RECORD-PROTECTION is SHARED/RECORD-ACCESS is GET RWR

SELECT item      ORDRNO   * PIC X(5)
SELECT item      PARTNO   * PIC X(12)
SELECT item      QUDELI   * PIC 9(5)V9(3)

END-OF-DESCRIPTION
```

2.2.6 PRIMER OPISA LOGIČNE STRUKTURE PODSHEME

```
SUBSCHEMA-LOGICAL-DESCRIPTION
SUBSCHEMA name is SALESA101

ACCESS subschema record ORDITM001 with set PRDORD
ali
ACCESS ORDITM001 PRDORD
```

Da se lahko dostopi do podrejenega zapisa, je treba navesti povezavo - set. Za vsak programski zapis se lahko navede en sam set. Kadar je potreben dostop do zapisov iste zbirke po različnih setih, je potrebno definirati več programskih zapisov. Do zapisov kombinirane zbirke se lahko dostopa tudi direktno po ključu zapisa. V tem primeru za ta programski zapis set ne sme biti naveden.

Kadar izberemo za "member" zapis set, ki ima ključ v kodiranem delu zapisa, lahko v izboru (SELECT ITEM) navedemo vsa nekodirana podatkovna polja in podatkovna polja z kodo enako kodi ključa seta.

```
ACCESS subschema record ORDITM002 with set ORDORD
ACCESS subschema record ORDITM003 with set ORDORD

END-OF-DESCRIPTION
```

2.2.7 PRIMER OPISA SEKVENČNE DATOTEKE

```
SEQUENTIAL-RECORD-DESCRIPTION
```

Ime subsheme kateri dodamo eno ali več sekvenčnih datotek.

```
SUBSCHEMA name is SALESA101
```

Ime sekvenčne datoteke je dolgo šest (6) alfanumeričnih znakov. SEQREC zapis je definiran v SCHEMA-DESCRIPTION.

```
SEQUENTIAL-RECORD name is SEQ001 from record SEQREC
ali
SEQUENTIAL-RECORD SEQ001 SEQREC
```

Dolzina zapisa je lahko od 14 do 510 znakov. Nenaglašena dolzina je 14.

```
RECORD-LENGTH is 120

ACCESS-MODE is OUT
```

IN	Dovoljeno je branje
OUT	Dovoljeno je pisanje
IN-OUT	Dovoljeno je branje in pisanje
EXTEND	Dovoljeno je razširjanje

Zaščita dostopa do sekvenčne datoteke je:

```
ACCESS-PROTECTION is SHARED
```

- SHARED Dovoljen dostop večim programom
- NOSHARED Ekskluzivni dostop enega programa

```
CR-LF characteristic is YES
```

YES Zapis ima CR/LF karakteristiko
NO Zapis nima CR/LF karakteristike

Specifikacija niza je standardna DELTA/M ali DELTA/V specifikacija.

```
FILE name is DBVSHEMA:SEQREC.SEQ
ali
FILE DR1:[100,100]SEQREC.SEQ

END-OF-DESCRIPTION
```


Poglavje 3

3. MANIPULACIJA PODATKOV V IDA BAZI

3.1. UVOD

Jezik manipulacijskih ukazov DML (Data Manipulation Language) omogoča uporabniškim programom branje, spreminjanje, dodajanje in brisanje zapisov v bazi podatkov.

3.1.1. NADREJENI ZAPISI

Dodajanje novega nadrejenega zapisa v zbirko zapisov se naredi s pomočjo funkcije INSG (Insert).

Brisanje obstoječega nadrejenega zapisa se naredi s pomočjo funkcije DELG (Delete). Nadrejeni zapis se lahko zbriše samo takrat, kadar nanj ni vezan noben podrejen zapis.

3.1.2. PODREJENI ZAPISI

Novi podrejeni zapis se lahko doda v set podrejenih zapisov na začetek, na konec ali kjerkoli v setu z uporabo uporabo funkcij INSA (Insert after), INSB (Insert before) in INSG (Insert last).

Funkciji INSA in INSB predpostavljata, da je izvršeno pozicioniranje zapisa, na katerega se nanašata. To pomeni, da mora biti nek obstoječi zapis najprej prebran in zaklenjen (tipično z uporabo GET funkcije), potem pa se lahko izza njega ali pred njim doda novi zapis.

Podrejeni zapis se lahko doda na začetek verige tako, da se z GETG funkcijo prebere in zaklene prvi zapis seta, potem pa se uporabi INSB funkcija.

V nekaterih primerih je podrejeni zapis povezan z večimi nadrejenimi zapisi iz različnih zbirk. Novi podrejeni zapis se lahko doda na konec vseh setov z uporabo funkcije INSG.

Prav tako je možna uporaba funkcij INSA in INSB, vendar se to nanaša samo na set, ki je bil izbran (pri kreiranju podsheme) za ta programski zapis.

Brisanje obstoječega podrejenega zapisa se naredi s pomočjo funkcije DELG. IDA Baza ga avtomatično izbriše iz vseh definiranih setov. Zapis mora biti pred brisanjem zaklenjen (tipično z uporabo GET funkcije).

Sprememba pripadnosti (nadrejenim zapisom) podrejenega zapisa; podrejeni zapis je povezan z enim ali več nadrejenimi zapisi. Kadar je potrebno to pripadnost (povezave) spremeniti, naj se uporabi naslednji postopek:

1. Branje in zaklepanje zapisa z uporabo GET funkcije.

2. Brisanje zapisa z uporabo DELG funkcije.

3. Spreminjanje ključev povezav v programskem vmesnem pomnilniku.

4. Ponovno dodajanje zapisa z uporabo ene od INS funkcij.

3.1.3. ZAKLEPANJE ZAPISOV

Bazo podatkov lahko istočasno uporablja veliko število uporabnikov. V situacijah, kadar želita dva uporabnika spreminjati isti zapis, je ogrožena integriteta podatkov. Naslednji primer ilustrira tako situacijo:

- Uporabnik A prebere zapis "proizvod" in ugotovi stanje zaloge 50 kosov.
- Uporabnik B prebere isti zapis "proizvod" in prav tako ugotovi stanje zaloge 50 kosov.
- Uporabnik A proda 40 kosov proizvoda in vpiše v bazo podatkov stanje zaloge 10.
- Uporabnik B proda 5 kosov proizvoda (od 50) in vpiše v bazo zalogo 45 kosov.
- V resnici znaša zaloga 5 kosov proizvoda, v bazi podatkov pa se nahaja podatek 45, ker je zadnja sprememba pokvarila integriteto podatkov.

Da bi se preprečile take napake, je v IDA Bazo vgrajen poseben zaščitni mehanizem, ki se imenuje zaklepanje zapisov. Ta omogoča posameznim programom, da si zagotovijo ekskluzivno pravico do uporabe enega ali več zapisov v določenem časovnem intervalu. Dokler ta program teh zapisov ne sprosti, jih ostali programi ne morejo spreminjati ali zaklepati, pač pa jih lahko berejo.

Eno od osnovnih pravil za pisanje programov, ki pristopajo v bazo podatkov je, da mora biti zapis, ki bo spremenjen ali izbrisan, pred tem zaklenjen.

Zapis se zaklene tako, da se prebere z GET funkcijo programski zapis, ki ima definirano (to se definira v času kreiranja podsheme) vsaj eno od naslednjih pristopnih pravic: Insert, Delete ali Rewrite.

Zaklepanje zapisov povzroča problem, ki se mu reče "nerazrešljiva situacija". Naslednji primer ilustrira tako situacijo.

- Uporabnik A prebere in zaklene zapis X.
- Uporabnik B prebere in zaklene zapis Y.
- Uporabnik A želi prebrati in zakleniti zapis Y. Ker je zaklenjen mora čakati, dokler ga uporabnik B ne bo sprostil.
- Uporabnik B želi prebrati in zakleniti zapis X. Ker je zaklenjen, mora čakati, dokler ga uporabnik A ne bo sprostil.
- Ker noben od obeh uporabnikov ne more nadaljevati, se tako stanje označi kot nerazrešljiva (deadlock) situacija.

IDA Baza razrešuje take situacije s pomočjo algoritma časovnega intervala. Pri kreiranju operativnega področja se definira časovni interval, znotraj katerega so zapisi lahko zaklenjeni. Ko se programu ta časovni interval izteče, se zaklenjeni zapis avtomatično sprosti, tako da je dosegljiv za druge programe.

Če želi program spremeniti (ali izbrisati) zapis, ki ga predhodno ni zaklenil, IDA Baza vrne statusno sporočilo "DI10" (zapis ni bil predhodno zaklenjen), sprememba pa se zavrne in ne izvrši.

Če želi program zakleniti zapis, ki ga je pred tem že zaklenil nek drug program, se vrne status DI04 (zapis je zaklenjen). V tem primeru lahko program ponavlja GET funkcijo, dokler zapis ni sproščen. Pri takem ponavljanju se avtomatsko izvede zakasnitev, ki traja eno sekundo.

Algoritem zaklepanja se razlikuje v odvisnosti od tega, ali je aktivirano logiranje logičnih transakcij ali ne.

Logiranje logičnih transakcij je aktivirano: časovni interval velja za celo logično transakcijo, to je med dvema terminatorjema logične transakcije. Pojem logične transakcije je bolj podrobno obravnavan v 6. poglavju.

Po izteku časovnega intervala se lahko zgodi, da nek drug program prevzame ekskluzivno pravico nad zapisom, ki je bil predmet te logične transakcije. V takem primeru IDA Baza razveljavi celotno logično transakcijo in vrne statusno sporočilo LG26 (logična transakcija je bila abortirana). Program mora ponavljati celo logično transakcijo. V kolikor drugi programi ne poizkušajo

zaklepati zapisov te logične transakcije. nima iztek časovnega intervala nobenega vpliva na izvajanje te logične transakcije.

Uspešno izvršen ukaz CONFRM izvede vpis vseh spremenjenih zapisov v fizično bazo podatkov in sprosti (odklene) vse zaklenjene zapise. Zaklenjeni zapisi se sprostijo tudi po izvajanju ukazov CANCEL in BYE. Seveda se logična transakcija v tem primeru razveljavi.

Zaradi zagotavljanja integritete baze podatkov IDA Baza zaklepa tudi nekatere zapise, ki so v povezavi s kakšnim zapisom, ki ga je neki program zaklenil. To se zgodi na primer z nadrejenim zapisom, kadar se na začetek ali konec njegovega pripadajočega seta doda nov podrejen zapis. Tako implicitno zaklenjeni zapisi se obravnavajo kot del logične transakcije.

Program lahko zaklene poljubno število zapisov iz vsake zbirke zapisov. Število vseh zaklenjenih zapisov v nekem trenutku je omejeno s "Skupnim številom vseh logičnih zapisov", ki se defira pri kreiranju operativnega področja. Če se ta limita preseže, vrne IDA Baza status DE18 v program, logična transakcija pa je razveljavljena. Ker je zaklepanje zapisov dinamično, se lahko logična transakcija ponavlja, dokler ni uspešno končana.

Logiranje transakcij ni aktivirano: Kadar logiranje transakcij ni aktivirano lahko program naekrat zaklene samo en zapis iz vsake zbirke. Zaklepanje naslednjega zapisa iste logične zbirke pomeni hkrati sproščanje trenutno zaklenjenega zapisa.

Čas zaklepanja se nanaša na vsak zaklenjen zapis posebej.

CONFRM in CANCEL se v tem primeru ignorirata, IDA Baza pa vrne status ****.

3.1.4. ZAŠČITA PODATKOV S POMOČJO GESLA

IDA Baza omogoča zaščito dostopa do podatkov s pomočjo gesla podsheme. V kolikor se uporabi ta mehanizem, mora program vprašati za geslo in potem odgovor navesti kot parameter pri ukazu "HELLO". Če je geslo nepravilno, je ukaz "HELLO" neuspešen (Status LG02) in program ne mora dostopati do baze podatkov.

Program lahko uporablja več podshem in vsaka od njih ima lahko svoje geslo. Vendar je v nekem trenutku lahko aktivna ena sama poshema. Program mora izvesti ukaz "BYE" preden z ukazom "HELLO" aktivira drugo podshemo.

Kadar gesla niso potrebna, se pri kreiranju podsheme geslo ne navaja. V tem primeru je geslo enako prvim šestim znakom imena sheme.

3.2. UKAZI IN FUNKCIJE MANIPULACIJSKEGA JEZIKA IDA BAZE

3.2.1. DML UKAZI

DML ukazi so realizirani z uporabo "CALL" ukazov, ki se uporabljajo v vseh višjih programskih jezikih.

HELLO in BYE: Ta ukaza označujeta začetek (HELLO) in konec (BYE) operacij na bazi podatkov.

DBMIO in SEQIO: Ta dva ukaza se uporabljata za izvajanje vhodno/izhodnih funkcij na zapisih baze podatkov (DBMIO) in sekvenčnih datotekah (SEQIO).

CONFRM in CANCEL: Ta dva ukaza se uporabljata za potrditev logičnih transakcij (CONFRM) oziroma za njihovo razveljavljanje (CANCEL). Kadar program spreminja zapise z DBMIO, te spremembe niso fizično implementirane na diskih vse do uspešne izvršitve CONFRM ukaza. Če je namesto CONFRM ukaza uporabljen ukaz CANCEL, so te spremembe razveljavljene. CONFRM in CANCEL se nanašata samo na bazo podatkov in ne tudi na sekvenčne datoteke.

LOGDAT: Ta ukaz se uporablja za beleženje, povsem pod kontrolo uporabniškega programa, neodvisno od beleženja IDA Baze. Če je potrebno, se LOGDAT ukaz lahko uporabi za beleženje vhoda.

3.2.2. DBMIO FUNKCIJE

Pri ukazu DBMIO se uporabljajo funkcije, ki natančno opisujejo vsak ukaz. Nekatere od njih zahtevajo, da je bil zapis predhodno zaklenjen npr. z branjem programskega zapisa, ki ima definirano eno od naslednjih pristopnih pravic: INSERT, DELETE ali UPDATE.

- Branje zapisov iz baze podatkov

GETD: Funkcija "GET Direct" se uporablja za direktno branje podrejenega zapisa na osnovi DB ključa, ki si ga je program predhodno shranil.

GETG: Funkcija "GET General" se uporablja za branje nadrejenega zapisa na osnovi ključa zapisa in za branje podrejenih zapisov, kot se ti pojavljajo v določenem setu.

GETP: Funkcija "GET Physical" se uporablja za branje zapisov neodvisno od logične strukture in sicer v takem vrstnem redu, kot so zapisi fizično zapisani na diskih.

GETR: Funkcija "GET Reverse" se uporablja za branje podrejenih zapisov v obratnem vrstnem redu (od zadaj naprej), ko se ti nahajajo v setu.

- **Azuriranje predhodno zaklenjenih zapisov**

RWRG: Funkcija "ReWRite General" se uporablja za azuriranje zapisov, ki so bili predhodno prebrani, zaklenjeni in spremenjeni.

- **Dodajanje novih zapisov v bazo podatkov**

INSG: Funkcija "INSert General" se uporablja za dodajanje novih nadrejenih zapisov na osnovi ključa zapisa ali za dodajanje podrejenega zapisa na konec vseh setov.

INSA: Fukcija "INSert After" se uporablja za dodajanje novih podrejenih zapisov izza prebranega in zaklenjenega zapisa v nekem setu.

INSB: Funkcija "INSert Before" se uporablja za dodajanje novih podrejenih zapisov pred prebrani in zaklenjeni zapis v nekem setu.

- **Brisanje zapisov iz baze podatkov**

DELG: Funkcija "DELete General" se uporablja za brisanje nadrejenih in podrejenih zapisov, ki so bili predhodno prebrani in zaklenjeni.

3.2.3. SEQIO FUNKCIJE

Tudi pri ukazu SEQIO se uporabljajo funkcije, ki natančno opisujejo ukaz.

- **Funkcije na nivoju datotek**

SCLO: Funkcija "Sequential CLOse" prekine programski dostop do sekvenčne datoteke in izvede logično zapiranje datoteke.

SOPE: Funkcija "Sequential OPEn" vzpostavi programski dostop do sekvenčne datoteke in izvede logično odpiranje datoteke.

SRWD: Funkcija "Sequential ReWinD" izvede pozicioniranje na prvi zapis sekvenčne datoteke (npr. previjanje magnetnega traku na začetek). Pred uporabo funkcije SRWD mora biti uspešno izvedena funkcija SOPE (datoteka mora biti "odprta").

- **Funkcije na nivoju zapisov**

SGET: Funkcija "Sequential GET" se uporablja za branje (naslednjega) zapisa sekvenčne datoteke.

SINS: Funkcija "Sequential INSert" se uporablja za dodajanje novega zapisa na konec sekvenčne datoteke.

SRWR: Funkcija "Sequential ReWRite" se uporablja za azuriranje predhodno prebranega in spremenjenega zapisa. Dolžina azuriranega zapisa mora ostati nespremenjena.

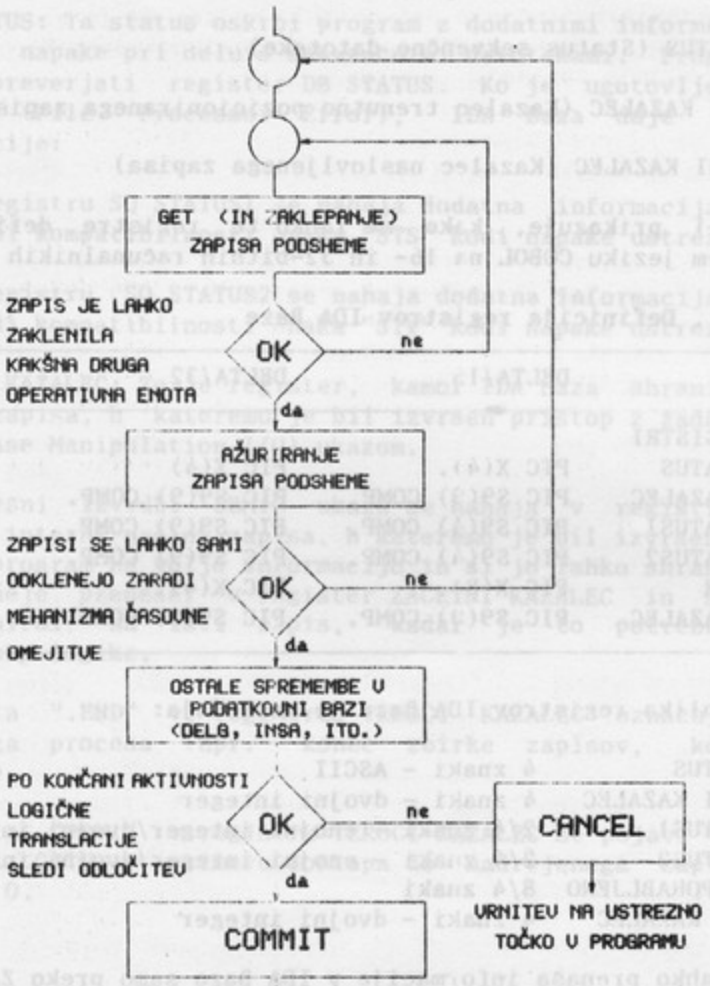
3.2.4. SPLOŠNI NAPOTKI

- Pred vsako spremembo zapisa (dodajanje brisanje, ažuriranje) v bazi podatkov mora biti ta zapis predhodno prebran in zaklenjen, da se prepreči ostalim programom možnost spreminjanja istega zapisa. Pristopna pravica za spreminjanje zapisa se deklarira v času definiranja podsheme.

- Posamezen zapis je lahko večkratno definiran v isti podshemi v obliki večih programskih zapisov. Tipično je, da se pristopne pravice teh programskih zapisov razlikujejo.

- Enkrat so definirane samo pravice za branje zapisa, drugič pa tudi spreminjanje. To omogoča najmanjše prekrivanje zaklepanja zapisov z drugimi uporabniki in povečuje performanse sistema.

Slika 3-1 prikazuje splošno proceduro za kontrolo zaklenjenih zapisov.



Slika 3-1. Procedura za kontrolo zaklenjenih zapisov

- Zaradi doseganja boljših performans, je bazi podatkov namenjen prostor na disku predhodno formatiran s programom DBF (Data Base Formatting). Ta prostor je formatiran v obliki celic, ki bodo sprejele podatke. Podakovni prostor je inicializiran s presledki (spaces). INSert funkcija se nanaša samo na podatkovne elemente programskega zapisa definiranega v podshemi. Zato se priporoča, da programski zapis, ki se uporablja za dodajanje novih zapisov, vsebuje vse podatkovne elemente zapisa. Vsa numerična polja, ki v času dodajanja zapisa ne vsebujejo podatkov je koristno inicializirati na ničle, s čimer se da izogniti konverzijskim problemom pri kasnejši uporabi.

Registri IDA Baze: Kontrolne informacije med IDA Bazo in programom se izmenjujejo v delovnem spominu (working storage) programa, ki se imenuje "DB REGISTRI".

Obstajajo štirje registri IDA Baze:

- DB STATUS (Status baze podatkov)
- SQ STATUS (Status sekvenčne datoteke)
- TEKOČI KAZALEC (Kazalec trenutno pozicioniranega zapisa)
- ZAČETNI KAZALEC (Kazalec naslovljenega zapisa)

Tabela 3-1 prikazuje, kako se lahko te registre definira v programskem jeziku COBOL na 16- in 32-bitnih računalnikih DELTA.

Tabela 3-1. Definicija registrov IDA Baze

REGISTER		DELTA/16	DELTA/32
01	DB-REGISTRI		
05	DB-STATUS	PIC X(4).	PIC X(4)
05	ZAC-KAZALEC	PIC S9(9) COMP.	PIC S9(9) COMP.
05	SQ-STATUS1	PIC S9(4) COMP.	PIC S9(9) COMP.
05	SQ-STATUS2	PIC S9(4) COMP.	PIC S9(9) COMP.
05	FILLER	PIC X(8).	PIC X(4).
05	TEK-KAZALEC	PIC S9(9) COMP.	PIC S9(9) COMP.

Splošna oblika registrov IDA Baze je naslednja:

- DB STATUS 4 znaki - ASCII
- ZAČETNI KAZALEC 4 znaki - dvojni integer
- SQ STATUS1 2/4 znaki - enojni integer/dvojni integer
- SQ STATUS2 2/4 znaki - enojni integer/dvojni integer
- SE NEUPORABLJENO 8/4 znaki
- TEKOČI KAZALEC 4 znaki - dvojni integer

Program lahko prenaša informacije v IDA Bazo samo preko ZAČETNEGA KAZALCA. Vsi drugi registri se uporabljajo za prenos informacij iz IDA Baze v program.

DB STATUS: Ta register uporablja IDA Baza za indikacijo pravilnosti ali napake pri izvajanju posameznega ukaza, ki ga je zahteval program. Kadar se je ukaz izvedel pravilno, je vrnjen DB STATUS ****. V nekaterih primerih se je ukaz sicer izvedel pravilno, kljub temu pa je vrnjeno opozorilo v obliki ****.

Obstajajo trije tipi DB STATUSOV:

Statusna koda	Pomen
****	Ukaz je uspešno izvršen
**xx	Opozorilo
xxxx	Napaka

Zelo važno opozorilo:

ZA VSAKIM KLICEM NA BAZO SE MORA OBVEZNO TESTIRATI DB STATUS. ČE SE POJAVI NEPRIČAKOVANI STATUS, SE GA NE SME IGNORIRATI!

SQ STATUS: Ta status oskrbi program z dodatnimi informacijami v primeru napake pri delu s sekvenčnimi datotekami. Program mora vedno preverjati register DB STATUS. Ko je ugotovljen status "FP02" (File Processor Error), IDA Baza daje naslednje informacije:

- V registru SQ STATUS1 se nahaja dodatna informacija, ki je zaradi kompatibilnosti enaka "STS" kodi napake ustreznega RMS.
- V registru SQ STATUS2 se nahaja dodatna informacija, ki je zaradi kompatibilnosti enaka "STV" kodi napake ustreznega RMS.

TEKOČI KAZALEC: To je register, kamor IDA Baza shrani interni naslov zapisa, h kateremu je bil izvršen pristop z zadnjim DBMIO (Data Base Manipulation I/O) ukazom.

Po uspešni izvedbi DBMIO ukaza se nahaja v registru TEKOČI KAZALEC interni naslov zapisa, h kateremu je bil izvršen pristop. Tu ima program na voljo informacijo in si jo lahko shrani, da bi jo pozneje prenesel v register ZAČETNI KAZALEC in se s tem pozicioniral na isti zapis, kadar je to potrebno zaradi programske logike.

Konstanta ".END" v registru TEKOČI KAZALEC označuje konec logičnega procesa (npr. konec zbirke zapisov, konec seta zapisov).

Konstanta "PRVI" v registru TEKOČI KAZALEC se pojavi samo pri funkciji GETP v primeru dostopa do nadrejenega zapisa z DB ključem 0.

V zvezi s TEKOČIM KAZALCEM je treba omeniti pojem "POZICIONIRANI ZAPIS". S tem je mišljen zapis, h kateremu je bil izvršen zadnji pristop, na primer zapis, ki je bil zadnji prebran. Pozicioniranje pa se da izvesti tudi tako, da se v nekem trenutku zapomni TEKOČI KAZALEC in se ga pri izvedbi neke druge funkcije (tipično pri GETD funkciji) prenese v register ZAČETNI KAZALEC. S tem se izvrši pozicioniranje zapisa.

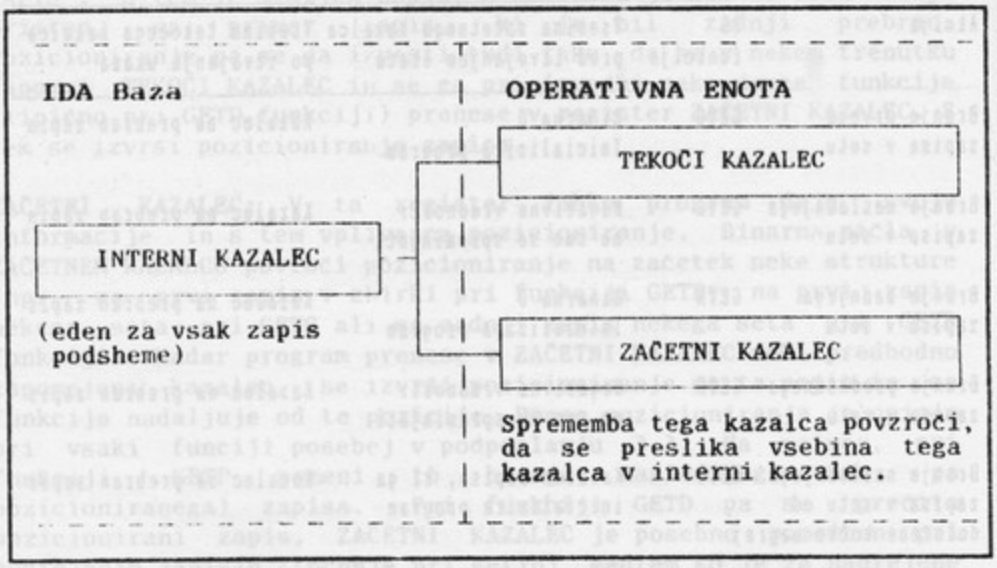
ZAČETNI KAZALEC: V ta register lahko program daje svoje informacije in s tem vpliva na pozicioniranje. Binarna ničla v ZAČETNEM KAZALCU povroči pozicioniranje na začetek neke strukture (npr. na prvi zapis v zbirki pri funkciji GETP, na prvi zapis nekega seta pri GETG ali na zadnji zapis nekega seta pri GETR funkciji). Kadar program prenese v ZAČETNI KAZALEC neki predhodno zapomnjeni kazalec, se izvrši pozicioniranje na ta zapis in se funkcija nadaljuje od te pozicije. Pomen pozicioniranja je opisan pri vsaki funciji posebej v podpoglavju 3.3. Na primer, pri funkciji GETP pomeni to branje naslednjega (ne pa pozicioniranega) zapisa. Pri funkciji GETD pa se prečita pozicionirani zapis. ZAČETNI KAZALEC je posebno pomemben pri podrejenih zapisih (točneje pri setih), medtem ko je za nadrejene manj pomemben (ker so ti že tako enosmiselno identificirani z vrednostjo ključa zapisa).

Tabela 3-2. Uporaba tekočega in začetnega kazalca

Akcija	DB funkcija	Vsebina začetnega kazalca pred izvajanjem ukaza	Vsebina tekočega kazalca po izvajanju ukaza
Branje prvega zapisa v setu	GETG	Binarna 0 Inicializira program	Kazalec na prebran zapis
Branje naslednjega zapisa v setu	GETG	Negativna vrednost: ne sme se spreminjati	Kazalec na prebran zapis
Branje zadnjega zapisa v setu	GETR	Binarna 0 Inicializira program	Kazalec na prebran zapis
Branje predhodnega zapisa v setu	GETR	Negativna vrednost: ne sme se spreminjati	Kazalec na prebran zapis
Branje naslednjega zapisa v setu od določene točke naprej	GETG	Kazalec zapisa, ki ga inicializira program	Kazalec na prebran zapis
Branje naslednjega zapisa v setu od določene točke nazaj	GETR	Kazalec zapisa, ki ga inicializira program	Kazalec na prebran zapis
Direktno branje zapisa	GETD	Kazalec zapisa, ki ga inicializira program	Kazalec na prebran zapis
Sprememba zadnjega prebranega zapisa	RWRG *	Negativna vredost: ne sme se spreminjati	Kazalec na spremenjeni zapis
Brisanje zadnjega prebranega zapisa	DELG *	Negativna vrednost: ne sme se spreminjati	0 ali kazalec na predhodni zapis
Dodajanje novega zapisa na konec vseh setov	INSG	Se ignorira	Kazalec na dodani zapis
Dodajanje novega zapisa izza zadnjega prebranega zapisa	INSA *	Negativna vrednost: ne sme se spreminjati	Kazalec na dodani zapis
Dodajanje novega zapisa pred zadnji prebrani zapis	INSB *	Negativna vrednost: ne sme se spreminjati	Kazalec na dodani zapis

* Zapis mora biti zaklenjen pred uporabo funkcije

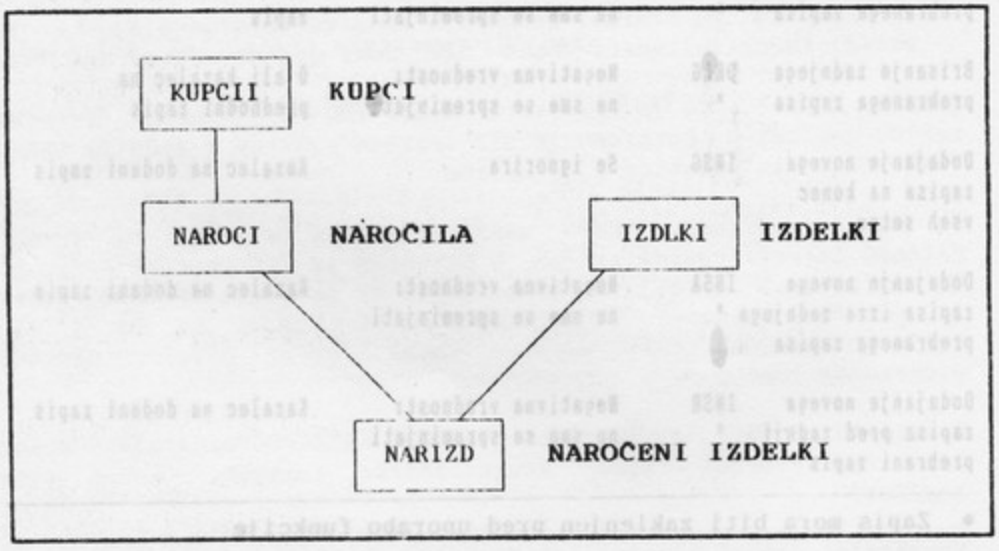
Slika 3-2 pojasnjuje preslikavo kazalcev IDA Baze.



Slika 3-2. Preslikava kazalcev IDA Baze

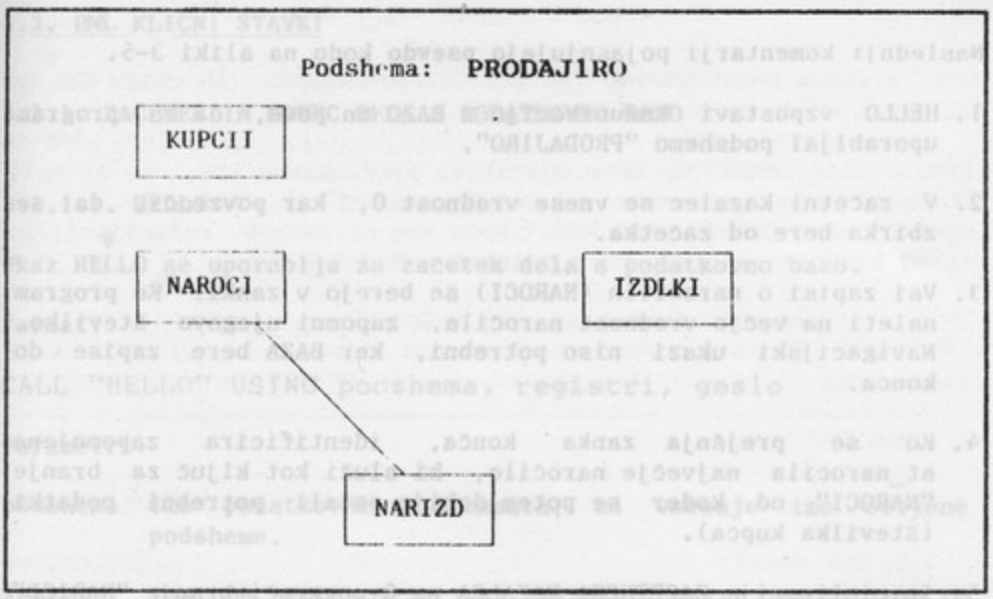
NAVIGACIJA V BAZI PODATKOV

Uporabo DML ukazov se da najbolje razložiti na konkretnem primeru. Tak primer prikazuje slika 3-3.



Slika 3-3. Primer baze podatkov

Naloga je izpis največjega naročila iz baze podatkov, ki jo prikazuje slika 3-3. Najprej je treba kreirati podshemo, ki ima samo "read only" dovoljene pristopne pravice, ker tak program ne bo izvajal nobenih sprememb v bazi podatkov.



Slika 3-4. Podshema "prodajIro"

Pri direktnem čitanju po ključu je potrebno definirati samo eno povezavo (npr. med "NAROCI" in "NARIZD"), "NAROCI" bo prebran v celoti s funkcijo "GETP".

Sledi psevdo koda, ki izvršuje opisano nalogo. Ker je to pač psevdo koda, vsebuje mnogo poenostavitev. Prava programska koda mora vedno po izvajanju operacij na bazi podatkov testirati status.

```
HELLO PRODAJIRO                                  Komentar 1

ZACETNI_KAZALEC = 0                              Komentar 2
GETP NAROCI
REMEMBER ST_NAROCI, VREDNOST

DO WHILE STATUS NOT "END."                       Komentar 3
   IF VREDNOST > ZAPOMNJENA VREDNOST
      THEN REMEMBER NOVO ST_NAROCI, VREDNOST
   GETP NAROCI
END OF DO WHILE

NAROCI_OWNKEY = ZAPOMNJENA ST_NAROCI             Komentar 4
GETG NAROCI
KUPCII_OWNKEY = ST_KUPCA IZ NAROCI
GETG KUPCII
PRINT NAROCI HEADING

NARIZD_SETKEY = ZAPOMNJENI ST_NAROCI             Komentar 5
ZACETNI_KAZALEC = 0
GETG NARIZD
DO WHILE DB_STATUS NOT EQUAL "END."
   IZDLKI_OWNKEY = ST_IZDELKA IZ NARIZD
   GETG IZDLKI
   PRINT DETAILNA VRSTICA WITH OPIS_IZDELKA
   GETG NARIZD
END OF DO WHILE

PRINT NAROCI TRAILER                             Komentar 6
BYE
STOP
```

Slika 3-5. Primer psevdo kode

Naslednji komentarji pojasnjujejo psevdo kodo na sliki 3-5.

1. HELLO vzpostavi komunikacijo z BAZO in pove, da bo program uporabljal podshemo "PRODAJIRO".

2. V začetni kazalec se vnese vrednost 0, kar povzroči, da se zbirka bere od začetka.

3. Vsi zapisi o naročilih (NAROCI) se berejo v zanki. Ko program naleti na večjo vrednost naročila, zapomni njegovo številko. Navigacijski ukazi niso potrebni, ker BAZA bere zapise do konca.

4. Ko se prejšnja zanka konča, identificira zapomnjena st_narocila največje naročilo, ki služi kot ključ za branje "NAROCI" od koder se potem dobijo ostali potrebni podatki (številka kupca).

5. Inicializacija ZACETNEGA KAZALCA na 0 povzroči branje "NARIZD" (naročenih izdelkov) od začetka (za to konkretno naročilo). BAZA izvaja navigacijo s pomočjo SETA. V zanki se berejo posamezne pozicije naročenih izdelkov in izpisujejo se detajlne vrstice z opisom izdelkov. Ko je SETA konec, se zanka konča.

6. Po potrebi se izpiše še zaključek poročila. Ukaz BYE pomeni odjavo iz BAZE, ki sprosti resurse za druge uporabnike.

Ker ni bila izvedena nobena sprememba na bazi podatkov, nista potrebna ukaza CONFRM oziroma CANCEL.

3.3. DML KLICNI STAVKI

3.3.1. ZAČETEK IN KONEC DELA S PODATKOVNO BAZO

3.3.1.1. HELLO

Ukaz HELLO se uporablja za začetek dela s podatkovno bazo.

Format:

```
CALL "HELLO" USING podshema, registri, geslo
```

Parametri:

podshema Ime podatkovnega elementa, ki vsebuje ime željene podsheme.

registri Ime DB registrov; vsi naslednji ukazi uporabljajo te registre.

geslo Ime podatkovnega elementa, ki vsebuje geslo željene podsheme.

Akcija:

Ta ukaz vzpostavi komunikacijo med IDA Bazo in programom. Če je ukaz uspešno izveden, IDA Baza vrne status štiri zvezdice (****).

Program lahko uporablja samo eno podshemo naenkrat za dostop do podatkovne baze. Po ukazu HELLO, mora program izvesti BYE preden lahko dostopi do podatkovne baze preko druge podsheme.

Primeri za uporabo funkcije: HELLO v DODATKU D

Tip zapisa	označba v primeru
1. CELA PODSHEMA	(2)

3.3.1.2. BYE

Ukaz BYE se uporablja za zaključek dela s podatkovno bazo.

Format:

```
CALL "BYE"
```

Parametri: Brez.

Akcija:

BYE prekine povezavo med IDA Bazo in programom. Po ukazu BYE je veljaven edino ukaz HELLO.

Pazi: Ukaz COMMIT se mora uporabiti pred ukazom BYE, če se želi ohraniti vse spremembe od zadnjega COMMIT do BYE. Če se ne uporabi ukaz COMMIT pred BYE, bodo vse spremembe razveljavljene (glej komentar v priloženem primeru - Dodatek D).

Primeri za uporabo funkcije: BYE v DODATKU D

Tip zapisa	Označba v primeru
1. CELA PODSHEMA	(1)

3.3.2. DML UKAZI

Naslednji ukazi so našteti v abecednem zaporedju:

- CANCEL
- COMMIT
- DBMIO
- LOGDAT
- SEQIO

3.3.2.1. CANCEL

Ukaz CANCEL se uporablja za razveljavitev vseh odloženih sprememb od zadnjega COMMIT, CANCEL, ali HELLO ukaza.

Format:

```
CALL "CANCEL" [USING sporočilo]
```

Parametri:

sporočilo Ime podatkovnega elementa, ki vsebuje indentifikacijo zadnjega COMMIT ukaza (30 znakov).

Akcija:

Vse DML funkcije, ki modificirajo podatkovno bazo in ki so izvedene izza zadnjega COMMIT, CANCEL, ali HELLO ukaza so razveljavljene. Efekt je isti, kot da se niso izvajale. Vsi rezervirani zapisi so sproščeni.

Primeri za uporabo funkcije: CANCEL v DODATKU D

Tip zapisa	Sklicevaje na primer	Komentar
1. CELA PODSHEMA	(31)	na BYE se izvede avtomatsko

3.3.2.2. COMMIT

Ukaz COMMIT se uporablja za potrditev vseh odloženih sprememb. Sinonimni ukaz je CONFRM.

Format:

```
CALL "COMMIT" [USING :poročilo]
```

Parametri:

sporočilo Ime podatkovnega elementa, ki vsebuje indentifikacijo zadnjega COMMIT ukaza (30 znakov).

Akcija:

Vse odložene DML funkcije od zadnjega CANCEL, COMMIT, ali HELLO so potrjene. Vsi rezervirani zapisi se sprostijo.

Primeri za uporabo funkcije: COMMIT v DODATKU D

Tip zapisa	Označba v primeru
1. CELA PODSHEMA	(7),(10),(19),(33),(39)

3.3.2.3. DBMIO

Ukaz DBMIO se uporablja za izvajanje V/I funkcij na podatkovni bazi.

Format:

```
CALL "DBMIO" USING funkcija,zapis,V/Ipodr.,[ključ]
```

Parametri:

Funkcija v podatkovnem elementu "funkcija" se izvede s programskim zapisom navedenim v podatkovnem elementu "zapis" Če funkcija uporablja ključ, mora biti naveden v podatkovnem elementu "ključ". Če funkcija dodaja ali spreminja zapis, potem mora biti nova vsebina v podatkovnem elementu "V/Ipodr.". Če funkcija čita programski zapis, bo ta prebran v "V/Ipodr.". Status se vrača v DB STATUS registru. Uspešno izvajanje ukaza/funkcije je označeno z vrednostjo štirih zvezdic (****).

Akcija:

Spisek devetih funkcij za ukaz DBMIO.

Funkcija	Akcija
DELG	Brisanje trenutnega zapisa
GETD	Citanje zapisa z DB ključem
GETG	Citanje zapisa s ključem zapisa ali seta
GETP	Citanje naslednjega zapisa v fizični sekvenci
GETR	Citanje verige zapisov s ključem seta v nasprotni smeri
INSA	Dodajanje zapisa s ključem seta izza trenutnega
INSB	Dodajanje zapisa s ključem seta pred trenutnega
INSG	Dodajanje zapisa
RWRG	Azuriranje trenutnega zapisa

Natančni opisi delovanja posameznih funkcij so v podpoglavju 3.3.3.

3.3.2.4. LOGDAT

Ukaz LOGDAT se uporablja za pisanje informacij na log datoteko, ki je skupna vsem programom.

Format:

```
CALL "LOGDAT" USING V/Ipodr.,dolzina
```

Akcija:

Vsebina podatkovnega elementa "V/Ipodr." v dolzini, ki je v podatkovnem elementu "dolzina", se doda fiksnemu delu in zapiše v datoteko "DBMINPDAT.LOG".

Izgled fiksnega dela, ki ga doda IDA Baza:

Element	DELTA/M	DELTA/V
Task ID (Process ID)	4	4
Stevilka terminala	2	8
UIC	2	6
Datum	6	6
Cas	6	8
Rezervirano	12	6
	32	38

Podatki iz uporabnikovega programa se nahajajo takoj izza fiksnega dela.

Primer uporabe funkcije LOGDAT v dodatku D pod oznako (4).

3.3.2.5. SEQIO

Ukaz SEQIO se uporablja za delo s sekvenčnimi datotekami.

Format:

```
CALL "SEQIO" USING funkcija,zapis,V/Ipodr.,dolžina
```

Parametri:

Sekvenčna V/I funkcija navedena v podatkovnem elementu "funkcija" se izvede s programskim zapisom v podatkovnem elementu "zapis". Če je želena funkcija dodajanja ali ažuriranja, potem mora biti vsebina zapisa v podatkovnem elementu "V/Ipodr.". Funkcija čitanja vrne vsebino zapisa v podatkovni element "V/Ipodr.". Pri dodajanju novega zapisa mora biti v podatkovnem elementu "dolžina" njegova dolžina. Pri čitanju IDA Baza vrne v ta element dolžino čitanega zapisa. Pri ažuriranju zapisa se dolžina ne sme spremeniti.

Akcija:

Spisek šestih funkcij za ukaz SEQIO.

Funkcija	Akcija
SOPE	Odpiranje sekvenčne datoteke.
SCLO	Zapiranje sekvenčne datoteke.
SGET	Čitanje sekvenčne datoteke.
SINS	Dodajanje v sekvenčno datoteko.
SRWR	Ažuriranje sekvenčne datoteke.
SRWD	Pozicioniranje odprte sekvenčne datoteke na začetek.

Natančni opisi vsake posamezne funkcije so v podpoglavju 3.3.4.

3.3.3. DBMIO FUNKCIJE

3.3.3.1. DELG

Uporabi DELG za:

- Brisanje rezerviranega nadrejenega zapisa.
- Brisanje POZICIONIRANEGA(*) in rezerviranega podrejenega zapisa.

Format: Funkcija DBMIO ukaza.

(*) POZICIONIRAN zapis je zapis, ki je bil zadnji dosežen v okviru navedenega programskega zapisa ali zapis katerega kazalec se nahaja v registru ZAČETNI KAZALEC.

Parametri:

funkcija DELG

zapis Ime programskega zapisa, ki naj se briše.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Nadrejeni zapis: Ključ zapisa.
Podrejeni zapis: Ključ seta.

Akcija:

Če parameter "zapis" označuje nadrejeni programski zapis, potem DELG funkcija dostopi do pravega zapisa z navedenim ključem zapisa in ga briše.

Če parameter "zapis" označuje podrejeni programski zapis, potem DELG funkcija dostopi do POZICIONIRANEGA(*) zapisa. Predhodni zapis v setu postane POZICIONIRANI(*) zapis in njegov kazalec se prenese v interni kazalec za programski zapis in v register TEKOČI KAZALEC.

To je zelo pomembno, če se uporabi funkcija GETR izza funkcije DELG. GETR funkcija dostopa do predhodnega zapisa v setu in na ta način izpusti POZICIONIRANI(*) zapis. V tem primeru je potrebno uporabiti funkcijo GETD namesto GETR.

Zapis mora biti rezerviran preden se izvaja DELG funkcija.

Primeri za uporabo funkcije: DELG v DODATKU D

Tip zapisa	Označba v primeru	Komentar
1. NADREJENI		enako kot podrejeni
2. PODREJENI	(36)	
3. KOMB. S SETOM	(38)	enako kot podrejeni
4. KOMB. BREZ SETA		enako kot podrejeni

3.3.3.2. GETD

Uporabi GETD za:

- Čitanje podrejenega zapisa, katerega kazalec se nahaja v registru ZAČETNI KAZALEC.
- Rezervacijo čitanega zapisa

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija GETD

zapis Ime programskega zapisa, ki naj se čita.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Se ne upošteva.

Akcija:

Programski zapis naveden v parametru "zapis" in s kazalcem navedenim v registru ZAČETNI KAZALEC bo prebran v področje navedeno v parametru "V/Ipodr.". Kazalec prebranega zapisa se vrne v registru TEKOČI KAZALEC, če je operacija čitanja uspešna.

Funkcija GETD je namenjena direktnem čitanju podrejenih zapisov katerih kazalci so znani. Funkcija GETD izvede tudi rezervacijo čitanega zapisa, če programski zapis tako določa.

Primeri za uporabo funkcije: GETD v DODATKU D

Tip zapisa	Sklicevanje na primer	Komentar
1. NADREJENI		ni mogoče-izvede se GETG
2. PODREJENI	(28)	
3. KOMB. S SETOM	(14)	
4. KOMB. BREZ SETA		ni mogoče-izvede se GETG

3.3.3.3. GETG

Uporabi GETG za:

- Čitanje nadrejenega zapisa s ključem zapisa.
- Zaporedno branje seta podrejenih zapisov, katerih nadrejeni zapis je indentificiran s set ključem.
- Rezervacijo nadrejenega ali podrejenega zapisa, če programski zapis tako določa.

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija GETG

zapis Ime programskega zapisa, ki naj se čita.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Nadrejeni zapis: Ključ zapisa.
Podrejeni zapis: Ključ seta.

Akcija:

Če parameter "zapis" označuje nadrejeni programski zapis, potem GETG funkcija prečita zapis s ključem zapisa, ki je naveden v parametru "ključ" v področje, ki je navedeno v parametru "V/Ipodr.".

Če parameter "zapis" označuje podrejeni programski zapis, potem GETG funkcija prečita zapis v setu, katerega nadrejeni zapis je indentificiran s ključem seta, ki je naveden v parametru "ključ". Vsebina registra ZAČETNI KAZALEC določa kateri zapis v setu se bo prečital in prenesel v področje, ki je navedeno v parametru "V/Ipodr.".

Če je vsebina registra ZAČETNI KAZALEC ničla, to pomeni, da se bo čital prvi zapis v setu in njegov kazalec bo po uspešno izvedeni funkciji v registru TEKOČI KAZALEC, njegova negativna vrednost pa v registru ZAČETNI KAZALEC. Dokler je v registru ZAČETNI KAZALEC negativna vrednost bodo naslednje GETG funkcije zaporedno čitale zapise v setu. Konec seta je označen z znaki "END." v registru DB STATUS.

GETG funkcija je namenjena čitanju in rezervaciji zapisov, če programski zapis tako določa. Če se želi čitati podrejene zapise od začetka seta, potem se mora registru ZAČETNI KAZALEC prirediti vrednost nič (binarna ničla). Možno je tudi začeti branje nekje na sredini seta, na ta način, da se registru ZAČETNI KAZALEC priredi vrednost kazalca zapisa. Ta način je možen samo takrat kadar se zapomni kazalec pri predhodni obdelavi istega seta.

Večinoma se register ZAČETNI KAZALEC uporablja za navodilo IDA Bazi da mora začeti čitati od začetka seta. Register ZAČETNI KAZALEC je namenjen samo vpisovanju vrednosti, zato je njegova vsebina za uporabnika nepomembna.

Primeri za uporabo funkcije: GETG v DODATKU D

Tip zapisa	Označbe v primeru
1. NADREJEN	(5),(8),(12),(17),(20),(23),(25),(30)
2. PODREJEN	(21),(26),(27),(35)
3. KOMB. S SETOM	(13),(37)
4. KOMB. BREZ SETA	(11),(22),(24),(34)

3.3.3.4. GETP

Uporabi GETP za:

- Čitanje nadrejenega ali podrejenega zapisa zaporedno iz

zbirke zapisov. Zaporedje čitanja je fizično, kar pomeni, da se zapisi čitajo v istem vrstnem redu, kot so fizično zapisani.

- Rezervacijo čitanega zapisa

Format: Funkcija DBMIO ukaza.

Parametri:
funkcija GETP

zapis Ime programskega zapisa, ki naj se čita.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Se ne upošteva.

Akcija:
Programski zapis naveden v parametru "zapis" bo prebran v področje navedeno v parametru "V/Ipodr.". Če je register ZAČETNI KAZALEC nič (binarna ničla), potem se bo čital prvi zapis v zbirki. Kazalec prebranega zapisa se vrne v registru TEKOČI KAZALEC, če je operacija čitanja uspešna. Naslednja GETP funkcija bo čitala drugi zapis v zbirki in tako do konca zbirke, ki je označena z znaki "END." v registru DB STATUS.

Funkcija GETP izvede tudi rezervacijo čitanega zapisa, če programski zapis tako določa.

Primeri za uporabo funkcije: GETP v DODATKU D

Tip zapisa	Označbe v primeru	Komentar
1. NADREJENI	(40)	
2. PODREJENI	(41)	
3. KOMB. S SETOM		enako kot podrejeni
4. KOMB. BREZ SETA		enako kot nadrejeni

3.3.3.5. GETR

Uporabi GETR za:

- Zaporedno branje seta podrejenih zapisov v obratni smeri, katerih nadrejeni zapis je indentificiran s set ključem.
- Rezervacijo podrejenega zapisa, če programski zapis tako določa.

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija GETR

zapis Ime programskega zapisa, ki naj se čita.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Ključ seta.

Akcija:

GETR funkcija prečita zapis v setu, katerega nadrejeni zapis je indentificiran s ključem zapisa, ki je naveden v parametru "ključ". Vsebina registra ZAČETNI KAZALEC določa kateri zapis v setu se bo prečital in prenesel v področje, ki je navedeno v parametru "V/Ipodr.".

Če je vsebina registra ZAČETNI KAZALEC nič (binarna ničla), to pomeni, da se bo čital zadnji zapis v setu in njegov kazalec bo po uspešno izvedeni funkciji v registru TEKOČI KAZALEC, njegova negativna vrednost pa v registru ZAČETNI KAZALEC. Dokler je v registru ZAČETNI KAZALEC negativna vrednost bodo naslednje GETR funkcije zaporedno čitale zapise v setu od konca proti začetku. Začetek seta je označen z znaki "END." v registru DB STATUS.

GETR funkcija je namenjena čitanju in rezervaciji zapisov, če programski zapis tako določa. Če se želi čitati podrejene zapise od konca seta, potem se mora registru ZAČETNI KAZALEC prirediti vrednost nič (binarna ničla). Možno je tudi začeti branje (v obratni smeri) nekje na sredini seta, na ta način, da se registru ZAČETNI KAZALEC priredi vrednost kazalca zapisa. Ta način je možen samo takrat kadar shrani kazalec pri predhodni obdelavi istega seta.

Večinoma se register ZAČETNI KAZALEC uporablja za navodilo IDA Bazi, da mora začeti čitati od konca seta. Register ZAČETNI KAZALEC je namenjen samo vpisovanju vrednosti, zato je njegova vsebina za uporabnika nepomembna.

Primeri za uporabo funkcije: GETR

Tip zapisa	Komentar
1. NADREJENI	ni mogoče - izvede se GETG
2. PODREJENI	enako kot GETG
3. KOMB. S SETOM	enako kot GETG
4. KOMB. BREZ SETA	ni mogoče - izvede se GETG

3.3.3.6. INSA

Uporabi INSA za dodajanje podrejenega zapisa IZZA določene točke v setu, ki ga indentificira ključ seta.

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija INSA

zapis Ime programskega zapisa, ki naj se doda.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Ključ seta.

Akcija:

Vsebina parametra "V/Ipodr." se na podlagi programskega zapisa navedenega v parametru "zapis" zapiše v set, ki ga indentificira ključ seta naveden v parametru "ključ". Zapis se doda logično IZZA POZICIONIRANEGA(*) zapisa.

Preden se uporabi funkcijo INSA mora biti zapis izza katerega se želi novi zapis dodati rezerviran. Rezervacija se izvrši s katerokoli GET funkcijo istega programskega zapisa, ki mora seveda imeti pravilne dostopne pravice. Po uspešno izvedeni INSA funkciji je POZICIONIRAN(*) in rezerviran pravkar dodan zapis, kar pomeni, da se lahko doda naslednji zapis v isti set brez ponovnega čitanja.

Če je v shemi definiranih več nadrejenih zapisov za programski zapis, potem se zapis doda logično na konec vseh setov razen tistega, ki je naveden v programskem zapisu.

Če se funkcija INSA izvede uspešno, se registru TEKOČI KAZALEC priredi vrednost kazalca pravkar dodanega zapisa.

Primeri za uporabo funkcije: INSA v DODATKU D

Tip zapisa	Komentar
1. NADREJENI	ni mogoče - izvede se INSG
2. PODREJENI	enako kot INSB
3. KOMB. S SETOM	enako kot INSB
4. KOMB. BREZ SETA	ni mogoče - izvede se INSG

3.3.3.7. INSB

Uporabi INSB za dodajanje podrejenega zapisa PRED določeno točko v setu, ki ga indentificira ključ seta.

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija INSB

zapis Ime programskega zapisa, ki naj se doda.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Ključ seta.

Akcija:

Vsebina parametra "V/Ipodr." se na podlagi programskega zapisa navedenega v parametru "zapis" zapiše v set, ki ga indentificira ključ seta naveden v parametru "ključ". Zapis se doda logično PRED POZICIONIRANI(*) zapis.

Preden se uporabi INSB funkcija mora biti zapis pred katerega se želi novi zapis dodati rezerviran. Rezervacija se izvrši s katerokoli GET funkcijo istega programskega zapisa, ki mora seveda imeti pravilne dostopne pravice. Po uspešno izvedeni INSB funkciji je POZICIONIRAN(*) in rezerviran pravkar dodan zapis, kar pomeni, da se lahko doda naslednji zapis v isti set brez ponovnega čitanja.

Če je v shemi definiranih več nadrejenih zapisov za programski zapis, potem se zapis doda logično na konec vseh setov razen tistega, ki je naveden v programskem zapisu.

Če se funkcija INSB izvede uspešno, se registru TEKOČI KAZALEC priredi vrednost kazalca pravkar dodanega zapisa.

Primeri za uporabo funkcije: INSB v DODATKU D

Tip zapisa	Označbe v primeru	Komentar
1. NADREJENI		ni mogoče - izvede se INSG
2. PODREJENI		enako kot s setom
3. KOMB. S SETOM	(15)	
4. KOMB. BREZ SETA		ni mogoče - izvede se INSG

3.3.3.8. INSG

Uporabi INSG za:

- Dodajanje nadrejenega zapisa s ključem zapisa.
- Dodajanje podrejenega zapisa na konec vseh setov

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija INSG

zapis Ime programskega zapisa, ki naj se doda.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Nadrejeni zapis: Ključ zapisa.
Podrejeni zapis: Ključ seta.

Akcija:

Če parameter "zapis" označuje nadrejeni programski zapis, potem se vsebina parametra "V/Ipodr." s ključem zapisa, ki je naveden v parametru "ključ" in na podlagi programskega zapisa navedenega v parametru "zapis" doda.

Če parameter "zapis" označuje podrejeni programski zapis, potem se vsebina parametra "V/Ipodr." na podlagi programskega zapisa navedenega v parametru "zapis" doda na konec seta, ki ga indentificira ključ seta naveden v parametru "ključ". Če za podrejeni zapis obstoja več nadrejenih zapisov tako, da podrejeni zapis pripada več setom, se doda logično na konec vseh setov.

Vsi nadrejeni zapisi morajo obstojati preden se lahko doda podrejeni zapis.

Po uspešno izvedeni INSG funkciji je POZICIONIRAN(*) in rezerviran pravkar dodan zapis. Njegov kazalec se nahaja v registru TEKOČI KAZALEC.

Primeri za uporabo funkcije: INSG v DODATKU D

Tip zapisa	Označbe v primeru	Komentar
1. NADREJEN	(6),(9)	
2. PODREJEN	(18)	
3. KOMB. S SETOM	(16)	
4. KOMB. BREZ SETA		enako kot nadrejen zapis

3.3.3.9 RWRG

Uporabi RWRG za:

- Ažuriranje nadrejenega zapisa s ključem zapisa.
- Ažuriranje POZICIONIRANEGA(*) podrejenega zapisa.

Format: Funkcija DBMIO ukaza.

Parametri:

funkcija RWRG

zapis Ime programskega zapisa, ki naj se ažurira.

V/Ipodr. Področje definirano v programu za vsebino programskega zapisa.

ključ Nadrejeni zapis: Ključ zapisa.
Podrejeni zapis: Ključ seta.

Akcija:

Če parameter "zapis" označuje nadrejeni programski zapis, potem vsebina parametra "V/Ipodr." s ključem zapisa, ki je naveden v parametru "ključ" in na podlagi programskega zapisa navedenega v parametru "zapis" zamenja vsebino obtoječega zapisa.

Če parameter "zapis" označuje podrejeni programski zapis, potem vsebina parametra "V/Ipodr." na podlagi programskega zapisa navedenega v parametru "zapis", zamenja POZICIONIRANI(*) zapis, ki ga indentificira ključ seta naveden v parametru "ključ".

Po uspešno izvedeni funkciji RWRG je POZICIONIRAN(*) in rezerviran pravkar dodan zapis. Njegov kazalec se nahaja v registru TEKOČI KAZALEC.

Primeri za uporabo funkcije: RWRG v DODATKU D

Tip zapisa	Označbe v primeru	Komentar
1. NADREJENI	(32)	
2. PODREJENI	(29)	
3. KOMB. S SETOM		enako kot podrejeni
4. KOMB. BREZ SETA		enako kot nadrejeni

3.3.4. SEQIO FUNKCIJE

Naslednje funkcije so naštete v abecednem vrstem redu.

- SCLO
- SGET
- SINS
- SOPE
- SRWD
- SRWR

3.3.4.1 SCLO

Uporabi SCLO za zapiranje (CLOSE) sekvenčne datoteke.

Format: Funkcija SEQIO ukaza.

Parametri:

funkcija SCLO

zapis Ime sekvenčnega programskega zapisa, ki naj se zapre.

V/Ipodr. Se ne upošteva.

dolzina Se ne upošteva.

Akcija:

Sekvenčna datoteka, katere programski zapis je naveden v parametru "zapis" se zapre.

Če se funkcija uspešno izvede, potem se v register DB STATUS vrne ****. Vsaka druga vrednost pomeni napako. Vrednost "FPO2" pomeni, da je prišlo do napake v datotečnem sistemu. V tem primeru je napaka datotečnega sistema (normalno RMS) v registru SQ STATUS.

Sekvenčna datoteka za katero ni več potrebna nobena V/I funkcija naj se zapre.

3.3.4.2. SGET

Uporabi SGET za branje sekvenčne datoteke.

Format: Funkcija SEQIO ukaza.

Parametri:

funkcija SGET

zapis Ime sekvenčnega programskega zapisa, ki naj se čita.

V/Ipodr. Področje definirano v programu za zapis.

dolzina Dolzina prebranega zapisa (informacija).

Akcija:

Zapis iz sekvenčne datoteke, katere programski zapis je naveden v parametru "zapis" se prečita.

Če se funkcija uspešno izvede, potem se v register DB STATUS vrne ****. Vsaka druga vrednost pomeni napako. Vrednost "FPO2" pomeni, da je prišlo do napake v datotečnem sistemu. V tem primeru je napaka datotečnega sistema (normalno RMS) v registru SQ STATUS.

3.3.4.3. SINS

Uporabi SINS za dodajanje zapisov v sekvenčno datoteko.

Format: Funkcija SEQIO ukaza.

Parametri:

funkcija SINS

zapis Ime sekvenčnega programskega zapisa, ki naj se doda.

V/Ipodr. Področje definirano v programu za zapis.

dolžina Dolžina zapisa, ki naj se doda.

Akcija:

Zapis, katerega vsebina se nahaja v parametru "V/Ipodr.", v dolžini, ki jo podaja parameter "dolžina" je dodan v datoteko, ki jo navaja parameter "zapis".

Če se funkcija uspešno izvede, potem se v register DB STATUS vrne ****. Vsaka druga vrednost pomeni napako. Vrednost "FPO2" pomeni, da je prišlo do napake v datotečnem sistemu. V tem primeru je napaka datotečnega sistema (normalno RMS) v registru SQ STATUS.

3.3.4.4. SOPE

Uporabi SOPE za odpiranje (OPEN) sekvenčne datoteke.

Format: Funkcija SEQIO ukaza.

Parametri:

funkcija SOPE

zapis Ime sekvenčnega programskega zapisa, ki naj se odpre.

V/Ipodr. Se ne upošteva.

dolžina Se ne upošteva.

Akcija:

Sekvenčna datoteka, katere programski zapis je naveden v parametru "zapis" se odpre.

Če se funkcija uspešno izvede, potem se v register DB STATUS vrne ****. Vsaka druga vrednost pomeni napako. Vrednost "FPO2" pomeni, da je prišlo do napake v datotečnem sistemu. V tem primeru je napaka datotečnega sistema (normalno RMS) v registru SQ STATUS.

3.3.4.5. SRWD

Uporabi SRWD za postavljanje (rewind) sekvenčne datoteke na začetek.

Format: Funkcija SEQIO ukaza.

Parametri:

funkcija SRWD

zapis Ime sekvenčnega programskega zapisa, ki naj se postavi na začetek.

V/Ipodr. Se ne upošteva.

dolžina Se ne upošteva.

Akcija:

Sekvenčna datoteka, katere programski zapis je naveden v parametru "zapis" se postavi na prvi zapis.

Če se funkcija uspešno izvede, potem se v register DB STATUS vrne ****. Vsaka druga vrednost pomeni napako. Vrednost "FPO2" pomeni, da je prišlo do napake v datotečnem sistemu. V tem primeru je napaka datotečnega sistema (običajno RMS) v registru SQ STATUS.

3.3.4.6. SRWR

Uporabi SRWR za ažuriranje zapisov sekvenčne datoteke.

Format: Funkcija SEQIO ukaza.

Parametri:

funkcija SRWR

zapis Ime sekvenčnega programskega zapisa, ki naj se ažurira.

V/Ipodr. Področje definirano v programu za zapis.

dolžina Se ne upošteva.

Akcija:

Zapis, katerega vsebina se nahaja v parametru "V/Ipodr." in je bil predhodno prebran in spremenjen je ažuriran v sekvenčni datoteki, ki jo navaja parameter "zapis".

Če se funkcija uspešno izvede, potem se v register DB STATUS vrne ****. Karšnakoli druga vrednost pomeni napako. Vrednost "FP02" pomeni, da je prišlo do napake v datotečnem sistemu. V tem primeru je napaka datotečnega sistema (običajno RMS) v registru SQ STATUS.

4. PROGRAM ZA FORMATIRANJE BAZE PODATKOV (DBF)

4.1. UVOD

Program za formatiranje baze podatkov (DBF) se uporablja za formatiranje fizične strukture baze podatkov tako, da se optimalno izkoristi prostor na disku in zagotovi optimalno delo s podatki, zapisanimi v bazi (dobre performanse).

DBF program se uporablja, ko nastopijo naslednji trije slučaji:

1. Za kreiranje zbirk baze podatkov na disku preden se uporabi nova baza podatkov, definirana z DDC programom (primarno formatiranje). Primarno formatiranje se uporablja tudi takrat, ko se doda nova zbirka zapisov v obstoječo bazo podatkov.

2. Za reformatiranje ene ali več zbirk zapisov v obstoječi fizični strukturi potem, ko se spremeni definicija baze podatkov (npr. doda se nov podatkovni element, spremeni povezava med zapisi itd.). Vsi podatki v reformatirani zbirki zapisov so uničeni (izgubljeni). Da se podatki ohranijo, se pred reformatiranjem uporabi DBGet servisni program, za zapis sekvenčne datoteke. Pazi, da pripadajoči programski zapis vsebuje vse možne elemente podatkov.

3. Za razširitev PODREJENE(*) zbirke zapisov v obstoječi fizični strukturi, ko ta postane polna (status baze podatov DE07) ali, ko je napolnjena do 85% (status DE13). Status je vrnjen izvajalnemu programu, lahko pa se zazna tudi pri kontroli delovanja baze podatkov s pomočjo DBControl programa.

   OPOMBA:

   Razširitev je do nadaljnjega možna samo na podrejenih tipih zbirk zapisov.

Postopek za razširitev je v tem trenutku naslednji:

- Najprej se zaustavi vsa aktivnost na bazi (bazo se zaustavi z DBControl programom)
- Sledi razširitev (povečanje števila zapisov) fizične strukture z DDC programom
- Naredi dejansko razširitev s programom DBF
- Nadaljuje se z delom (postavitev baze z DBControl programom itd)

Torej za razširitev zbirke zapisov ni potreben ponoven vnos obstoječih podatkov. Za neodvisne in kombinirane vrste zapisov, ko bo to podprto, se priporoča razširitev do 20% od celotne velikosti, sicer se učinkovitost dela na bazi s temi zapisi poslabša.

4.2. IZVAJANJE PROGRAMA ZA FORMATIRANJE BAZE PODATKOV

Program se požene tako, da se izbere "3 DBF" iz IDA Baza menija, ali z ukazom:

DELTA/M: DBF<CR>

DELTA/V: $ DBF<CR>

Na zaslonu se prikaže DBF meni (glej sliko 4-1).

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA              31-DEC-87
Baza ) DBF

                              1. Primarno formatiranje
                              2. Reformatiranje
                              3. Razsiritev

                        Izberi: 2
```

Slika 4-1. DBF meni

Lahko izberemo med tremi funkcijami. Po izboru funkcije se prikaže naslednja zaslonska slika, ki je enaka za vse tri funkcije:

```
ISKRA DELTA                    IDA PROGRAMSKA OBOJA                31-DEC-87
Baza ) DBF ) Primarno formatiranje

Ime operativnega podrocja     : _______
Geslo operativnega podrocja   : ______
Lista tipov zapisov           : ____________________________
```

Slika 4-2. Zaslon pri funkciji: Primarno formatiranje

Vnesi ime in geslo operativnega področja, ki ga želiš formatirati.

Potem vnesi listo imen zapisov, ki jih želiš izvajati, ločenih z vejicami. Če želiš izvajati vse zapise definirane v shemi, vnesi "ALL.".

Takoj, ko so vnešena imena zapisov se nad njimi izvaja izbrana funkcija. Ko je izvajanje funkcije končano, se avtomatsko vrne DBF meni.

Delo z DBF programom se konča tako, da se vnese ukaz EXIT, ali preprosto <RETURN>.

4.3. PRIMER RAZŠIRITVE PODREJENEGA ZAPISA "NARIZD"

Primer je vzet iz testne baze podatkov PRODAJ, ki je opisana kot testni primer skozi cel priročnik in je tudi kot testni primer pripravljena na vašem računalniku.

Prvič: Potrebno je zaustaviti (če je aktivno) operativno področje PRODAJ1.

Drugič: Editira se DDC datoteka - v našem primeru: PRODAJ000.DDC. Na koncu opisa fizične strukture se doda želena razširitev (primer je na sliki 4-3). Razširitev lahko izvršimo v isti kontejner datoteki (kot primer 4-3), v drugi obstoječi ali v popolnoma novi (kot je opisano v poglavju 2).

```
PHYSICAL-STRUCTURE-DESCRIPTION
PHYSICAL-STRUCTURE name is PRODAJ
PASSWORD     is PRODAJ
               LOGICAL CONTAINER name is PRODAJ
               CONTAINER file name is LB:[6,65]PRODAJ.CON
               CONNECT record         DPSRLX
               OCCURENCY number    is 100
               BLOCK contains            1    records
               CONNECT record         DPSRLX
               OCCURENCY number    is 100
               BLOCK contains            1    records
               CONNECT record         DPSTLX
               OCCURENCY number    is 100
               BLOCK contains            1    records
               CONNECT record         KUPCII
               OCCURENCY number    is 100
               BLOCK contains            1    records
               CONNECT record         IZDLKI
               OCCURENCY number    is 100
               BLOCK contains            1    records
               CONNECT record         NAROCI
               OCCURENCY number    is 100
               BLOCK contains            1    records
               CONNECT record         NARIZD
               OCCURENCY number    is 100
               BLOCK contains            1    records
*-------------------------------------------------------------
* Primer razsiritve zbirke zapisa NARIZD v istem kontejnerju

* Zbirka je razsirjena v istem kontejnerju
* in to OBVEZNO NA KONCU za zadnjim zapisom v kontejnerju
               CONNECT record         NARIZD
               OCCURENCY number    is 50
*-------------------------------------------------------------
END-OF-DESCRIPTION
```

Slika 4-3. DDC datoteka: Razširitev fizične strukture

Tretjič: Izvede se sprememba opisa baze podatkov z DDC programom, v našem primeru z ukazom:

```
$ DDC PRODAJ000
```

ali

```
> DDC PRODAJ000
```

Četrtič: Sedaj je potrebno spremembo opisa (deskriptorja) baze podatkov tudi realizirati na disku. Za ta namen se uporabi DBF program in sicer funkcijo "3. Razširitev". Zaslon Razširitev je prikazan na sliki 4-4. Torej odgovori se na vprašanje: ime in geslo operativnega področja in, lista zapisov kot je prikazano. DBF program potem, ko razširja javlja sporočilo, kar se tudi vidi iz slike 4-4.

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                31-DEC-87
Baza ) DBF ) Razsiritev

Ime operativnega podrocja     : PRODAJ1
Geslo operativnega podrocja   : PRODAJ
Lista tipov zapisov           : NARIZD


Procesiram --- NARIZD ---   EXTN
```

Slika 4-4. DBF zaslon pri funkciji: Razširitev

S tem je proces razširjanja končan. Bazo podatkov se zopet aktivira in nadaljuje z delom.

Razširitev nadrejenega tipa zapisov (nadrejene in kombinirane zbirke zapisov) poteka podobno kot pri podrejenih zapisih, vendar je potrebno paziti na:

- Zbirko nadrejenega tipa je potrebno reorganizirati, tj. izpisati jo v sekvenčno datoteko še preden se razširja.
- Zaradi reorganizacije je z DBF potrebno to zbirko poleg razširitve tudi reformatirati. Najprej jo je treba razširiti, potem pa reformatirati. Obratno program DBF ne dovoli.
- Po uspešnem delu z DBF razširjeno zbirko zapisa ponovno napolnimo.

OPOZORILO: Pri razširjanju zbirk zapisov se ne sme editirati DDC datoteke, tako da se popravlja število zapisov (occurency number), temveč, da se povečano število doda na koncu opisa fizične strukture za želeno kontejner datoteko.

Število zapisov se lahko popravlja le kadar reorganiziramo celotno kontejner datoteko (DBF → primarno formatiranje).

Beležke:

5. PROGRAM ZA KONTROLO DELA BAZE PODATKOV (DBControl)

5.1. UVOD

Program za kontrolo dela baze podatkov se uporablja za upravljanje baze podatkov pred in med normalno uporabo. Uporabnik lahko začne z delom na bazi šele, ko se aktivira DBControl program.

5.2. IZVAJANJE PROGRAMA ZA KONTROLO DELA BAZE PODATKOV

Program se požene tako, da se izbere "2 DBControl" iz IDA Baza menija ali pa z ukazom:

DELTA/V: $ DBC !

DELTA/M: > DBC ;

Zatem se prikaže DBControl meni (glej sliko 5-1).

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA              31-DEC-87
Baza ) DBControl

                               1. Aktiviranje baze podatkov
                               2. Zaustavljanje baze podatkov
                               3. Kontrola dela baze podatkov

                        Izberi : 1
```

Slika 5-1. DBControl meni

Če želiš zapustiti meni, vnesi EXIT ali pritisni <RETURN>.

5.2.1. AKTIVIRANJE BAZE PODATKOV

Izberi "1. Aktiviranje baze podatkov" iz DBControl menija. Ta izbor povzroči prikaz menija Aktiviranje baze podatkov (glej sliko 5-2).

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                31-DEC-87
Baza ) DBControl ) Aktiviranje baze podatkov

Ime operativnega podrocja      : ______
Geslo operativnega podrocja    : ______

                               1. Brez logiranja
                               2. Logiranje funkcij
                               3. Logiranje transakcij
                               4. Logiranje funkcij in transakcij

                        Izberi : 1
```

Slika 5-2. Meni: Aktiviranje baze podatkov

Ko se prikaže ta zaslon, vnesi ime in geslo operativnega področja, ki ga želiš aktivirati.

Dalje izberi tip logiranja. Lahko logiraš samo funkcije ali transakcije, ali pa oboje hkrati (glej poglavje 6: Beleženje sprememb in obnova baze podatkov).

Ko si izbral tip logiranja, počakaš približno 5 sekund, da se baza postavi (dobiš sporočilo na zaslon "IDA Baza je AKTIVNA"), in se avtomatsko vrneš na DBControl meni. V kolikor je pred tem prišlo do nenormalne prekinitve (npr. zaradi izpada električnega toka), se pred samim startanjem avtomatsko izvede obnova baze podatkov (WARM RESTART).

5.2.2. ZAUSTAVLJANJE BAZE PODATKOV

Izberi "2. Zaustavljanje baze podatkov" iz DBControl menija (glej sliko 5-1). Izbor povzroči prikaz menija Zaustavljanje baze podatkov (glej sliko 5-3).

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                31-DEC-87
Baza ) DBControl ) Zaustavljanje baze podatkov

Ime operativnega podrocja      : ______
Geslo operativnega podrocja    : ______
```

Slika 5-3. Meni: Zaustavljanje baze podatkov

Vnesi ime in geslo operativnega področja, ki ga želiš zaustaviti. Če na bazi ni aktivnih uporabnikov, se baza zaustavi brez vprašanj za ime in geslo operativnega področja. Sicer javi število še aktivnih programov in zahteva vnos, kot je navedeno na sliki 5-3. Ko je proces zaustavljanja končan, se avtomatsko vrneš na DBControl meni.

5.2.3. KONTROLA DELOVANJA BAZE PODATKOV

Izberi "3. Kontrola dela baze podatkov" iz DBControl menija (glej sliko 5-1). Izbor povzroči prikaz zaslona Kontrola delovanja baze podatkov, kot je prikazan na sliki 5-4.

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                    31-DEC-87
Baza ) DBControl ) Kontrola dela baze podatkov

Ime sheme                :____ St. vhodov v rezerv. listi :___
St. aktivnih programov   :____ St. READONLY procesov      :___
St. DB klicev            :____ St. UPDATE procesov        :___
St. DB klicev/sek        :____ Tip logiranja              :___

zapis     STATUS                zapis     STATUS
1 -                             13-
2 -                             14-
3 -                             15-
4 -                             16-
5 -                             17-
6 -                             18-
7 -                             19-
8 -                             20-
9 -                             21-
10-                             22-
11-                             23-
12-                             24-
```

Slika 5-4. Meni: Kontrola delovanja baze podatkov

Zaslon kontrole delovanja baze podatkov dinamično prikazuje delovanje baze podatkov za shemo, ki je navedena v zgornjem delu zaslona. Slika zaslona se obnovi z novimi podatki cca vsake 3 sekunde.

Pomen okrajšav za tip logiranja:

- NI nobeno logiranje
- FUNK. logiranje funkcij
- TRANS. logiranje transakcij
- FU&TR logiranje funkcij in transakcij

Zbirke zapisov imajo lahko naslednje statuse:

Status	Pomen	Akcija
****	Normalno	Nobena
Poln	Zapis je poln	Potrebna je takojsnja razsiritev
Zaklenjen	Zapis je zaklenjen	Bazo je potrebno obnoviti
Napolnjen 85%	Zapis je napolnjen do 85%	Potrebna je razsiritev
Ni odprt	Zapis se ni uporabljen	Nobena

Za prekinitev prikazovanja baze podatkov vtipkaj E<CR>.

6. BELEŽENJE SPREMEMB IN OBNOVA BAZE PODATKOV

6.1. UVOD

IDA Baza vsebuje vse potrebne varovalne mehanizme, da zaščiti podatke, kadar se zgodi ena od naslednjih napak:

- Napaka logične transakcije
- Programska napaka
- Izpad celotnega sistema
- Napaka na diskovni enoti

Napaka logične transakcije: Programska enota (npr. transakcijski program) iz kateregakoli razloga (nepravilno vnešeni podatki, pričakovani podatki ne obstajajo itd.) ne more izvesti logične transakcije do konca. V tem primeru se morajo razveljaviti vse spremembe, ki jih je povzročila nepopolna logična transakcija.

Programska napaka: Napaka v programski logiki (npr. deljenje z 0, neveljavna instrukcija itd.) lahko povzroči nenormalni zaključek programa. V tem primeru ostane lahko logična transakcija nekončana in mora biti zaradi tega razveljavljena.

Izpad sistema (zaradi napake v operacijskem sistemu, napake v elektroniki ali zaradi izpada električnega toka) povzroči nekotrolirano stanje na sistemu. Vsebina glavnega pomnilnika je lahko izgubljena ali pokvarjena. Lahko se zgodi, da ostane večje število logičnih transakcij nekončanih. Te morajo biti pri ponovni vzpostavitvi sistema razveljavljene.

Napaka na diskovni enoti: Tu so mišljene predvsem tiste napake, ki povzročijo izgubo podatkov (npr. head crash) na disku. Vsebina fizičih datotek je izgubljena. Obstajati mora način, da se ti izgubljeni podatki obnovijo.

6.2. BELEŽENJE IN OBNOVA LOGIČNIH TRANSAKCIJ

Ta mehanizem je oblikovan in realiziran zato, da varuje integriteto podatkov v treh primerih napak:

- Napaka logične transakcije
- Programska napaka
- Izpad celotnega sistema

V vseh treh primerih je "logična transakcija" edina obnovljiva enota z vidika IDA Baze, ki formalno definira logično transakcijo kot zaporedje DML ukazov med dvema omejevalcema logične transakcije. Ti omejevalci so:

- HELLO
- BYE
- CANCEL
- COMMIT

Samo COMMIT pomeni potrditev logične transakcije, ki povzroči, da se izvedene spremembe (dodajanje novih zapisov in spreminjanje ali brisanje obstoječih zapisov) fizično realizirajo v bazi podatkov. Ostali ukazi povzročijo, da se logična transakcija razveljavi, kar pomeni odstranitev vseh nepotrjenih sprememb iz baze podatkov.

Kadar je neka logična transakcija razveljavljena (bodisi z uporabo ukaza CANCEL ali da jo je razveljavila IDA Baza), nima to nobene posledice za ostale logične transakcije, ki se izvajajo v okviru konteksta drugih programov.

V primeru programske napake, ki povzroči nenormalno prekinitev izvajanja tega programa, IDA Baza avtomatsko izvede ukaz BYE za ta program, nepopolna logična transakcija se razveljavi in vse njene posledice so s tem odstranjene. To je razlog, da z vidika IDA Baze pomeni BYE razveljavitev (enako kot CANCEL) logične transakcije.

Ko se v primeru izpada sistema le-ta ponovno vzpostavi, IDA Baza avtomatsko izvede obnovo v času aktiviranja IDA Baze s programom DBC. Za obnovo je potrebno od 1 do 10 sekund. V bazi podatkov se realizirajo samo potrjene logične transakcije, nepotrjene pa se razveljavijo in nimajo nobenih posledic.

6.3. BELEŽENJE FUNKCIJ IN OBNOVA BAZE PODATKOV

Ta mehanizem varuje podatke v primeru, kadar so fizično uničeni podatki na diskih. To se lahko zgodi v primeru head crash-a ali pa tudi v primeru neprevidne uporabe sistemskih ali drugih ukazov, ki imajo za posledico brisanje dela ali cele baze podatkov.

Beleženje funkcij pomeni beleženje vseh DML ukazov in funkcij, ki spreminjajo bazo podatkov, v datoteko logiranih funkcij, ki se

mora praviloma nahajati na drugi diskovni enoti kot baza podatkov.

Obnovitev na meji logičnih transakcij je možna samo v primeru, če je bilo aktivirano hkrati z beleženjem funkcij tudi beleženje transakcij.

Nova datoteka logiranih funkcij se mora inicializirati vedno in samo takrat, kadar se naredijo varnostne kopije fizičnih datotek baze podatkov. Pri tem velja, da se smatra reformatiranje vseh datotek enako kot varnostno kopiranje. Delno reformatiranje zahteva varnostno kopiranje in inicializacijo datoteke logiranih funkcij.

V določenih primerih lahko koristi paralelno varnostno kopiranje DB fizičnih datotek in datoteke logiranih funkcij. V primeru napake na mediju varnostnih kopij se lahko uporabi starejše varnostne kopije fizičnih datotek in vse datoteke logiranih funkcij od istega datuma naprej. Pri tem je važen pravilen vrstni red datotek logiranih funkcij za obnavljanje.

V primeru izpada diskovne datoteke se da obnoviti bazo podatkov na naslednji način:

1. Kopiranje fizičnih datotek z varnostnih kopij, kar zagotavlja vzpostavitev zadnje konsistentne baze podatkov. Za to operacijo se uporabljajo standarni sistemski programi (npr. Backup ali Bru).

2. Uporaba obnovitvenega programa (DBRES).

3. Zaradi nadaljnje uporabe beleženja funkcij, je po uspešni obnovitvi baze potrebno napraviti nove varnostne kopije fizičnih datotek in izbrisati staro arhivsko datoteko .

4. Nadaljuje se normalno z aktiviranjem baze in z rednim delom.

6.4. UPORABA DBRESTORE PROGRAMA

DBREStore program se aktivira z uporabo ukaza DBRES(tore), ki izpiše ekran, kot ga prikazuje slika 6-1.

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                31-DEC-87
Baza ) DBREStore

Ime sheme          :______
Geslo              :______
Datum (dd/mm/ll)   :__/__/__
Cas (hh:mm:ss)     :__:__/__
```

Slika 6-1. DBREStore meni

Najprej je treba vnesti ime sheme in geslo sheme.

Potem je treba vnesti datum in čas, do katerega se želi obnoviti bazo podatkov. Default odgovor (<CR>) pomeni, da bo baza podatkov obnovljena z vsemi funkcijami iz datoteke logiranih funkcij.

Če je vnešeni časovni parameter (datum in čas) večji od časa izpada diska, se bo pojavilo poročilo:

"Ugotovljen je izpad IDA Baze"

Čas obnovitve se lahko pomakne na določeno časovno točko, na primer zaradi tega, da se ponovno izvaja paketna obdelava, ki je bila prekinjena zaradi izpada sistema. Vsi DB ukazi in funkcije, ki so se zgodile po tem času, ne bodo upoštevane pri obnovitvi.

7. POMOŽNI PROGRAMI BAZE PODATKOV

7.1. UVOD

Pomožni programi baze podatkov (DBGet, DBPut, DBDel) omogočajo:

- Primarno polnjenje zbirke zapisov
- Dodajanje novih zapisov v že obstoječe zbirke zapisov
- Prepis zbirke zapisov v sekvenčno datoteko
- Brisanje zapisov ali seta zapisov iz zbirke zapisov
- Reorganizacijo baze podatkov

Pri uporabi pomožnih programov se morajo upoštevati naslednja pravila:

1. Vsi pomožni programi se izvajajo nad programskimi zapisi. Izbrani programski zapis mora imeti pravilne pristopne pravice za izvajanje operacij (INSG za DBGet pomožni program, GETP za DBGet pomožni program in GETG DELG za DBDel pomožni program).
2. Prvi podatkovni element v programskem zapisu mora biti ključ zapisa (ključ za neodvisne zapise ali ključ povezave za odvisne zapise). Izjema so označeni zapisi, kjer mora biti prvi podatek oznaka zapisa.
3. DBPut pomožni program zahteva, da za vsako podatkovno polje izbranega programskega zapisa obstaja polje v vhodni datoteki. Zaradi tega morajo podatkovna polja definirana v programskem zapisu odgovarjati (po zaporedju, dolžini in tipu) strukturi vhodne datoteke.
4. Pomožni programi kreirajo datoteke "poročilo" o izvajanju. V datoteki se nahajajo vhodni parametri (ime podsheme, ime programskega zapisa, ime sekvenčne datoteke), napake do

katerih je prišlo med izvajanjem pomožnih programov in statistika izvedenih operacij. Datoteka se kreira na direktoriju z logičnim imenom DBV_SHEMA na DELTA/V oziroma na SY:S1,620 na DELTA/M sistemu..

5. Pri vseh pomožnih programih je prevzeta specifikacia sekvenčnih datotek specifikacija terminala ("TI:" za DELTA/M; "SYS$OUTPUT" za DELTA/V).

7.2. DBGet - PREPIS ZAPISOV BAZE PODATKOV V SEKVENČNO DATOTEKO

Pomožni program DBGet se uporablja za prepis zbirk zapisov baze podatkov v sekvenčne datoteke. Zapisi baze podakov se berejo v zaporedju po katerem so fizično zapisani v bazi podatkov.

Opozorilo: Programski zapis podsheme mora imeti GETP pristopno pravico.

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                31-DEC-87
Baza ) Pomozni programi ) DBGet

Ime podsheme                          :
Geslo podsheme                        :
Ime zapisa podsheme                   :
Specifikacija sekvencne datoteke      :
Velikost zapisa sekvencne
datoteke ((RETURN) = max)             :
Stevilo zapisov
((RETURN) = VSE)                      :
```

Slika 7-1. Zaslon DBGet programa

Vnese se ime podsheme, ki vsebuje programski zapis za prepis v sekvenčno datoteko, geslo podsheme, ime programskega zapisa za prepis v sekvenčno datoteko in dovoljeno ime za sekvenčno datoteko.

Potem se vnese dolžina sekvenčnega zapisa, ki ga piše pomožni program. Lahko se vnese številka do 1024. Če se vnese <RETURN>, bo sekvenčni zapis enake dolžine, kot je dolžina podatkovnih elementov definiranih v programskem zapisu.

Vnese se število zapisov za prepis v sekvenčno datoteko. Za prepis celotne zbirke zapisov zadostuje <RETURN>.

Po vnosu vseh podatkov se prepisujejo podatki v sekvenčno datoteko.

7.3. DBPut - POLNJENJE ZBIRKE ZAPISOV

Pomožni program DBPut se uporablja za prepis sekvenčnih datotek v zbirke zapisov.Zapisi sekvenčne datoteke se berejo in dodajajo v zbirke zapisov navedenega zapisa. Veljavni podatki v zapisu sekvenčne datoteke se začnejo na 1. poziciji. Take so npr. datoteke, ki jih kreira DBGet pomožni program.

V primeru dovoljenih napak se te napake in vsebina zapisa zapišejo v datoteko "poročilo", izvajanje DBPut programa pa se nadaljuje. Če pa pride do nedovoljene napake, se izvajanje pomožnega programa prekine, podatki o napaki in vsebina zapisa pa se zapiše v datoteko "poročilo".

Priporoča se uporaba SORT opcije za prepis v zbirke nadrejenih zapisov zaradi izboljšanja performans.

Opozorilo: Za uporabo DBPut programa mora imeti programski zapis INSG pristopno pravico.

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA              31-DEC-87
Baza ) Pomozni programi ) DBPut

Ime podsheme                 :
Geslo podsheme               :
Ime zapisa podsheme          :
Sortiran vnos (samo za zapise
tipa OWNER)          (D/N) :
Ali so zapisi oznaceni (pri
zapisih tipa MEMBER) (D/N) :
Specifikacija sek. datoteke  :
Dovoljeno stevilo napak      :
```

Slika 7-2. Zaslon DBPut programa

Vnese se ime podsheme, ki vsebuje programski zapis za prepis iz sekvenčne datoteke, geslo podsheme in ime programskega zapisa za zapis, ki se bo polnil.

Zapisi se pred zapisovanjem v bazo podakov lahko sortirajo. To velja samo za nadrejene zapise. Sortiranje izboljšuje performanse. Kadar je sortiranje zaželeno, se vnese "D" kot odgovor na ustrezno vprašanje. V nasprotnem primeru se vnese "N" ali samo <RETURN>.

V primeru, da se prepisujejo označeni zapisi (REQUAL), se na naslednje vprašanje odgovori z "D". V nasprotnem primeru se vnese "N" ali samo <RETURN>.

Naslednje vprašanje je ime sekvenčne datoteke, ki vsebuje podatke za prepis.

Zadnje vprašanje se nanaša na dovoljeno število napak. To je lahko vsako število, vendar, če je vnešena negativna vrednost ali samo <RETURN> je število dovoljenih napak 0. V slučaju, da se

med izvajanjem programa preseže dovoljeno število napak, se izvajanje programa prekine, če pa število dovoljenih napak še ni preseženo, se napaka in zapis zapišeta v datoteko sporočil (napake, ki ne prekinejo izvajanje programa, so "DI01", "DI02", "DI11" in "DI12").

Po vnosu vseh podatkov se prepisujejo podatki iz sekvenčne datoteke v zbirke zapisov.

Priporočila za uporabo pomožnega programa DBPut:

- DBPut vključuje (OPCIJA) sort, za katerega je treba upoštevati WORKING SET in prostor za delovne datoteke, ki so na nenaglašenem disku.
- Pri dodajanju zapisov v neodvisne zbirke (tipi O in S), posebno pri inicialni polnitvi se priporoča izbira opcije SORT. S tem se dosežejo pomembno boljše performanse.
- Pri dodajanju zapisov v odvisne zbirke se hitrost polnitve bistveno izboljša, če se vhodno sekvencialno datoteko predhodno sortira po naslenjem pravilu:

  Prvi sortirni pojem mora biti ključ primarne povezave (po kateri je definiran SET za programski zapis, ki se dodaja).

  Ostali sortirni pojmi so ključi za ostale povezave in morajo biti definirani v takem vrstnem redu, kot je njihova povprečna dolžina verig. Ključ z daljšimi verigami je višji sortirni pojem od ključa s krajšimi verigami.

  Kadar je povprečna dolžina verig neznana ali izenačena, je vrstni red sortirnih pojmov poljuben.

  Način sortiranja je naraščajoče (ascending).
- Pri masovnem dodajanju je priporočljivo definirati večje število V/I področij in sicer:
  - Pri dodajanju neodvisnih zapisov 32 V/I področij za zbirko, v katero se dodajajo zapisi.
  - Pri dodajanju odvisnih zapisov 4 do 6 V/I področij za zbirko odvisnih zapisov in po 32 V/I področij za vsako nadrejeno.

7.4. DBDel - BRISANJE ZAPISOV BAZE PODATKOV

Pomožni program DBDel se uporablja za brisanje zapisov baze podatkov. Ključi za brisanje se nahajajo v sekvenčni datoteki. DBDel se lahko uporablja za:

- Brisanje neodvisnih zapisov na osnovi vrednosti ključa.
- Brisanje verig zapisov na osnovi vrednosti ključa povezave.
- Brisanje neodvisnih zapisov in pripadajočih verig na osnovi vrednosti ključa.

OPOZORILO: Programski zapis podsheme mora imeti GETG in DELG pristopno pravico.

```
ISKRA DELTA                    IDA PROGRAMSKA ORODJA                31-DEC-87
Baza ) Pomozni programi ) DBDel

Ime podsheme                  :
Geslo podsheme                :
Nacin (1,2,3)                 :
Ime zapisa podsheme           :
Nadrejeni programski zapis    :
Specifikacija sek. datoteke   :
```

Slika 7-3. Zaslon DBDel programa

Vnese se ime podsheme, ki vsebuje programski zapis za prepis sekvenčne datoteke in geslo podsheme.

Vnese se številka, ki določi način izvajanja programa:

1. Brisanje zapisov na osnovi ključa
2. Brisanje verig na osnovi ključa povezave
3. Brisanje zapisov in verig na osnovi ključa (povezave)

Potem se vnese ime programskega zapisa za brisanje.

V slučaju, da je izbran način dela "3", je potrebno še ime nadrejenega programskega zapisa za brisanje.

Po vnosu vseh potrebnih podatkov se brišejo podatki iz zbirk zapisov.

V primeru napak "DI05", "DI09" in "DI15" se te zapišejo z vsebino ključa, ki jih je povzročil, v datoteko poročil.

7.5. REORGANIZACIJA BAZE PODATKOV

Občasno je potrebno zbirke zapisov reorganizirati zaradi:

- Spreminjanja dolžine zapisa zaradi dodajanja novih elementov.

- Spreminjanja logične strukture z dodajanjem novih povezav.

  Zbirka zapisov neodvisnih zapisov je zapolnila več kot 90% definiranega prostora.

Reorganizacija baze se izvaja po zbirkah zapisov. Potrebno je prepisati zbirke zapisov v sekvenčne datoteke, spremeniti shemo baze in/ali fizično strukturo (dodajanje novih elementov ali povezav) in ponovno napolniti zbirke zapisov.

OPOZORILO: Med izvajanjem reorganizacije ni dovoljeno izvajanje drugih operacij nad bazo podatkov.

Postopek reorganizacije:

1. Prepis obstoječih zbirk zapisov v sekvenčne datoteke z uporabo DBGet pomožnega programa.

2. Z uporabo DDC se izvedejo potrebne spremembe (spremebme zapisov, povezav, fizične strukture in podobno).

3. Reformatiranje (in po potrebi razširitev) obstoječe fizične strukture se izvede kot je to opisano v poglavju 2 in poglavju 4.

4. Polnjenje zbirk zapisov z vsebino sekvenčnih datotek, nastalih pri koraku 1, z uporabo pomožnega programa DBPut.

8. OPTIMIZIRANJE IDA BAZE

8.1. UVOD

IDA Baza je oblikovana tako, da se lahko instalira na vse računalnike DELTA, od najmanjših do največjih. Razumljivo je, da so cilji optimizacij na velikih sitemih popolnoma drugačni kot na malih računalnikih. V prvem primeru se skuša doseči čim krajši odzivni čas, saj mora IDA Baza podpirati veliko število istočasno aktivnih programov. Na malih sistemih pa se želi čimbolj zmanjšati pomnilniški prostor, ki ga uporablja IDA Baza. Ti dve zahtevi sta protislovni.

8.2. OBLIKOVANJE V/I PODROČIJ

Na odzivni čas najbolj vpliva število posegov na diskovne enote. Te se da zmanjšati s pravilno uporabo V/I področij.

Pri nadrejenih zapisih je uporaba V/I področij razmeroma enostavna. Najslabši odzivni čas je v primeru, da nadrejeni zapis deli V/I področje z vsemi drugimi zapisi. Zato je ta možnost prepovedana v DDC. Izboljšuje pa se v naslednjih primerih:

- če zbirka zapisov deli V/I področje samo z nekaterimi drugimi nadrejenimi in odvisnimi zbirkami
- če zbirka zapisov deli V/I področje samo z nekaterimi zbirkami istega tipa (npr. nadrejene zbirke samo z nadrejenimi)
- če ima zbirka zapisov svoje lastno V/I področje, ki ga ne deli z drugimi zbirkami
- če ima zbirka zapisov toliko V/I področij, kolikor je istočasno aktivnih programov, ki delajo V/I operacije s to zbirko
- če se število V/I področij iz prejšnje alineje poveča za 2.

Pri podrejenih in kombiniranih zapisih je treba upoštevati dejstvo, da so ti zapisi povezani s svojimi nadrejenimi zapisi. Praktično to pomeni, da je treba obravnavati pri optimizaciji V/I področij tudi vse nadrejene zapise, s katerimi je določen zapis povezan. Posamezne funkcije DML imajo različne zahteve glede V/I področij za zbirke podrejenih zapisov:

- GETP in GETD funkciji sta ovisni samo od V/I področij zbirke podrejenih zapisov
- GETC in GETR funkciji sta odvisni od V/I področij zbirk podrejenih zapisov in od V/I področij zbirke nadrejenih zapisov glede na izbrani set
- INSG, INSA, INSB in DELG funkcije so odvisne od V/I področij zbirk podrejenih zapisov in od V/I področij vseh zbirk nadrejenih zapisov, s katerimi je v povezavi

Treba je zagotoviti več V/I področij za tiste zbirke zapisov, na katerih se pogosteje izvajajo V/I operacije.

8.3. OBLIKOVANJE OPERATIVNIH PODROČIJ

Pri oblikovanju posameznih operativnih področij je treba analizirati funkcije programov, ki so istočasno aktivni. Zato se praviloma oblikuje za isto bazo podatkov (shemo) več operativnih področij. Tipična operativna področja, ki se pojavljajo skoraj pri vsaki bazi podatkov so:

- Operativno področje za primarno polnjenje in reorganizacije
- Operativno področje za interaktivni režim dela
- Operativno področje za paketni režim dela

Posebno na malih sistemih se priporoča večje število operativnih področij, saj gre navadno manjše število istočasno aktivnih programov. Zato je priporočljivo analizirati tudi dinamiko aktivnosti. V situacijah, ko se da definirati ponovljive skupine simultanih programov, se področje optimizira glede na te programe. Kadar se takih skupin ne da predvideti, je koristno, če se za več zapisov (ki naj ne bi bili med seboj povezani in imajo enako dolžino logičnega zapisa) definira skupno V/I področje, ki se večkrat ponovi.

8.4. OBLIKOVANJE FIZIČNE STRUKTURE

Cilj optimizacije fizične strukture je zmanjševanje mehanskih premikov na diskovnih enotah. Z vidika IDA Baze bi bilo najboljše, če bi bila vsaka zbirka enakih zapisov na svoji diskovni enoti. Ker to praktično ni mogoče, je treba upoštevati naslednja pravila za razvrščanje zbirk po diskovnih enotah:

- Minimalna zahteva je, da obstajata vsaj dva kontejnerja, od katerih eden vsebuje zbirke nadrejenih, drugi pa zbirke podrejenih zapisov. Vsak od navedenih kontejnerjev naj bo na svoji diskovni enoti.

- V kolikor je na voljo več diskovnih enot, je treba razporediti zbirke in kontejnerje tako, da so na ločenih diskih tiste zbirke, h katerim se pristopa istočasno.

  Pri tem je treba upoštevati, da pomenijo nekatere funkcije (GETG, GETR, INSG, INSA, INSB in DELG za podrejene zapis) posege v zbirke nadrejenih zapisov in posege v zbirko podrejenih zapisov.

- Za določene kratke zbirke, h katerim se pristopa zelo pogosto, se lahko definira toliko V/I področij, da so vsi zapisi v glavnem pomnilniku. V tem primeru te zbirke ni več treba upoštevati pri optimizacijah fizične strukture.

8.5. OBLIKOVANJE LOGIČNIH BLOKOV

Dolžina logičnih blokov tudi vpliva na velikost potrebnega glavnega pomnilnika za V/I področja, poleg tega pa tudi na izkoristek diskovnega prostora. Ker zapisi niso deljeni med logičnimi bloki, je treba upoštevati dejstvo, da je del bloka neizkoriščen.

- Dolžina neizkoriščenega dela bloka se lahko izračuna:

  L = dolžina bloka - dolžina zapisa * število zapisov v bloku

- Dolžina zapisa za nadrejene zapise:

  L = dolžina podatkov + 8 + število setov 8

- Dolžina zapisa za odvisne zapise:

  L = dolžina podatkov + število setov 8

Dolžina podatkovnega dela kombiniranega zapisa je enaka kot pri odvisnem zapisu, področje indeksov pa je pod kontrolo IDA Baze in se nanj ne more vplivati.

Najbolje je, da se dolžina bloka oblikuje tako, da v bloku ni neizkoriščenega prostora. IDA Baza je optimizirana za zapise dolge do 256 znakov, zato je priporočljivo, da zapisi niso daljši, če to ni nujno potrebno.

Zaradi internega razreševanja sinonimov je zelo nepriporočljivo oblikovanje logičnih blokov tako, da je v enem bloku en sam zapis. To onemogoča, da bi bil sinonim lahko razrešen v svojem bloku, kar zahteva dodatne V/I operacije na disku. To se vidi iz kratkega opisa tega algoritma.

Procesor sinonimov deluje pri dodajanju novih nadrejenih zapisov po naslednji shemi:

1. Ugotavlja ali je zapis, ki trenutno zaseda naslovno celico tja razporejen zaradi vrednosti ključa (pravi sinonim) ali pa ga je tja razporedil algoritem za iskanje prazne celice (navidezni sinonim).

   Tu se razreševanje razveja v dve smeri; v eni se razrešujejo pravi sinonimi, v drugi pa navidezni sinonimi.

   Razreševanje pravih sinonimov:

2. V kolikor za naslovno celico že obstaja veriga sinonimov, se naslovi in prebere zadnji zapis v verigi, sicer se predpostavi, da je edini zapis v verigi tudi zadnji.

3. Poišče se nezasedena celica. Za to iskanje je značilno, da se najprej skuša najti nezasedeno celico v bloku, kjer je zadnji zapis. Če take celice ni, se skuša najti nezasedeno celico v sekciji, kjer se nahaja zadnji zapis. Če tudi tu ni razpoložljive prazne celice, se išče prazno celico v sekcijah, ki slede.

4. V prazno celico se vpiše novi zapis.

5. Ker so sinonimi medsebojno obojestransko povezani s kazalci (veriga naprej in veriga nazaj), je potrebno hkrati popraviti kazalce verige sinonimov.

   Razreševanje navideznih sinonimov:

6. Poišče se nezasedeno celico in se vanjo premesti zapis, ki je označen kot navidezni sinonim. Pri tem se popravi tudi kazalce v verigi sinonimov, ki ji ta zapis pripada. S tem se sprosti celica, ki jo je prej zasedal navidezni sinonim.

7. V sproščeno celico se vpiše novi zapis, ki v tem primeru ne more imeti sinonimov.

8.6. OBLIKOVANJE VELIKOSTI ZBIRK NADREJENIH ZAPISOV

Ker se za dostop do nadrejenih (lahko tudi do kombiniranih) zapisov uporablja pršilni (hash) algoritem, je zelo ugodno, da je število razpoložljivih celic vsaj za petnajst odstotkov večje od dejanskega števila zapisov. S tem se pomembno izboljša pristopni čas do vseh zapisov v bazi podatkov. Ko število zapisov preseže to mejo, je treba zbirko povečati in reorganizirati.

8.7. OPTIMIZACIJA LOGIČNIH TRANSAKCIJ

Da bi se zmanjšalo medprogramsko čakanje na sprostitev zapisov, je koristno upoštevati naslednje navodilo za programiranje logičnih transakcij.

1. Najprej se izvaja branje (vse vrste GET) in spreminjanje zapisov (RWRG).

2. Potem se izvaja dodajanje(INSG) in brisanje (DELG) nadrejenih programskih zapisov.

3. Nazadnje se izvaja dodajanje (vse vrste INS) in brisanje (DELG) podrejenih programskih zapisov.

4. Čim hitreje naj sledi CONFRM ali CANCEL.

Naslednji tekst pojasnjuje razlog za tako oblikovanje logičnih transakcij.

Branje in spreminjanje zaklepa kvečjemu po en zapis zbirke.

Dodajanje in brisanje nadrejenih zapisov zbirke ponavadi zaklepa samo prizadeti zapis.

Dodajanje in brisanje podrejenih zapisov praviloma zaklepa več zapisov (praviloma najmanj štiri). Poleg tega pa ima še eno neugodno lastnost: zaradi povečevanja operativne hitrosti je učinkovitost diskovnih operacij na zbirkah podrejenih in kombiniranih zapisov povečana tako, da so celice združene v kontrolne sekcije. V okviru kontrolne sekcije se vodi tudi seznam razpoložljivih celic. Ker dodajanje oziroma brisanje zapisov spreminja ta seznam, mora biti tudi ta zaklenjen. To povzroči, da postane celotna kontrolna sekcija zaklenjena za dodajanje oziroma brisanje. Drugi programi lahko samo berejo in spreminjajo zapise zaklenjene kontrolne sekcije. Šele operacija CONFRM (ali CANCEL) ponovno odklene seznam razpoložljivih celic. Potem lahko tudi drugi programi izvajajo dodajanje in brisanje na tej kontrolni sekciji.

Beležke:

A. KAJ JE REFORMAT IN KAJ JE RAZŠIRITEV . . .

A.1. Reorganizacija zbirk zapisov Ida Baze s servisnimi programi DDC, DBF, DBGET in DBPUT

Tabela A-1: Vrste reorganizacij na IDA Bazi

Tip zapisa	Spremembe v fizični strukturi	Dolžina zapisa NI spremenjena	Dolžina zapisa JE spremenjena*1
PODREJENI ZAPISI	Velikost zbirke NI spremenjena*2	REFORMAT*4	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
	Velikost zbirke JE spremenjena*3	RAZŠIRITEV*6	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
NADREJENI ZAPISI	Velikost zbirke NI spremenjena*2	REFORMAT*7	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
	Velikost zbirke JE spremenjena*3	RAZŠIRITEV*8	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
KOMBINIRANI ZAPISI	Velikost zbirke NI spremenjena*2	REFORMAT*9	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
	Velikost zbirke JE spremenjena*3	RAZŠIRITEV*10	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5

*1) Dolžina zapisa se spremeni zaradi dodane ali odvzete povezave (linka) v logični strukturi, in dodanih ali odvzetih elementov v zapisu v strukturi sheme.

*2) Velikost zbirke zapisa je število zapisov te zbirke zapisov opisane v Fizični strukturi sheme. To število se ne spremeni.

*3) Število zapisov določene zbirke zapisov opisane v Fizični strukturi se spremeni.

*4) REFORMAT PODREJENEGA ZAPISA

- DBGET — Prepis zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.
  OPOZORILO: Logični vrstni red podrejenih zapisov v verigah se lahko spremeni !!!!
- DBF — REFORMAT zbirke zapisa v kontejnerju.
- DBPUT — Prepis iz sekvenčne datoteke v zbirko zapisa v kontejnerju.

*5) REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU

- DBGET — Prepis vseh zbirk zapisov iz kontejner datoteke v sekvenčne datoteke.
- DBGET — Če so v kontejnerju razen podrejenih zapisov še nadrejeni ali kombinirani, moramo prepisati v sekvenčne datoteke še njihove podrejene zbirke zapisov kjerkoli so.
- DDC — Sprememba opisa sheme ali logične strukture ali fizične strukture in izdelava novih deskriptorjev za zbirke v tem kontejnerju. Glej primer A.
- DELETE — Brisanje kontejnerja.
- DBF — Primarno formatiranje zapisov v tem kontejnerju.
- DBF — Reformatiranje tistih zbirk, ki so podrejene nadrejenim ali kombiniranim zbirkam v tem kontejnerju.
- POPRAVLJANJE sekvenčnih datotek, če so spremenjene liste elementov za te zbirke zapisov. To ni potrebno, če je spremenjena samo logična struktura (dodan ali odvzet set za zapis).
- DBPUT — Prepis sekvenčnih datotek v nadrejene zbirke zapisov v BAZI.
- DBPUT — Prepis sekvenčnih datotek v kombinirane zbirke zapisov v BAZI.
- DBPUT — Prepis sekvenčnih datotek v podrejene zbirke zapisov v BAZI.

*6) RAZŠIRITEV PODREJENEGA ZAPISA

- DDC Sprememba opisa fizične strukture z izdelavo novega deskriptorja. Glej primer A.
- DBF Razširitev zapisa v kontejnerju.

*7) REFORMAT NADREJENEGA ZAPISA

- DBGET Prepis nadrejene zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.
- DBGET Prepis vseh podrejenih zbirk te nadrejene zbirke v sekvenčne datoteke.
- DBF REFORMAT nadrejene zbirke zapisa.
- DBF REFORMAT podrejenih zbirk zapisov.
- DBPUT Prepis iz sekvenčne datoteke v nadrejeno zbirko zapisa.
- DBPUT Prepis iz sekvenčnih datotek v podrejene zbirke zapisov.

*8) RAZŠIRITEV NADREJENEGA ZAPISA

- DBGET Prepis nadrejene zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.
- DBGET Prepis podrejenih zbirk tega zapisa v sekvenčne datoteke.
- DDC Sprememba opisa fizične strukture z izdelavo novega deskriptorja) Glej primer A.
- DBF Razširitev nadrejenega zapisa.
- DBF Reformat nadrejenega zapisa.
- DBF Reformat podrejenih zapisov.
- DBPUT Prepis sekvenčnih datotek v nadrejene zbirke.
- DBPUT Prepis sekvenčnih datotek v podrejene zbirke.

A9) REFORMAT KOMBINIRANEGA ZAPISA

- DBGET Prepis kombinirane zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.
- DBGET Prepis vseh podrejenih zbirk te kombinirane zbirke v sekvenčne datoteke.
- DBF REFORMAT kombinirane zbirke zapisa.
- DBF REFORMAT podrejenih zbirk zapisov.
- DBPUT Prepis iz sekvenčne datoteke v kombinirano zbirko zapisa.
- DBPUT Prepis iz sekvenčnih datotek v podrejene zbirke zapisov.

A10) RAZŠIRITEV KOMBINIRANEGA ZAPISA

- DBGET Prepis kombinirane zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.
- DBGET Prepis podrejenih zbirk tega zapisa v sekvenčne datoteke.
- DDC Sprememba opisa fizične strukture z izdelavo novega deskriptorja. Glej primer A.
- DBF Razširitev kombiniranega zapisa.
- DBF Reformat kombiniranega zapisa.
- DBF Reformat podrejenih zapisav.
- DBPUT Prepis sekvenčne datoteke v kombinirano zbirko.
- DBPUT Prepis sekvenčnih datotek v podrejene zbirke.

Primer: A

```
LOGICAL container name is NAROCILA
CONTAINER file name is NAROCILA.CON
CONNECT ...
CONNECT record NAROCI       ' Opis pred razširitvijo
OCCURENCY number is 2000    ' ...
BLOCK contains 7            ' ...
' Razširitev obvezno vpišemo na koncu kontejnerja
' kadar razširjamo zapis v že obstoječem kontejnerju.

CONNECT record NAROCI       ' Razširitev podrejenega
OCCURENCY number is 500     ' za 500 zapisov

LOGICAL container ....)     ' Opis pred razširitvijo
' Razširitev lahko vpišemo tudi v nov kontejner.
' Ta kontejner mora biti opisan kot zadnji v obstoječi
' fizični strukturi.
```

A.2. Reorganizacija nadrejenih in kombiniranih zapisov na DELTA/V s servisnim programom DBRWO

Tabela A-2: Vrste reorganizacij na nadrejenih zbirkah zapisov posebej za DELTA/V

Tip zapisa	Spremembe v fizični strukturi	Dolžina zapisa NI spremenjena	Dolžina zapisa JE spremenjena*1
NADREJENI	Velikost zbirke NI spremenjena*2	REFORMAT*7	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
ZAPISI	Velikost zbirke JE spremenjena*3	RAZŠIRITEV*8	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
KOMBINIRANI	Velikost zbirke NI spremenjena*2	REFORMAT*9	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5
ZAPISI	Velikost zbirke JE spremenjena*3	RAZŠIRITEV*10	REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU*5

Na DELTA/V opreracijskem sistemu obstaja za reorganizacijo nadrejenih zbirk zapisov poseben servisni program DBRWO. Uporaba in podroben opis tega programa je na SYS$UPDATE:DBVRWO.DOC. Glavna značilnost programa DBRWO je, da pri prepisu nadrejenih zbirk zapisov v sekvenčne datoteke, prepiše tudi informacije o povezavah (linkih). Zato ni potrebno pri reorganizacijah nadrejenih zbirk zapisov reorganizirati tudi njihovih podrejenih zapisov.

*1) Dolžina zapisa se spremeni zaradi dodane ali odvzete povezave (linka) v logični strukturi, in dodanih ali odvzetih elementov v zapisu v strukturi sheme.

*2) Velikost zbirke zapisa je število zapisov te zbirke zapisov opisane v Fizični strukturi sheme. To število se ne spremeni.

*3) Število zapisov določene zbirke zapisov opisane v Fizični strukturi se spremeni.

*5) REORGANIZACIJA VSEH ZAPISOV V KONTEJNERJU

- DBGET Prepis vseh podrejenih zbirk zapisov iz kontejner datoteke v sekvenčne datoteke.

- DBRWO Prepis vseh nadrejenih zbirk zapisov in kombiniranih zbirk zapisov.

- DDC Sprememba opisa sheme ali logične strukture ali fizične strukture in izdelava novih deskriptorjev za zbirke v tem kontejnerju. Glej primer A.

- DELETE Brisanje kontejnerja.

- DBF Primarno formatiranje zapisov v tem kontejnerju.

- POPRAVLJANJE sekvenčnih datotek, če so spremenjene liste elementov za te zbirke zapisov. To ni potrebno, če je spremenjena samo logična struktura (dodan ali odvzet set za zapis).

- DBPUT Prepis sekvenčnih datotek v nadrejene zbirke zapisov v BAZI.

- DBPUT Prepis sekvenčnih datotek v kombinirane zbirke zapisov v BAZI.

- DBPUT Prepis sekvenčnih datotek v podrejene zbirke zapisov v BAZI.

*7) REFORMAT NADREJENEGA ZAPISA

- DBRWO/GET Prepis nadrejene zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.

- DBF REFORMAT nadrejene zbirke zapisa.

- DBRWO/PUT Prepis iz sekvenčne datoteke v nadrejeno zbirko zapisa.

*8) RAZSIRITEV NADREJENEGA ZAPISA

- DBRWO/GET Prepis nadrejene zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.

- DDC Sprememba opisa fizične strukture z izdelavo novega deskriptorja. Glej primer A.

- DBF Razširitev nadrejenega zapisa.

- DBF Reformat nadrejenega zapisa.

- DBRWO/PUT Prepis sekvenčnih datotek v nadrejene zbirke.

*9) REFORMAT KOMBINIRANEGA ZAPISA

- DBRWO/GET Prepis kombinirane zbirke (samo INDEX) zapisa iz kontejner datoteke v sekvenčno datoteko.

- DBF REFORMAT kombinirane (samo INDEX) zbirke zapisa.

- DBRWO/PUT Prepis iz sekvenčne datoteke v kombinirano (samo INDEX) zbirko zapisa.

*10) RAZŠIRITEV KOMBINIRANEGA ZAPISA

- DBRWO/GET Prepis kombinirane (samo INDEX) zbirke zapisa iz kontejner datoteke v sekvenčno datoteko.

- DDC Sprememba opisa fizične strukture z izdelavo novega deskriptorja. Glej primer A.

- DBF Razširitev kombiniranega zapisa.

- DBF Reformat kombiniranega (samo INDEX) zapisa.

- DBRWO/PUT Prepis sekvenčne datoteke v kombinirano (samo INDEX) zbirko.

UKAZI IN FUNKCIJE JEZIKA ZA UPRAVLJANJE S PODATKI

BYE
Zaključek dela z bazo podatkov na trenutno aktivni podshemi.
Format ukaza: CALL "BYE"

CANCEL
Od zadnje COMMIT, CANCEL ali HELLO operacije, preklic vseh sprememb, narejenih na bazi podatkov.
Format ukaza: CALL "CANCEL"

COMMIT (velja tudi CONFRM)
Spremembe na bazi podatkov se za tem ukazom dejansko implementirajo na disku. Istočasno se sprostijo vsi zaklenjeni zapisi.
Format ukaza: CALL "COMMIT"

DBMIO
Na bazi podatkov se izvede imenovana vhodno/ izhodna funkcija.
Format ukaza: CALL "DBMIO" USING funkcija, ime-programskega-zapisa, V/I-področje, ključ

DELG
Funkcija, ki se navede v DBMIO ukazu za brisanje neodvisnega zapisa definiranega s ključem, ali brisanje odvisnega zapisa, kateremu je bil trenutno izvršen pristop.

GETD
Funkcija, ki se navede v DBMIO ukazu za branje (in zaklepanje) odvisnega zapisa, katerega DB ključ zapisa se nahaja v DB registru: začetni kazalec.

GETG
Funkcija, ki se navede v DBMIO ukazu za branje (in zaklepanje) neodvisnega zapisa definiranega s ključem, ali branje (in zaklepanje) seta odvisnega zapisa, določenega s ključem povezave.

GETP
Funkcija, ki se navede v DBMIO ukazu za branje (in zaklepanje) vseh vrst tipov zapisov po istem vrstnem redu, kot so zapisani na disk.

GETR
Funkcija, ki se navede v DBMIO ukazu za branje (in zaklepanje) odvisnih zapisov v obratnem vrstem redu kot pri GETG (od konca seta proti začetku).

HELLO (velja tudi READY)
Za definirano podshemo: Inicializacija dela z bazo podatkov.
Format ukaza: CALL "HELLO" USING ime-podsheme, interni-registri, geslo

INSA
Funkcija, ki se navede v DBMIO ukazu za dodajanje odvisnega zapisa v set pred zapis, kateremu je bil trenutno izvršen pristop.

INSB
Funkcija, ki se navede v DBMIO ukazu za dodajaje odvisnega zapisa v set za zapis, kateremu je bil trenutno izvršen pristop.

INSG
Funkcija, ki se navede v DBMIO ukazu za dodajanje neodvisnega zapisa določenega z izbranim ključem, ali dodajanje odvisnega zapisa na konec seta.

LOGDAT
Zapisovanje informacij v datoteko, skupno za vse programe.
Format ukaza: CALL "LOGDAT" USING V/I-področje, dolžina

RWRG
Funkcija, ki se navede v DBMIO ukazu za ažuriranje nadrejenega zapisa, določenega z izbranim ključem, ali ažuriranje podrejenega zapisa, kateremu je bil nazadnje izvršen pristop (ali zapis katerega DB ključ je v DB registru začetni kazalec).

SCLO
Funkcija (navede se v SEQIO ukazu), ki zapre sekvenčno datoteko.

SEQIO
Izvajanje vhodno/ izhodnih operacij na sekvenčnih datotekah.
Format ukaza: CALL "DBMIO" USING funkcija, ime-programskega-zapisa,V/I-področje,dolžina

SGET
Funkcija, ki se navede v SEQIO ukazu za branje zapisa iz sekvenčne datoteke.

SINS
Funkcija, ki se navede v SEQIO ukazu za dodajanje zapisa v sekvenčno datoteko.

SOPE
Funkcija (navede se v SEQIO ukazu), ki odpre sekvenčni zapis.

SRWD
Funkcija, ki se navede v SEQIO ukazu za pozicioniranje odprte sekvenčne datoteke na prvi zapis.

SRWR
Funkcija, ki se navede v SEQIO ukazu za ažuriranje zadnjega prebranega zapisa v sekvenčni datoteki.

Beležke:

DML SPOROČILA

**** Uspešno izvedena funkcija.

**FU Ena ali več zbirk zapisov je polna.

**GG Izvršena je funkcija GETG namesto specificirane. To se zgodi, kadar program navede nepredvideno funkcijo, npr. GETR za nadrejeni zapis.

**IG Izvršena funkcija INSG namesto specificirane. To se zgodi, kadar program navede nepredvideno funkcijo, npr. INSB za nadrejeni zapis.

**LK Na eni ali večih zbirkah zapisov niso dovoljene nobene funkcije razen branja. Potrebna je obnova.

**LL Ena ali več podrejenih zbirk zapisov je napolnjenih do specificirane meje.

DE01 Poškodovan kontrolni zapis. Dodajanje oziroma brisanje zapisa odvisnega ali kombiniranega tipa je sicer uspešno izvedeno, vendar je IDA Baza pri tem odkrila poškodovani Kontrolni zapis. Običajno to povzroči napaka na aparaturni opremi. Potrebna je obnova te zbirke zapisov.

DE02 V podshemi navedeno operativno področje ni prisotno. Aktivirano je bilo drugo operativno področje; napaka ponavadi pomeni neažurno podshemo.

DE03 Dostop do področja navedenega v podshemi ni dovoljen.

DE04 Zahtevana zbirka zapisov ni odprta. Navadno ta zbirka zapisov ni še formatirana, ali pa se je spremenila fizična struktura.

DE05 Zbirka zapisov ne obstaja v operativnem področju. Verjetno napačna podshema. Napaka se javi pri "HELLO" funkciji.

DE06 Napačen dostop do zapisa. Obvesti administratorja baze podatkov, ker se podshema (tip zapisa) ne ujema z operativnim področjem.

DE07 Zbirka zapisov je polna, dodajanje zapisov zato ni možno. Število zapisov v zbirki zapisov je doseglo maksimalno število zapisov, specificirano pri oblikovanju fizične strukture.

DE08 Nepravilna vsebina DB ključa. Napaka, ki se pogosto pojavi po nenormalnih prekinitvah, kadar logiranje transakcij ni bilo aktivno. Potrebna je obnova ustrezne zbirke zapisov.

DE09 V/I napaka. (Usodna napaka) Napaka pri poizkusu branja ali pisanja na disk. Glej dodatno sporočilo, ki identificira tip napake (na konzolnem terminalu ali "DBV_SHEMA:ime sheme.LOG").

DE10 Premajhno komunikacijsko področje. Napaka v podshemi. Verjetno je isti element zapisa naveden večkrat, npr. zaradi redefinicije. Potrebno je popraviti definicijo tega programskega zapisa v podshemi.

DE11 Napačno mesto zapisa. (Usodna napaka) DB ključ kaže izven prostora, namenjenega zbirki zapisov na disku, zaradi napake na aparaturni opremi ali pokvarjene baze podatkov. Obnovitev ene ali več zbirk zapisov je nujna. Napaka se pojavi, če transakcijsko logiranje ni aktivno.

DE12 Fizična struktura zbirke zapisov ni pravilno formatirana.

DE13 Zbirka zapisov je napolnjena preko meje 85%. Možno je še dodajanje, dokler se ne vrne status DE07, vendar to ni priporočljivo. Nujno je potrebno povečati ustezno zbirko zapisov.

DE14 Zbirka zapisov je zaključenja. (Usodna napaka) Ta napaka se pojavi samo, ko pride do nasilne prekinitve procesa baze podatkov, baza podatkov pa ni bila aktivirana s transakcijskim logiranjem. Če je bilo pri aktiviranju baze izbrano funkcijsko logiranje, potem je možno vzpostaviti korektno stanje baze podatkov iz zadnje kopije in izvesti obnovo na podlagi arhivske datoteke. Če pri aktiviranju baze ni bilo izbrano nobeno logiranje, potem je možna obnova baze samo iz zadnje kopije DB kontejnerjev.

DE15 Definicija povezave pri branju podrejenega zapisa kaže na zapis, ki sploh ni vključen v aktivni podshemi. Potrebno je ažurirati podshemo, glede na operativno področje.

DE16 V/I PODROČJE za navedeni zapis ni definirano v operativnem področju. V podshemi je izbran zapis, za katerega še ne obstaja fizična struktura.

DE17 Pri izvajanju funkcije GET-X na podrejenem zapisu je bil prebran zapis, pri katerem je primarni ključ prazen.

DE18 Rezervacijska lista je polna, dodatne rezervacije niso možne. Če je aktivno logiranje transakcij, potem je trenutna transakcija zavrnjena. Administrator baze podatkov lahko poveča listo tako da se pojavi opis operativnega področja.

DE19 Poizkus ažuriranja zaklenjene (status **LK) zbirke zapisov. Možno je samo branje.

DE20 Preveliko število sočasno dovoljenih aktivnih programov. Tabela aktivnih programov je polna. Ta status se vrne pri izvajanju funkcije "HELLO".

DE21 Napačno definiran pristop do področja. Obstaja možnost, da je podshema (datoteka, kjer je opis podsheme) pokvarjena.

DE22 Predolg deskriptor podsheme (DELTA/M).

DE25 Interna napaka v programu DBMUSE.

DI01 Poizkus dodajanja zapisa z enakim ključem, ko ponovitev ni dovoljena. Ključ nadrejenega zapisa že obstaja.

DI02 Nepravilno dodajanje zapisa. Eno od polj, ki bi moralo vsebovati ključ, je prazno.

DI03 Nepravilno ime podatkovnega polja v podshemi. Neažurna podshema.

DI04 Zapis je rezerviral drugi program. Program mora počakati, da se zapis sprosti.

DI05 Napaka pri brisanju nadrejenega zapisa. Pri poizkusu brisanja nadrejenega zapisa je IDA Baza ugotovila, da obstaja vsaj še eden podrejeni zapis, ki je nanj vezan.

DI06 Napačna oznaka zapisa. Napaka pri definiciji podsheme.

DI07 Napačen DB register začetni kazalec. IDA Baza pred izvršitvijo zahtevane funkcije preveri vsebino DB registra začetni kazalec in, če ugotovi napako, vrne DI07. Če bi se funkcija nadaljevala, bi se vrnil status DE11 - usodna napaka.

DI08 Napačna definicija povezave v podshemi. Podshema ni ažurna glede na aktivno področje.

DI09 Nadrejeni zapis z definiranim ključem ne obstaja. Zapis s tem ključem še ni bil dodan.

DI10 Zapis ni rezerviran (zaklenjen). Pred modifikacijo mora program prebrati zapis na pravilen način. Npr. za zbirko zapisov, ki jo hočemo ažurirati moramo prej prebrati s programskim zapisom, ki ima definirano "RWRG" pravico dostopa.

DI11 Nepravilni sekundarni ključ. Pri vsakem dodajanju podrejenjih zapisov je ena od povezav primarna, druge so pa sekundarne. Ta status pomeni, da nadrejeni zapis za primarno povezavo sicer obstaja, vendar manjka eden od nadrejenih zapisov za sekundarno povezavo.

DI12 Parameter ključ v DB klicu baze ne odgovarja polju v V/I področju.

DI13 Pri brisanju ali dodajanju podrejenega zapisa je ugotovljeno, da enega izmed nadrejenih zapisov ni možno spreminjati (ažurirati). Obstaja možnost, da je eden od nadrejenih zapisov pokvarjen, ali pa se aktivna podshema ne ujema z operativnim področjem.

DI14 Pokvarjena baza podatkov na diskih. Pri izvajanju funkcije "DELG" nad odvisnimi zapisi ni najden eden od nadrejenih zapisov.

DI15 Napaka pri brisanju kombiniranega zapisa. Na kombinirani zapis je vezan vsaj eden odvisni tip zapisa.

DI16 Povezovalna pot (set) odvisnega zapisa ni definirana. Neažurna podshema glede na aktivno področje.

DI17 Napaka pri dodajanju nadrejenega zapisa. Uporabljena ja bila funkcija "INSA" ali "INSB" namesto INSG. Dodajanje po kontrolnem ključu je možno samo s funkcijo INSG.

ENO1 Usodna napaka v IDA Baza okolju. Ta status se vrne v primeru, da je prišlo do usodne napake, uporabnik pa še vedno poizkuša izvajati IDA Baza funkcije.

ENO2 Običajno IDA Baza ni aktivna.

END. Konec seta ali konec zbirke zapisov. Konec zapisov v nekem setu ali konec vseh zapisov pri branju s funkcijo GETP.

FPO1 Na definirani sekvenčni datoteki je "OPEN" že izvršen.

FPO2 Napaka pri izvajanju V/I operacij na sekvenčnih datotekah. Statusi so enako kot RMS statusi. IDA Baza vrne RMS STS status v prvi DB register, ki je predviden za SQ status, v drugem pa je RMS STVS. Ti statusi so opisani v ustreznih RMS priročnikih.

FPO3 Definirana sekvenčna datoteka ni odprta.

FPO4 Preseženo je število sočasno odprtih datotek.

LG01 Presežено je maksimalno število sočasno aktivnih programov (DELTA/M).

LG02 Uporabnik je navedel napačno geslo. Uporabnik ni pooblaščen za uporabo navedene podsheme.

LG03 Podshema s tem imenom ne obstaja.

LG04 Program nima pravice izvajati sekvenčnih funkcij (DELTA/M).

LG05 "LOGER" ni aktiven (DELTA/M).

LG06 Presežena je največja dovoljena dolžina logiranih podatkov pri logiranju funkcij. Interna napaka (Usodna napaka).

LG07 V/I napaka pri "logiranju funkcij". (Usodna napaka) To je napaka, ki se pojavi pri pisanju arhivske datoteke, zato je potrebno proučiti tudi vrnjen RMS status. Če se pojavi ta napaka, je potrebno delo z bazo čim hitreje zaključiti, narediti novo verzijo kopije baze in ponovno začeti z delom.

LG08 Presežena je dovoljena dolžina logiranih podatkov pri funkciji LOGDAT (480 bytov).

LG20 Napaka pri alokaciji transakcijskega vmesnega pomnilnika. (Opozorilna napaka) Do te napake pride zaradi pomanjkanja fizične memorije na DELTA/M ali zaradi pomanjkanja virtualnega adresnega prostora na DELTA/V (PGFLQUOTA).

LG21 Logiranje transakcij ni aktivno. (Opozorilna napaka) Aplikacijski program poizkuša izvesti "CONFRM" ali "CANCEL", čeprav je bilo logiranje transakcij prekinjeno zaradi napake.

LG22 Transakcijski vmesni pomnilnik je poln do 85%. (Opozorilna napaka) Izvedi "CONFRM" (ali "CANCEL") brž ko je možno.

LG23 Transakcijski vmesni pomnilnik je poln. (Usodna napaka) IDA Baza ne more logirati vseh sprememb na bazi za aplikacijski program. Za to transakcijo je avtomatsko izveden "CANCEL". Spremembe za zadnjo transakcijo so izgubljene.

LG24 I/O napaka pri pisanju transakcijske datoteke. (Opozorilna napaka) V/I napaka je nastala pri pisanju transakcijskega vmesnega pomnilnika na disk. Logiranje transakcij je zaustavljeno in vsi naslednji "CONFRM" (ali "CANCEL") bodo dobili LG21 status. Potrebno je takoj zaustaviti IDA Bazo in odpraviti vzroke V/I napake.

LG25 Interna napaka pri "CONFRM". (Usodna napaka) Pri implementaciji sprememb v bazo podakov na disku je prišlo do I/O napake. Logiranje transakcij je zaustavljeno in vsi naslednji "CONFRM" (ali "CANCEL") bodo dobili LG21 status. Potrebno je takoj zaustaviti IDA Bazo in odpraviti vzroke V/I napake.

LG26 Transakcija je bila prekinjena. (Opozorilna napaka) Drugi aplikacijski program je ukradel enega od rezerviranih zapisov, ker je time-out za transakcijo potekel.

PR01 Napačno število parametrov. Napačni ali manjkajoči parametri v DB klicu.

PR02 Ponovljena "HELLO" funkcija brez vmesne "BYE" funkcije.

PR03 Napačna funkcija. Parameter funkcija v DB klicu je napačen, ali pa ta parameter ni na meji besede (velja samo za DELTA/M). Običajno pa se ta status javi, če programski zapis nima ustreznih pristopnih pravic.

PR04 Napačno branje zapisa pri branju označenih zapisov. Interna napaka običajno neažurna podshema glede na operativno področje.

PR05 Napačno ime programskega zapisa.

PR06 Poizkus izvajanja katerekoli funkcije, če funkcija "HELLO" ni bila (uspešno) izvršena.

PR07 Napačna verzija podsheme.

PRIMER PROGRAMA

```
*-----------------------------------------------------------
*
IDENTIFICATION DIVISION.
*
PROGRAM-ID.    DBDEMO.
*
* NASLOV:      DEMONSTRACIJSKI-PROGRAM
*
AUTHOR.        ISKRA DELTA      .
*
DATE-WRITTEN.  26-JUN-1985.
*
* VERZIJA:     1.0
*
*-----------------------------------------------------------
```

```
*+----------------------------------------------------------
*
*                 O P I S    P R O G R A M A
*
*
* NALOGA PROGRAMA:
*                   Program je napisan, kot primer programiranja
*                   z IDA-BAZA programskimi orodji in ni primer
*                   lepega programiranja. Je sestavni del
*                   prirocnika za programerje, ki bodo le-ta
*                   orodja uporabljali.
*
*                   GESLO za izvajanje programa (iz PODSHEME) se
*                   imenuje :   "PRODAJ"
*
* VHOD:
*                   Vsi podatki, ki sluzijo za vhod se vnasajo
*                   preko terminala. Zaradi bolj nazornega prikaza
*                   uporabe IDA-BAZA orodij, smo uporabili stan-
*                   dardni COBOL nacin (DISPLAY,ACCEPT) in ne
*                   IDA-EKRAN orodij.
*
* IZHOD:
*                   Vsi izpisi podatkov so ravno tako izpisani
*                   preko terminala z DISPLAY ukazi.
*
*
* SPREMEMBE:
*
*    Avtor:          Identifikacija:        Datum:
*
*
*    Opis :
*
*
*----------------------------------------------------------
```

```
*-----------------------------------------------------------------
*
ENVIRONMENT DIVISION.
*
*-----------------------------------------------------------------

CONFIGURATION SECTION.

SOURCE-COMPUTER. DELTA-V.
OBJECT-COMPUTER. DELTA-V.
SPECIAL-NAMES.
       DECIMAL-POINT IS COMMA.
```

```
*-----------------------------------------------------------------
*
DATA DIVISION.
*
*-----------------------------------------------------------------

WORKING-STORAGE SECTION.


***             IDA-BAZA POLJA IN KONSTANTE

***             Polja za izpis usodne napake
01      LOG-BUFER.
 05     LOG-BUFER-1.
  10    STATE           PIC X(05)  VALUE "START".
  10    FILLER          PIC X      VALUE "/".
  10    IME-PROGRAMA    PIC X(08)  VALUE "DBDEMO".
  10    FILLER          PIC X      VALUE "/".
  10    VERZIJA         PIC XX     VALUE "01".
 05     LOG-BUFER-2.
  10    DB-STATS        PIC X(04).
  10    FILLER          PIC X      VALUE "/".
  10    DB-PODSHEMA     PIC X(09)  VALUE "PRODAJITI".
  10    FILLER          PIC X      VALUE "/".
  10    DB-IME-ZAPISA   PIC X(05).
  10    FILLER          PIC X      VALUE "/".
  10    DB-KLJUC        PIC X(20).
  10    FILLER          PIC XX     VALUE "/".
  10    DB-FUNC         PIC X(04).
 05     LOG-BUFER-3.
  10    SB-FUNC         PIC X(04).
  10    FILLER          PIC X      VALUE "/".
  10    SQ-STATUS-1     PIC -9(4).
  10    FILLER          PIC X      VALUE "/".
  10    SQ-STATUS-2     PIC -9(4).

01      LOG-BUFER-DOL   PIC S9(04) COMP VALUE 15.
01      EXIT-STAT       PIC S9(04) COMP VALUE 1.
```

```
***                    Polja za kontekst IDA_BAZA skupaj
***                    s statusi, ki jih zelimo kontrolirati
01      DB-INTERNI-REGISTRI.
 05     DB-STAT.
*                              PRAVILNO DOKONCAN KLIC BAZE
       88  STAR      VALUE "*****".
*                              ZAPIS NE OBSTOJA
       88  OWNF      VALUE "DI01".
*                              KONEC VERIGE
       88  ENDP      VALUE "END.".
*                              ZAPIS UPORABLJA DRUG UPORABNIK
*                        IN    ZAPIS NI ZAKLENJEN
       88  HELD      VALUE "DI10","DI04".
*                              DVOJNI KLJUC
       88  DUPM      VALUE "DI02".
*                              NEPRAVILNO BRISANJE - OBSTAJA SE POVEZAVA
       88  IMDL      VALUE "DI05".
*                              AVTOMATSKI "CANCEL"
       88  CNCL      VALUE "LG26".
*                              TRANSAKCIJE NAPOLNJENE DO 85%
       88  TRLL      VALUE "LG22".
  10    DB-STAT-1        PIC XX.
       88  WARNING   VALUE "**".
  10    DB-STAT-2        PIC XX.
 05     ZAC-KAZALEC      PIC S9(9) COMP.
 05     SQ-STAT1         PIC S9(9) COMP.
 05     SQ-STAT2         PIC S9(9) COMP.
 05     REZERVA          PIC X(04).
 05     TEK-KAZALEC      PIC S9(9) COMP.

***                    Vse funkcije, ki jih uporabljamo
01      GETG            PIC X(04) VALUE "GETG".
01      GETR            PIC X(04) VALUE "GETR".
01      GETD            PIC X(04) VALUE "GETD".
01      GETP            PIC X(04) VALUE "GETP".
01      INSG            PIC X(04) VALUE "INSG".
01      INSA            PIC X(04) VALUE "INSA".
01      INSB            PIC X(04) VALUE "INSB".
01      RWRG            PIC X(04) VALUE "RWRG".
01      DELG            PIC X(04) VALUE "DELG".
```

```
*COPY "DBV_SHEMA:PRODAJ1T1.LIB".
***                    Slovar podshema obicajno vkljucimo s "COPY"
***                    tu pa je vkljucen v program zaradi vecje
***                    nazornosti.
*=============================================================
* 28-JUN-85   7:35:55  DBV_SHEMA:PRODAJ1T1.LIB
*=============================================================
01      SHEMA       PIC X(6)   VALUE "PRODAJ".
01      PODROCJE    PIC X(7)   VALUE "PRODAJ1".
01      PODSHEMA    PIC X(9)   VALUE "PRODAJ1T1".
01      PROJEKT     PIC X(8)   VALUE "DEMODB  ".
01      GESLO       PIC X(6).
*=============================================================
*       ZAPIS       = KUPCII
*       TIP ZAPISA  = OWNER
*       FUNKCIJA    = GETP GET
*=============================================================
01      KUPCII001   PIC X(9)   VALUE "KUPCII001".
01      KUPCII-001.
        05          KUPCIIOWNKEY    PIC X(6).
        05          KUPCIIIMEKUP    PIC X(50).
        05          KUPCIINASLOV    PIC X(60).
        05          KUPCIITELEFO    PIC 9(9).
        05          KUPCIISIFDEO    PIC X(6).
```

```
*-------------------------------------------------------------
*       ZAPIS       = KUPCII
*       TIP ZAPISA  = OWNER
*       FUNKCIJA    = GET  INS  RWR
*-------------------------------------------------------------
01      KUPCII002  PIC X(9)   VALUE "KUPCII002".
01      KUPCII-002.
        05          KUPCIIOWNKEY    PIC X(6).
        05          KUPCIIIMEKUP    PIC X(50).
        05          KUPCIINASLOV    PIC X(60).
        05          KUPCIITELEFO    PIC 9(9).
        05          KUPCIISIFDEO    PIC X(6).
*-------------------------------------------------------------
*       ZAPIS       = IZDLKI
*       TIP ZAPISA  = OWNER
*       FUNKCIJA    = GETP GET
*-------------------------------------------------------------
01      IZDLKI001  PIC X(9)   VALUE "IZDLKI001".
01      IZDLKI-001.
        05          IZDLKIOWNKEY    PIC X(12).
        05          IZDLKIIMEIZD    PIC X(60).
        05          IZDLKICENAIZ    PIC 9(7)V9(2).
        05          IZDLKIKOLICI    PIC 9(7)V9(3).
*-------------------------------------------------------------
*       ZAPIS       = IZDLKI
*       TIP ZAPISA  = OWNER
*       FUNKCIJA    = GET  INS  RWR
*-------------------------------------------------------------
01      IZDLKI002  PIC X(9)   VALUE "IZDLKI002".
01      IZDLKI-002.
        05          IZDLKIOWNKEY    PIC X(12).
        05          IZDLKIIMEIZD    PIC X(60).
        05          IZDLKICENAIZ    PIC 9(7)V9(2).
        05          IZDLKIKOLICI    PIC 9(7)V9(3).
*-------------------------------------------------------------
*       ZAPIS       = IZDLKI
*       TIP ZAPISA  = OWNER
*       FUNKCIJA    = GET  INS  RWR
*-------------------------------------------------------------
01      IZDLKI003  PIC X(9)   VALUE "IZDLKI003".
01      IZDLKI-003.
        05          IZDLKIOWNKEY    PIC X(12).
        05          IZDLKICENAIZ    PIC 9(7)V9(2).
        05          IZDLKIKOLICI    PIC 9(7)V9(3).
*-------------------------------------------------------------
*       ZAPIS       = NAROCI
*       TIP ZAPISA  = OWNER IN MEMBER
*       FUNKCIJA    = GETP GET
*-------------------------------------------------------------
01      NAROCI001  PIC X(9)   VALUE "NAROCI001".
01      NAROCI-001.
        05          NAROCIOWNKEY    PIC X(5).
        05          NAROCISIFKUP    PIC X(6).
        05          NAROCIDATNAR    PIC X(6).
        05          NAROCIROKDOB    PIC X(6).
*-------------------------------------------------------------
*       ZAPIS       = NAROCI
*       TIP ZAPISA  = OWNER IN MEMBER
*       FUNKCIJA    = GET  INS  RWR
*-------------------------------------------------------------
01      NAROCI002  PIC X(9)   VALUE "NAROCI002".
01      NAROCI-002.
        05          NAROCIOWNKEY    PIC X(5).
        05          NAROCISIFKUP    PIC X(6).
        05          NAROCIDATNAR    PIC X(6).
        05          NAROCIROKDOB    PIC X(6).
```

```
*=================================================================
*       ZAPIS       = NAROCI
*       TIP ZAPISA  = OWNER IN MEMBER
*       FUNKCIJA    = GET  DEL
*=================================================================
01      NAROCI003  PIC X(9)   VALUE "NAROCI003".
01      NAROCI-003.
        05          NAROCIOWNKEY    PIC X(5).
        05          NAROCISIFKUP    PIC X(6).
        05          NAROCIDATNAR    PIC X(6).
        05          NAROCIROKDOB    PIC X(6).
*=================================================================
*       ZAPIS       = NAROCI
*       TIP ZAPISA  = OWNER IN MEMBER
*       FUNKCIJA    = GETP GET
*=================================================================
01      NAROCI004  PIC X(9)   VALUE "NAROCI004".
01      NAROCI-004.
        05          NAROCIOWNKEY    PIC X(5).
        05          NAROCISIFKUP    PIC X(6).
        05          NAROCIDATNAR    PIC X(6).
        05          NAROCIROKDOB    PIC X(6).
*=================================================================
*       ZAPIS       = NARIZD
*       TIP ZAPISA  = MEMBER
*       FUNKCIJA    = GETP GET
*=================================================================
01      NARIZD001  PIC X(9)   VALUE "NARIZD001".
01      NARIZD-001.
        05          NARIZDSTVNAR    PIC X(5).
        05          NARIZDSIFIZD    PIC X(12).
        05          NARIZDNARKOL    PIC 9(5)V9(3).
        05          NARIZDDOBKOL    PIC 9(5)V9(3).
*=================================================================
*       ZAPIS       = NARIZD
*       TIP ZAPISA  = MEMBER
*       FUNKCIJA    = GET  INS  RWR
*=================================================================
01      NARIZD002  PIC X(9)   VALUE "NARIZD002".
01      NARIZD-002.
        05          NARIZDSTVNAR    PIC X(5).
        05          NARIZDSIFIZD    PIC X(12).
        05          NARIZDNARKOL    PIC 9(5)V9(3).
        05          NARIZDDOBKOL    PIC 9(5)V9(3).
*=================================================================
*       ZAPIS       = NARIZD
*       TIP ZAPISA  = MEMBER
*       FUNKCIJA    = GET  RWR
*=================================================================
01      NARIZD003  PIC X(9)   VALUE "NARIZD003".
01      NARIZD-003.
        05          NARIZDSTVNAR    PIC X(5).
        05          NARIZDSIFIZD    PIC X(12).
        05          NARIZDNARKOL    PIC 9(5)V9(3).
        05          NARIZDDOBKOL    PIC 9(5)V9(3).
```

```
*================================================================
*       ZAPIS       = NARIZD
*       TIP ZAPISA  = MEMBER
*       FUNKCIJA    = GET  DEL
*================================================================
01      NARIZD004  PIC X(9)   VALUE "NARIZD004".
01      NARIZD-004.
        05          NARIZDSTVNAR     PIC X(5).
        05          NARIZDSIFIZD     PIC X(12).
        05          NARIZDNARKOL     PIC 9(5)V9(3).
        05          NARIZDDOBKOL     PIC 9(5)V9(3).
*================================================================
*       ZAPIS       = NARIZD
*       TIP ZAPISA  = MEMBER
*       FUNKCIJA    = GETP GET
*================================================================
01      NARIZD005  PIC X(9)   VALUE "NARIZD005".
01      NARIZD-005.
        05          NARIZDSTVNAR     PIC X(5).
        05          NARIZDSIFIZD     PIC X(12).
        05          NARIZDNARKOL     PIC 9(5)V9(3).
        05          NARIZDDOBKOL     PIC 9(5)V9(3).
*================================================================
****                              Programske spremenljivke in
****                              pomozna polja
*----------------------------------------------------------------
01       DA-NE   PIC X.
  88     DA-NE-OK   VALUE "D","N".
  88     NE       VALUE "N".
  88     JA       VALUE "D".
01       AKCIJA     PIC 9.
01       KLJUC-IZDLK      PIC X(12).
01       KLJUC-NARO       PIC X(5).
01       KLJUC-KUPC       PIC X(6 .
01       CIFRA            PIC X(1 ).
01       I-DESET          PIC Z(6)9,9(3).
01       I-DEVET          PIC Z(6)9,9(2).
01       I-OSEM           PIC Z(4)9,9(3).
01       VHOD             PIC X(40).
01       IZHOD            PIC S9(13)V9(5) COMP-3.
01       ST-DECIMALK              PIC 99.
01       ST-CELIH                 PIC 99.
01       NAPACNO          PIC X.
01       ODDANO           PIC 9(5)V9(3).
01       ST-CALL          PIC 99.
01       REFERENCA        PIC S9(9) COMP.
01       CONFRMN         PIC X(30).
01       STARI-BUFFER     PIC X(60).
```

```
*-----------------------------------------------------------------
*
PROCEDURE DIVISION.
*
*-----------------------------------------------------------------
**** Program je izdelan po konceptu IDA-COGEN
*-----------------------------------------------------------------
GLAVNA-PROGRAMSKA SECTION.
*-----------------------------------------------------------------
ZACETEK-GLAVNA.

        PERFORM ZACETEK-IDA-BAZA.
        PERFORM LOGPRO.
        GO TO MENU.
KONEC-GLAVNA.
        MOVE   "KONEC" TO  STATE.
        PERFORM LOGPRO.
        CALL "BYE".                                              → 1
        CALL "NCEXIT" USING EXIT-STAT.

*-----------------------------------------------------------------
IDA-BAZA SECTION.
*-----------------------------------------------------------------
ZACETEK-IDA-BAZA.

***     Prikljucitev na bazo podatkov IDA-BAZA

        CALL "NCGESL" USING GESLO.

        CALL "HELLO"  USING  DB-PODSHEMA                         → 2
                             DB-INTERNI-REGISTRI
                             GESLO.
        IF NOT STAT
           MOVE DB-STAT   TO DB-STATS
           PERFORM FATERR.


***     Klici nizov IDA-BAZA

DB-KUPCII001.
        CALL "DBMIO" USING   DB-FUNC
                             KUPCII001
                             KUPCII-001
                             KLJUC-KUPC

        MOVE KUPCII001    TO DB-IME-ZAPISA.
        MOVE KLJUC-KUPC   TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-KUPCII002.
        CALL "DBMIO" USING   DB-FUNC
                             KUPCII002
                             KUPCII-002
                             KLJUC-KUPC

        MOVE KUPCII002    TO DB-IME-ZAPISA.
        MOVE KLJUC-KUPC   TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-IZDLKI001.
        CALL "DBMIO" USING   DB-FUNC
                             IZDLKI001
                             IZDLKI-001
                             KLJUC-IZDLK

        MOVE IZDLKI001    TO DB-IME-ZAPISA.
        MOVE KLJUC-IZDLK  TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.
```

```
DB-IZDLKI001-GETP.
        MOVE GETP TO DB-FUNC
        CALL "DBMIO" USING   DB-FUNC
                             IZDLKI001
                             IZDLKI-001.

        MOVE IZDLKI001    TO DB-INE-ZAPISA.
        MOVE IZDLKIONNKEY OF IZDLKI-001 TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-IZDLKI002.
        CALL "DBMIO" USING   DB-FUNC
                             IZDLKI002
                             IZDLKI-002
                             KLJUC-IZDLK

        MOVE IZDLKI002    TO DB-INE-ZAPISA.
        MOVE KLJUC-IZDLK  TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-IZDLKI003.
        CALL "DBMIO" USING   DB-FUNC
                             IZDLKI003
                             IZDLKI-003
                             KLJUC-IZDLK

        MOVE IZDLKI003    TO DB-INE-ZAPISA.
        MOVE KLJUC-IZDLK  TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-NAROCI001.
        CALL "DBMIO" USING   DB-FUNC
                             NAROCI001
                             NAROCI-001
                             KLJUC-NARO

        MOVE NAROCI001    TO DB-INE-ZAPISA.
        MOVE KLJUC-NARO   TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-NAROCI002.
        CALL "DBMIO" USING   DB-FUNC
                             NAROCI002
                             NAROCI-002
                             KLJUC-KUPC

        MOVE NAROCI002    TO DB-INE-ZAPISA.
        MOVE KLJUC-KUPC   TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-NAROCI003.
        CALL "DBMIO" USING   DB-FUNC
                             NAROCI003
                             NAROCI-003
                             KLJUC-NARO

        MOVE NAROCI003    TO DB-INE-ZAPISA.
        MOVE KLJUC-NARO   TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.

DB-NAROCI004.
        CALL "DBMIO" USING   DB-FUNC
                             NAROCI004
                             NAROCI-004
                             KLJUC-KUPC

        MOVE NAROCI004    TO DB-INE-ZAPISA.
        MOVE KLJUC-KUPC   TO DB-KLJUC.
        MOVE DB-STAT      TO DB-STATS.
```

```
DB-NARIZD001.
       CALL "DBMIO" USING   DB-FUNC
                            NARIZD001
                            NARIZD-001
                            KLJUC-IZDLK.

       MOVE NARIZD001    TO DB-INE-ZAPISA.
       MOVE KLJUC-IZDLK  TO DB-KLJUC.
       MOVE DB-STAT      TO DB-STATS.

DB-NARIZD001-GETP.
       MOVE GETP TO DB-FUNC
       CALL "DBMIO" USING   DB-FUNC
                            NARIZD001
                            NARIZD-001

       MOVE NARIZD001    TO DB-INE-ZAPISA.
       MOVE NARIZDSTVNAR OF NARIZD-001 TO DB-KLJUC.
       MOVE DB-STAT      TO DB-STATS.

DB-NARIZD002.
       CALL "DBMIO" USING   DB-FUNC
                            NARIZD002
                            NARIZD-002
                            KLJUC-NARO

       MOVE NARIZD002    TO DB-INE-ZAPISA.
       MOVE KLJUC-NARO   TO DB-KLJUC.
       MOVE DB-STAT      TO DB-STATS.

DB-NARIZD003.
       CALL "DBMIO" USING   DB-FUNC
                            NARIZD003
                            NARIZD-003
                            KLJUC-NARO.

       MOVE NARIZD003    TO DB-INE-ZAPISA.
       MOVE KLJUC-NARO   TO DB-KLJUC.
       MOVE DB-STAT      TO DB-STATS.

DB-NARIZD004.
       CALL "DBMIO" USING   DB-FUNC
                            NARIZD004
                            NARIZD-004
                            KLJUC-NARO.

       MOVE NARIZD004    TO DB-INE-ZAPISA.
       MOVE KLJUC-NARO   TO DB-KLJUC.
       MOVE DB-STAT      TO DB-STATS.
DB-NARIZD005.
       CALL "DBMIO" USING   DB-FUNC
                            NARIZD005
                            NARIZD-005
                            KLJUC-NARO.

       MOVE NARIZD005    TO DB-INE-ZAPISA.
       MOVE KLJUC-NARO   TO DB-KLJUC.
       MOVE DB-STAT      TO DB-STATS.

KONEC-IDA-BAZA.
```

```
*-----------------------------------------------------------------
NAPAKE SECTION.
*-----------------------------------------------------------------
ZACETEK-NAPAKE.

FATERB. MOVE "ERRDB"   TO STATE.
        MOVE 3         TO EXIT-STAT.
        MOVE 80        TO LOG-BUFER-DOL.
        PERFORM LOGERB.
        DISPLAY " ".
        DISPLAY "NAPAKA NASTOPILA V " ST-CALL ". KLICU BAZE".
* Zaradi usodne napake postavimo stanje v bazi podatkov na
* nivo zadnje uspesne potrditve in izpisemo sporocilo.
        CALL "CANCEL" USING CONFRNM.                              → 3
        DISPLAY "***************************************"
        DISPLAY "***** ZADNJA USPESNA TRANSAKCIJA *****"
        DISPLAY CONFRNM
        DISPLAY "***************************************"
        PERFORM KONEC-GLAVNA.

FATERP. MOVE 2         TO EXIT-STAT.
        MOVE "ERRPR"   TO STATE.
        PERFORM LOGPRO.
        PERFORM KONEC-GLAVNA.

KONEC-NAPAKE.
```

```
*-----------------------------------------------------------------
LOGIRANJE SECTION.
* Izpis nastopa napake v arhiv za organizatorje obdelav
*-----------------------------------------------------------------
ZACETEK-LOGIRANJE.

LOGPRO. DISPLAY " ".
        CALL "LOGDAT" USING  LOG-BUFER-1 LOG-BUFER-DOL.

LOGERB. DISPLAY LOG-BUFER-1.                                      → 4
        DISPLAY LOG-BUFER-2.
        CALL "LOGDAT" USING  LOG-BUFER   LOG-BUFER-DOL.

KONEC-LOGIRANJE.
```

```
*-----------------------------------------------------------
MENU SECTION.
*-----------------------------------------------------------
IZPISI-MENU.
       DISPLAY "=========================================="
       DISPLAY "IDA-BAZA        DEMO PROGRAM V 1.0  "
       DISPLAY "        OBDELAVA NAROCIL                  "
       DISPLAY "=========================================="
       DISPLAY " 1. VNOS KUPCEV                           "
       DISPLAY " 2. VNOS IZDELKOV                         "
       DISPLAY " 3. DODAJANJE NAROCIL IN NAROCENIH IZDELKOV"
       DISPLAY " 4. BRANJE PODATKOV O IZDELKU V NAROCILIH  "
       DISPLAY " 5. POPRAVLJANJE PODATKOV O NAROCILIH      "
       DISPLAY " 6. BRISANJE NAROCIL                      "
       DISPLAY " 7. CITANJE IZDELKOV /GETP/    "
       DISPLAY " 8. CITANJE NAROCENIH IZDELKOV            "
       DISPLAY " 9. KONEC                  "
       DISPLAY "=========================================="
       DISPLAY "VNESI ZELJENO STEVILKO RUTINE  : " NO ADVANCING
       ACCEPT AKCIJA.
       IF AKCIJA < 0 OR > 9
            DISPLAY " NAPACEN VNOS"
            GO TO MENU.
*-----------------------------------------------------------
* RAZMEJITEV NA OBDELAVE
*-----------------------------------------------------------
       GO TO   VNOS-KUPCEV
               VNOS-IZDELKOV
               DODAJ-NAROCILO
               BERI
               POPRAVI
               BRISI
               CITANJE-IZDELKOV
               CITANJE-NAR-IZD
               KONEC-GLAVNA
       DEPENDING ON AKCIJA.
MENU-EXIT.
       EXIT.
```

```
*****************************************************************
**** VNOS  KUPCEV
*****************************************************************
VNOS-KUPCEV SECTION.
ZACNI.
        DISPLAY "VNESI SIFRO KUPCA    (6) : " NO ADVANCING
        ACCEPT KLJUC-KUPC
        IF KLJUC-KUPC = SPACE
                DISPLAY "PRAZEN KLJUC"
                GO TO ZACNI.
        MOVE GETG TO DB-FUNC                                    5
**
        PERFORM DB-KUPCII001
**
        IF STAR
        DISPLAY "SIFRA KUPCA ZE OBSTAJA "
        GO TO DOBI-ODG.
        IF NOT OWNF
        MOVE 1 TO ST-CALL
        GO TO FATERB.
        DISPLAY "VNESI IME KUPCA      (50) : " NO ADVANCING
        ACCEPT KUPCIIIMEKUP OF KUPCII-002
        DISPLAY "VNESI NASLOV KUPCA   (60) : " NO ADVANCING
        ACCEPT KUPCIINASLOV OF KUPCII-002
        DISPLAY "VNESI TEL. ST. KUPCA (9) : " NO ADVANCING
        ACCEPT KUPCIITELEFO OF KUPCII-002
        DISPLAY "VNESI SIFRO DEL. ORG (6) : " NO ADVANCING
        ACCEPT KUPCIISIFDEO OF KUPCII-002.
        MOVE KLJUC-KUPC TO KUPCIIOWNKEY OF KUPCII-002.
HELD-ZAN.
        MOVE INSG TO DB-FUNC                                    6
**
        PERFORM DB-KUPCII002
**
        IF HELD
                GO TO HELD-ZAN.
        IF DUPM
                DISPLAY "TA TRENUTEK KUPCA DODAL NEKDO DRUG"
                GO TO DOBI-ODG.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 2 TO ST-CALL
                GO TO FATERB.
        STRING "* VNOS KUPCII-KEY = " KUPCIIOWNKEY OF KUPCII-002
                DELIMITED BY SIZE INTO CONFRM.
        CALL "CONFRM" USING CONFRM.                             7
        DISPLAY "    STATUS: " DB-STAT.
DOBI-ODG.
        DISPLAY "NADALJUJES ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG
        ELSE
                IF NE
                        GO TO VNOS-KUPCEV-EXIT.
        GO TO ZACNI.
VNOS-KUPCEV-EXIT.
        GO TO MENU.
```

```
*****************************************************************
****    VNOS IZDELKOV
*****************************************************************
VNOS-IZDELKOV SECTION.
ZACNI.
        DISPLAY "VNESI SIFRO IZDELKA   (12) : " NO ADVANCING
        ACCEPT KLJUC-IZDLK
        IF KLJUC-IZDLK = SPACE
                DISPLAY "PRAZEN KLJUC"
                GO TO ZACNI.
        MOVE GETG TO DB-FUNC
**
        PERFORM DB-IZDLKI001
**
        IF STAR
                DISPLAY "SIFRA IZDELKA ZE OBSTAJA"
                GO TO DOBI-ODG.
        IF NOT OWNF
                MOVE 3 TO ST-CALL
                GO TO PATERB.
        DISPLAY "VNESI IME IZDELKA     (60) : " NO ADVANCING
        ACCEPT IZDLKIIMEIZD OF IZDLKI-002.
SPREJMI-STEV1.
      DISPLAY "VNESI CENO IZDELKA : _______,__[10D" NO ADVANCING.
        ACCEPT CIFRA
        MOVE CIFRA TO VHOD
        MOVE 7 TO ST-CELIH
        MOVE 2 TO ST-DECIMALK
        CALL "DOBIST" USING  VHOD ST-CELIH ST-DECIMALK IZHOD NAPACNO
        IF NAPACNO = "E"
                DISPLAY "NAPACEN VNOS - NENUMERICNO"
                GO TO SPREJMI-STEV1.
        MOVE IZHOD TO I-DEVET
        DISPLAY "[1A[36C" I-DEVET.
        MOVE IZHOD TO IZDLKICENAIZ OF IZDLKI-002.
SPREJMI-STEV2.
        DISPLAY "VNESI KOLICINO :  _______,___[11D" NO ADVANCING.
        ACCEPT CIFRA
        MOVE CIFRA TO VHOD
        MOVE 7 TO ST-CELIH
        MOVE 3 TO ST-DECIMALK
        CALL "DOBIST" USING  VHOD ST-CELIH ST-DECIMALK IZHOD NAPACNO
        IF NAPACNO = "E"
                DISPLAY "NAPACEN VNOS - NENUMERICNO"
                GO TO SPREJMI-STEV2.
        MOVE IZHOD TO I-DESET
        DISPLAY "[1A[36C" I-DESET.
        MOVE IZHOD TO   IZDLKIKOLICI OF IZDLKI-002.
HELD-ZAN.
        MOVE INSG TO DB-FUNC
        MOVE KLJUC-IZDLK TO IZDLKIOWNKEY OF IZDLKI-002.
**
        PERFORM DB-IZDLKI002
**
        IF HELD
                GO TO HELD-ZAN.
```

8

9

```
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 4 TO ST-CALL
                GO TO PATERB.
        STRING "* VNOS IZD-KEY = " IZDLKIOWNKEY OF IZDLKI-002
                DELIMITED BY SIZE INTO CONFRMM.
        CALL "CONFRM" USING CONFRMM.
        DISPLAY "    STATUS: " DB-STAT.
DOBI-ODG.
        DISPLAY "NADALJUJES ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG
        ELSE
                IF NE
                        GO TO VNOS-IZDELKOV-EXIT.
        GO TO ZACNI.
VNOS-IZDELKOV-EXIT.
        GO TO MENU.
```

10

```
*****************************************************************
****    DODAJANJE NOVIH NAROCIL
*****************************************************************
DODAJ-NAROCILO SECTION.
ZACNI.
        DISPLAY "VNESI STEVILKO NAROCILA (5) : " NO ADVANCING
        ACCEPT KLJUC-NARO
        IF KLJUC-NARO = SPACE
                DISPLAY "PRAZEN KLJUC"
                GO TO ZACNI.
PONOVNI-VNOS-NAROCILA.
        MOVE GETG TO DB-FUNC
**
        PERFORM DB-NAROCI001
**
        IF STAR
                DISPLAY "  STEV LKA NAROCILA ZE OBSTAJA "
                GO TO DOBI-ODG-DALJE.
        IF OWNF
                GO TO PONOVNI-VNOS-KUPCA.
        IF NOT STAR
                MOVE 5 TO ST-CALL
                GO TO PATERB.
PONOVNI-VNOS-KUPCA.
        DISPLAY "VNESI SIFRO KUPCA       (6) : " NO ADVANCING
        ACCEPT NAROCISIFKUP OF NAROCI-002
        IF NAROCISIFKUP OF NAROCI-002 = SPACE
                DISPLAY "PRAZEN KLJUC"
                GO TO PONOVNI-VNOS-KUPCA.
```

11

```
        MOVE GETG TO DB-FUNC
        MOVE NAROCISIFKUP OF NAROCI-002 TO KLJUC-KUPC
**                                                              12
        PERFORM DB-KUPCI1001
**
        IF DB-STAT = "DI09"
                DISPLAY " KUPCA SE NI V BAZI"
                GO TO PONOVNI-VNOS-KUPCA.
        IF NOT STAR
                MOVE 6 TO ST-CALL
                GO TO FATERB.
        DISPLAY "VNESI DATUM NAROCILA    (6) : " NO ADVANCING
        ACCEPT NAROCIDATNAR OF NAROCI-001
        DISPLAY "VNESI ROK DOBAVE        (6) : " NO ADVANCING
        ACCEPT NAROCIROKDOB OF NAROCI-001.
        MOVE 0 TO ZAC-KAZALEC.
DOLOCI-POZICIJO-NAROCILA.
*Citamo brez zaklepanja do ustreznega narocila v setu
        MOVE GETG TO DB-FUNC
**                                                              13
        PERFORM DB-NAROCI004
**
        IF ENDP
                GO TO DODAJ-NAROCILO-KOT-ZADNJE.
        IF NOT STAR
                MOVE 7 TO ST-CALL
                GO TO FATERB.
        IF NAROCIROKDOB OF NAROCI-001 < NAROCIROKDOB OF NAROCI-004
                GO TO DOLOCI-POZICIJO-NAROCILA.
* Sele sedaj zaklenemo zapis da ne polnimo transakcijskega podrocja
* vec kot je nujno potrebno
HELD-ZAN.
* ker menjamo programski zapis moramo novemu prirediti kontekst
* prejsnjega zapisa - zato napolnimo zacetni kazalec
        MOVE TEK-KAZALEC TO ZAC-KAZALEC
        MOVE GETD TO DB-FUNC
**                                                              14
        PERFORM DB-NAROCI002
**
        IF HELD
                GO TO HELD-ZAN.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 8 TO ST-CALL
                GO TO FATERB.
        MOVE NAROCIDATNAR OF NAROCI-001 TO NAROCIDATNAR OF NAROCI-002
        MOVE NAROCIROKDOB OF NAROCI-001 TO NAROCIROKDOB OF NAROCI-002.
        MOVE KLJUC-KUPC TO NAROCISIFKUP OF NAROCI-002
        MOVE KLJUC-NARO TO NAROCIORNKEY OF NAROCI-002
        MOVE INSD TO DB-FUNC
**                                                              15
        PERFORM DB-NAROCI002
**
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
```

```
        IF HELD
                GO TO HELD-ZAN.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 9 TO ST-CALL
                GO TO FATERR.
        GO TO VNOS-SPECIFIKACIJE.
DODAJ-NAROCILO-KOT-ZADNJE.
        MOVE NAROCIDATNAR OF NAROCI-001 TO NAROCIDATNAR OF NAROCI-002
        MOVE NAROCIROKDOB OF NAROCI-001 TO NAROCIROKDOB OF NAROCI-002.
        MOVE KLJUC-KUPC TO NAROCISIFKUP OF NAROCI-002
        MOVE KLJUC-NARO TO NAROCIONNKEY OF NAROCI-002
* Insert na konec verige ne zahteva zaklepanja predhodnega zapisa
* zato ga lahko direktno dodamo
        MOVE INSG TO DB-FUNC                                          16
**
        PERFORM DB-NAROCI002
**
        IF HELD
                GO TO HELD-ZAN.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 10 TO ST-CALL
                GO TO FATERR.
VNOS-SPECIFIKACIJE.
* Tu vnasamo v zanki posamezne postavke narocila
        MOVE NAROCIONNKEY OF NAROCI-002 TO NARIZDSTVNAR OF NARIZD-002.
        DISPLAY "VNESI SIFRO IZDELKA          (12) : " NO ADVANCING
        ACCEPT NARIZDSIFIZD OF NARIZD-002
        IF NARIZDSIFIZD OF NARIZD-002 = SPACE
                DISPLAY " KEY"
                GO TO VNOS-SPECIFIKACIJE.
        MOVE GETG TO DB-FUNC
        MOVE NARIZDSIFIZD OF NARIZD-002 TO IZDLKIONNKEY OF IZDLKI-001 17
        MOVE NARIZDSIFIZD OF NARIZD-002 TO KLJUC-IZDLK
**
        PERFORM DB-IZDLKI001
**
        IF DB-STAT = "DI09"
                DISPLAY "IZDELKA SE NI V BAZI"
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 11 TO ST-CALL
                GO TO FATERR.
SPREJMI-STEV4.
        DISPLAY "VNESI NAROCENO KOLICINO : ____,___[9D"
                                                NO ADVANCING
        ACCEPT CIFRA
        MOVE CIFRA TO VHOD
        MOVE 5 TO ST-CELIH
        MOVE 3 TO ST-DECIMALK
        CALL "DOBIST" USING  VHOD ST-CELIH ST-DECIMALK IZHOD NAPACNO
        IF NAPACNO = "E"
                DISPLAY "NAPACEN VNOS - NENUMERICNO"
```

```
                GO TO SPREJMI-STEV4.
        MOVE IZHOD TO I-OSEN
        DISPLAY "[1A[36C" I-OSEN.
        MOVE IZHOD TO   NARIZDNARKOL OF NARIZD-002
        MOVE ZERO TO  NARIZDDOBKOL OF NARIZD-002.
HELD-ZAN1.
        MOVE INSG TO DB-FUNC
**                                                              18
        PERFORM DB-NARIZD002
**
        IF HELD
                GO TO HELD-ZAN1.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 12 TO ST-CALL
                GO TO PATERR.
DOBI-ODG.
* Dokler vnasamo narocene izdelke za isto narocilo ne bomo
* potrjevali vnosov, ker zelimo imeti narocilo v celoti
* vneseno ali pa da narocila ni.
        DISPLAY "NADALJUJES TO NAROCILO   ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG
        ELSE
                IF NE
                        GO TO DOBI-ODG-DALJE.
        GO TO VNOS-SPECIFIKACIJE.
DOBI-ODG-DALJE.
* na tem mestu smo zakljucili logicno transakcijo "NAROCILO"
* torej jo moramo sedaj se potrditi, da bo v bazi podatkov
* tudi dejansko ostala.
        STRING "* VNOS NAROCI-KEY = " NAROCIONNKEY OF NAROCI-002
                DELIMITED BY SIZE INTO CONFRNN.
        CALL "CONFRN" USING CONFRNN.                            19
        DISPLAY "    STATUS: " DB-STAT.
DOBI-ODG-DALJE-DN.
        DISPLAY "NADALJUJES DRUGO NAROCILO ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG-DALJE-DN
        ELSE
                IF NE
                        GO TO DODAJ-NAROCILO-EXIT.
        GO TO ZACNI.
DODAJ-NAROCILO-EXIT.
        GO TO MENU.
```

```
*****************************************************************
****    CITANJE NAROCIL
*****************************************************************
BERI SECTION.
ZACNI.
        DISPLAY "VNESI SIFRO IZDELKA (12) : " NO ADVANCING
        ACCEPT KLJUC-IZDLK.
        IF KLJUC-IZDLK = SPACE
                DISPLAY "PRAZEN KLJUC"
                GO TO ZACNI.
        MOVE KLJUC-IZDLK TO IZDLKIOWNKEY OF IZDLKI-001
        MOVE GETG TO DB-FUNC
**                                                              20
        PERFORM DB-IZDLKI001
**
        IF DB-STAT = "DI09"
                DISPLAY " IZDELEK NE OBSTAJA V BAZI "
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 13 TO ST-CALL
                GO TO FATERB.
        DISPLAY " ".
        DISPLAY " NAROCILA ZA IZDELEK " KLJUC-IZDLK.
        MOVE 0 TO ZAC-KAZALEC REFERENCA.
ZANKA.
        MOVE REFERENCA TO ZAC-KAZALEC
        MOVE GETG TO DB-FUNC
**                                                              21
        PERFORM DB-NARISD001
**
        IF ENDP
                DISPLAY " "
                DISPLAY " KONEC IZPISA "
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 14 TO ST-CALL
                GO TO FATERB.
        MOVE TEK-KAZALEC TO REFERENCA
        MOVE NARIZDSTVNAR OF NARISD-001 TO NAROCIOWNKEY OF NAROCI-001
                                                KLJUC-NARO
        MOVE GETG TO DB-FUNC
**                                                              22
        PERFORM DB-NAROCI001
**
        IF NOT STAR
                MOVE 15 TO ST-CALL
                GO TO FATERB.
        MOVE GETG TO DB-FUNC
        MOVE NAROCISIFKUP OF NAROCI-001 TO KUPCIIOWNKEY OF KUPCII-001
                                                KLJUC-KUPC      23
**
        PERFORM DB-KUPCII001
**
        IF NOT STAR
                MOVE 16 TO ST-CALL
                GO TO FATERB.
```

```
        DISPLAY " ".
        DISPLAY " ST. NAROCILA : "      NARIZDSTVNAR OF NARIZD-001.
        MOVE NARIZDNARKOL OF NARIZD-001 TO I-DESET
        DISPLAY "NAR. KOLICINA : "      I-DESET
        MOVE NARIZDDOBKOL OF NARIZD-001 TO I-DESET
        DISPLAY "DOB. KOLICINA : "      I-DESET
        DISPLAY "DAT. NAROCILA : "      NAROCIDATNAR OF NAROCI-001.
        DISPLAY "ROK DOBAVE    : "      NAROCIROKDOB OF NAROCI-001.
        DISPLAY "SIF. KUPCA    : "      NAROCISIFKUP OF NAROCI-001.
        DISPLAY "IME KUPCA     : "      KUPCIIMEKUP OF KUPCII-001.
        DISPLAY "NASLOV KUPCA  : "      KUPCINASLOV OF KUPCII-001.
        DISPLAY "TELEFON KUPCA : "      KUPCITELEFO OF KUPCII-001.
        DISPLAY "SIFRA DO KUP. : "      KUPCISIFDEO OF KUPCII-001.
        DISPLAY " ".
        MOVE IZDLKIKOLICI OF IZDLKI-001 TO I-DESET
        DISPLAY "ZALOGA MAT.   : "      I-DESET
        DISPLAY " ".
        GO TO ZANKA.
DOBI-ODG.
        DISPLAY "NADALJUJES ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG
        ELSE
                IF NE
                        GO TO BERI-EXIT.
        GO TO ZACNI.
BERI-EXIT.
        GO TO MENU.
```

```
*****************************************************************
****    POPRAVLJANJE NAROCIL
*****************************************************************
POPRAVI SECTION.
ZACNI.
        DISPLAY "VNESI STEVILKO NAROCILA : " NO ADVANCING
        ACCEPT KLJUC-NARO
        IF KLJUC-NARO = SPACE
        DISPLAY "PRAZEN KLJUC"
                GO TO ZACNI.
        MOVE KLJUC-NARO TO NAROCIONAKEY OF NAROCI-001
        MOVE GETG TO DB-FUNC
**                                                              ▶ 24
        PERFORM DB-NAROCI001
**
        IF DB-STAT = "D109"
                DISPLAY "  STEVILKA NAROCILA NE OBSTAJA "
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 17 TO ST-CALL
                GO TO FATERR.
PONOVNI-VNOS-IZDELKA-1.
        DISPLAY "VNESI SIFRO IZDELKA      : " NO ADVANCING
        ACCEPT KLJUC-IZDLK
        IF KLJUC-IZDLK = SPACE
                DISPLAY "PRAZEN KLJUC"
```

```
                GO TO PONOVNI-VNOS-IZDELKA-1.
        MOVE KLJUC-IZDLK TO IZDLKIOMKEY OF IZDLKI-001
        MOVE GETG TO DB-FUNC
**                                                              25
        PERFORM DB-IZDLKI001
**
        IF DB-STAT = "DI09"
                DISPLAY " IZDELEK NE OBSTAJA V BAZI "
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 18 TO ST-CALL
                GO TO FATERB.
        MOVE 0 TO ZAC-KAZALEC.
PREVERI-ALI-JE-V-NAROCILU.
        MOVE GETG TO DB-FUNC                                    26
**
        PERFORM DB-NARIZD005
**
        IF DB-STAT = "END."
                DISPLAY " IZDELEKA NI V TEM NAROCILU"
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 19 TO ST-CALL
                GO TO FATERB.
        IF NARIZDSIFIZD OF NARIZD-005 NOT = KLJUC-IZDLK
                GO TO PREVERI-ALI-JE-V-NAROCILU.
        MOVE NARIZD-005 TO STARI-BUFFER.
SPREJMI-STEV3.
    DISPLAY "VNESI NOVO KOLICINO : _____,___[11D" NO ADVANCING.
        ACCEPT CIFRA
        MOVE CIFRA TO VHOD
        MOVE 7 TO ST-CELIH
        MOVE 3 TO ST-DECIMALK
        CALL "DOBIST" USING  VHOD ST-CELIH ST-DECIMALK IZHOD NAPACNO
        IF NAPACNO = "E"
                DISPLAY "NAPACEN VNOS - NENUMERICNO"
                GO TO SPREJMI-STEV3.
        MOVE IZHOD TO I-DESET
        DISPLAY "[1A[36C" I-DESET.
        MOVE 0 TO ZAC-KAZALEC.
ISCI-ZAPIS.
* Iscemo po celotni verigi za razliko od shranjenega kazalca, ker
* so bili vmes vnosi operaterja, in ker je lahko nekdo drug v tem
* casu vnesel spremembe, ki spremene vrednost kazalca.
        MOVE GETG TO DB-FUNC
**                                                              27
        PERFORM DB-NARIZD005
**
        IF NOT STAR
                MOVE 20 TO ST-CALL
                GO TO FATERB.
        IF NARIZDSIFIZD OF NARIZD-005 NOT = KLJUC-IZDLK
                GO TO ISCI-ZAPIS.
* Sedaj lahko zapis zaklenemo in ugotovimo ali ga ni slucajno
* v casu vnosa preko ekrana popravil nekdo drug
        MOVE TEK-KAZALEC TO ZAC-KAZALEC
        MOVE GETD TO DB-FUNC
**                                                              28
        PERFORM DB-NARIZD003
**
        IF HELD
                GO TO ISCI-ZAPIS.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
```

```
        IF NOT STAR
                MOVE 21 TO ST-CALL
                GO TO FATERR.
        IF STARI-BUFFER NOT = NARIZD-003
                DISPLAY "DRUG UPORABNIK RAVNOKAR AŽURIRAL ZAPIS"
                DISPLAY "PROSIM PREVERI IN PONOVI OPERACIJO"
                GO TO ZACNI.
* POPRAVEK KOLICIN V BAZI
        ADD ODDANO TO NARIZDDOBKOL OF NARIZD-002
        MOVE TEK-NAZALEC TO ZAC-NAZALEC
        MOVE RNRG TO DB-FUNC
**
        PERFORM DB-NARIZD002                                          29
**
        IF HELD
                GO TO ISCI-ZAPIS.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 22 TO ST-CALL
                GO TO FATERR.
HELD-ZANKA.
        MOVE GETG TO DB-FUNC
        MOVE KLJUC-IZDLK TO IZDLKIONAKEY OF IZDLKI-002
**                                                                    30
        PERFORM DB-IZDLKI002
**
        IF HELD
                GO TO HELD-ZANKA.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 23 TO ST-CALL
                GO TO FATERR.
* Preverimo se ali je stanje izdelka nespremenjeno
* in v kolikor je moramo izvesti "CANCEL" ker smo ze
* spreminjali stanje v bazi
        IF IZDLKI-001 NOT = IZDLKI-002
                CALL "CANCEL"                                         31
                IF NOT STAR
                        GO TO FATERR
                ELSE
                        DISPLAY "NEKDO DRUG SPREMENIL STANJE"
                        DISPLAY "PREVERI IN PONOVI VNOS"
                        GO TO ZACNI.
        MOVE IZHOD TO ODDANO
* SPREMENI KOLICINO
        SUBTRACT ODDANO FROM IZDLKIKOLICI OF IZDLKI-002.
        MOVE RNRG TO DB-FUNC
**                                                                    32
        PERFORM DB-IZDLKI002
**
        IF HELD
                GO TO HELD-ZANKA.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
```

```
        IF NOT STAR
                MOVE 24 TO ST-CALL
                GO TO FATERR.
        STRING "* MODIFY NAROCI-KEY = " NAROCIONNKEY OF NAROCI-002
                DELIMITED BY SIZE INTO CONFRNN.
* Ker popravljamo vedno le eno vrstico v narocilu lahko
* takoj potrdimo opravljeni popravek
        CALL "CONFRN" USING CONFRNN.                                    33
        DISPLAY "    STATUS: " DB-STAT.
DOBI-ODG.
        DISPLAY "NADALJUJES ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG
        ELSE
                IF NE
                        GO TO POPRAVI-EXIT.
        GO TO ZACNI.
POPRAVI-EXIT.
        GO TO MENU.
```

```
*****************************************************************
****    BRISANJE NAROCIL
*****************************************************************
BRISI SECTION.
ZACNI.
    DISPLAY "VNESI STEVILKO NAROCILA ZA BRISANJE (5): " NO ADVANCING
        ACCEPT KLJUC-NARO.
        IF KLJUC-NARO = SPACE
                DISPLAY "PRAZEN KLJUC"
                GO TO ZACNI.
        MOVE KLJUC-NARO TO NAROCIONNKEY OF NAROCI-001
        MOVE GETG TO DB-FUNC                                            34
**
        PERFORM DB-NAROCI001
**
        IF DB-STAT = "DI09"
                DISPLAY "  STEVILKA NAROCILA NE OBSTAJA "
                GO TO DOBI-ODG.
        IF NOT STAR
                MOVE 25 TO ST-CALL
                GO TO FATERR.
        MOVE 0 TO ZAC-KAZALEC.
BRISI-SPECIFIKACIJE.
        MOVE GETG TO DB-FUNC                                            35
**
        PERFORM DB-NARIZ0004
**
        IF HELD
                GO TO BRISI-SPECIFIKACIJE.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF ENDP
                GO TO BRISI-NAROCILO.
```

```
        IF NOT STAR
                MOVE 26 TO ST-CALL
                GO TO FATERR.
        MOVE DELG TO DB-FUNC
**                                                              36
        PERFORM DB-NARIZD004
**
        IF HELD
                GO TO BRISI-SPECIFIKACIJE.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 27 TO ST-CALL
                GO TO BRISI-EXIT.
        GO TO BRISI-SPECIFIKACIJE.
BRISI-NAROCILO.
        MOVE KLJUC-NARO TO NAROCIONNKEY OF NAROCI-003
        MOVE GETG TO DB-FUNC
**                                                              37
        PERFORM DB-NAROCI003
**
        IF HELD
                GO TO BRISI-NAROCILO.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 28 TO ST-CALL
                GO TO FATERR.
        MOVE DELG TO DB-FUNC
**                                                              38
        PERFORM DB-NAROCI003
**
        IF HELD
                GO TO BRISI-NAROCILO.
        IF CNCL
                DISPLAY "NASTOPIL AVTOMATSKI CANCEL"
                GO TO ZACNI.
        IF TRLL
                MOVE "*****" TO DB-STAT
                DISPLAY "POZOR CIMPREJ ZAKLJUCI TRANSAKCIJO".
        IF NOT STAR
                MOVE 29 TO ST-CALL
                GO TO FATERR.
        STRING "* BRISI NAROCI-KEY = " NAROCIONNKEY OF NAROCI-002
                DELIMITED BY SIZE INTO CONFRMM.
        CALL "CONFRM" USING CONFRMM.                            39
        DISPLAY "    STATUS: " DB-STAT.
DOBI-ODG.
        DISPLAY "NADALJUJES ? D/N : " NO ADVANCING
        ACCEPT DA-NE
        IF NOT DA-NE-OK
                DISPLAY "ODGOVORI Z D/N "
                GO TO DOBI-ODG
        ELSE
                IF NE
                        GO TO BRISI-EXIT.
        GO TO ZACNI.
BRISI-EXIT.
        GO TO MENU.
```

```
*****************************************************************
****    CITANJE VSEH IZDELKOV
*****************************************************************
CITANJE-IZDELKOV SECTION.
ZACNI.
        MOVE 0 TO ZAC-KAZALEC.
GETP-ZANKA.
**
        PERFORM DB-IZDLKI001-GETP
**
        IF ENDP
                DISPLAY "END."
                GO TO CITANJE-IZDELKOV-EXIT.
        IF NOT STAR
                MOVE 30 TO ST-CALL
                GO TO FATERB.
        DISPLAY IZDLKIOWNKEY OF IZDLKI-001
        GO TO GETP-ZANKA.
CITANJE-IZDELKOV-EXIT.
        GO TO MENU.
```

40

```
*****************************************************************
****    CITANJE NAROCENIH IZDELKOV
*****************************************************************
CITANJE-NAR-IZD SECTION.
ZACNI.
        MOVE 0 TO ZAC-KAZALEC.
GETP-ZANKA-1.
**
        PERFORM DB-NARIZD001-GETP
**
        IF ENDP
                DISPLAY "END."
                GO TO CITANJE-NAR-IZD-EXIT.
        IF NOT STAR
                MOVE 31 TO ST-CALL
                GO TO FATERB.
        DISPLAY NARIZDSIFIZD OF NARIZD-001
        GO TO GETP-ZANKA-1.
CITANJE-NAR-IZD-EXIT.
        GO TO MENU.
*****************************************************************
*---------------------------------------
*       KONEC PROGRAMA DBDEMO
*---------------------------------------
```

41

INTERNO OBVESTILO

Za vse delavce IDA

Od Frenka

Za potrebe uradnih distribucij IDA programskih paketov sem
dne 14-aug-1985 odprl account in naslednje directory-je:

	DELTA/V	DELTA/M
1. USERNAME	DISTRIBUCIJA	Katerikoli priv.
PASSWORD	IDA	uporabnik
2. Osnovni (login) directory...	uma2:[D]	.....dr0:[1,160]
Slovenske verzije so na.....	uma2:[D.IDA]	.....
Menu IDA.....	uma2:[D.IDA.MENU]	.....dr0:[2,161]
Baza.....	uma2:[D.IDA.BAZA]	.....
Baza V 1.2.....	uma2:[D.IDA.BAZA.V12]	.....dr0:[2,162]
Cogen.....	uma2:[D.IDA.COGEN]	.....dr0:[2,164]
Ekran.....	uma2:[D.IDA.EKRAN]	.....dr0:[2,163] ←
Leksikon.....	uma2:[D.IDA.LEKSIKON]	.....dr0:[2,165]
Dbprint.....	uma2:[D.IDA.DBPRINT]	.....
Angleske pa na.....	uma2:[D.VIT]	.....
Menu Vitrage.....	uma2:[D.VIT.MENU]	.....dr0:[1,161]
Baza.....	uma2:[D.VIT.VBASE]	.....
Baza V 1.0.....	uma2:[D.VIT.VBASE.V10]	.....
Baza V 1.2.....	uma2:[D.VIT.VBASE.V12]	.....dr0:[1,162]
Cogen.....	uma2:[D.VIT.VGEN]	.....dr0:[1,164]
Ekran.....	uma2:[D.VIT.VFORM]	.....dr0:[1,163]
Leksikon.....	uma2:[D.VIT.LEKSIKON]	.....dr0:[1,165]
Dbprint.....	uma2:[D.VIT.DBPRINT]	.....

To so samo distribucije, ki so zive pri strankah. Vse zacasne (NEURADNE)
distribucije, zadrzite na svojih directory-jih dokler ne postanejo URADNE.