RNDr. Jan ZLÁMAL
NHKG n.p. Ostrava - Kunčice

PROGRAMOVACÍ JAZYK DL/I

1. Úvod

Programovací jazyk DL/I je částí systému Information Management System /dále jen IMS/. Tento softwarový systém vyvinula firma IBM,která také dodává tento systém uživatelům počítačů IBM.IMS usnadňuje uživateli zavádění a používání databází,přičemž operace na databázích je možno provádět online i v dávkách.

Programovací jazyk DL/I - název vznikl zkrácením anglických slov Data Language I - slouží pro zpracování vstupních a výstupních dat.Tento jazyk umožňuje uživateli IMS přizpůsobit tento systém požadavkům na vstupní a výstupní data,která vznikají při tvorbě nových aplikací.

Programovací jazyk DL/I není programovacím jazykem v tom smyslu,jak jej obvykle chápeme.Příkazy jazyka DL/I jsou příkazy pro vyvolání procedur,které používá aplikační program. V systému IMS je možno používat tři programovací jazyky při psaní aplikačních programů.Jsou to: PL/I,Cobol a Assembler. Pomocí příkazu CALL ve všech těchto jazycích se provádí příkazy jazyka DL/I.

Aplikační program má dvě různá spojení s jazykem DL/I:

a/ popis databáze - t.j. logická datová struktura databáze, jejíž definice je externí vzhledem k aplikačnímu programu

b/ společné symbolické spojení programů,které umožňuje programovacímu jazyku DL/I zpracovat požadavky na vstupní a výstupní operace v průběhu práce aplikačního programu.

Programovací jazyk DL/I může být použit pro

- vytváření a údržbu databází
- sjednocování a standartizaci aplikací
- zjednodušení údržby programů při změnách,které jsou způsobeny novými požadavky na databáze
- přístup a uložení dat při sekvenční,index-sekvenční a přímé orgamizaci dat.

2. Hlavní rysy programovacího jazyka DL/I

Použití programovacího jazyka DL/I způsobuje,že aplikační program je nezávislý na přístupové metodě,na fyzické organizaci dat v paměťovém mediu a na charakteristice paměťového media,na kterém jsou data uložena.Tato nezávislost je získána již výše uvedenými spojeními aplikačního programu s DL/I, t.j. externí definicí databáze vzhledem k aplikačnímu programu a symbolickým spojením programů.

Jazyk DL/I umožňuje vyloučit uložení redundantních dat a pomáhá při integraci či využívání společných souborů.V každé organizaci se vyskytují v konvenčních souborech data,která se opakují.Tato skutečnost často způsobuje,že zvláště u diskových paměťových jednotek dochází ke snižování kapacity paměťových jednotek.Organizace paměti a přístupové metody,které používá jazyk DL/I,umožňují integraci souborů při zachování minimální redundace dat.Navíc jazyk DL/I poskytuje tuto možnost: jestliže se ukáže,že všechna vstupní data nemohou být umístěna do jedné databáze,pak uživatel má možnost provést nové rozdělení dat do struktur a tyto struktury uložit do různých databází.

Důležitým rysem jazyka DL/I je citlivost na jisté údaje. Pomocí této vlastnosti lze chránit každou aplikaci,která používá společné databáze.Aplikace má přístup pouze k těm úda-

jůn databáze,které byly definovamé jako citlivé pro tuto aplikaci.Každá aplikace,která používá společnou databázi, může být citlivá k jisté kombinaci údajů,přičemž každá kombinace může být jedinečná.Možnosti zpracování dané aplikace se nemění,jestliže v databázi provedeme jakékoliv změny necitlivých údajů.Kromě toho aplikace může být ještě dále omezena pouze na jisté operace,které může provádět se svými citlivými údaji.

Základní pojmy jazyka DL/I:

a/ Segment - je to část dat pevné délky,která obsahuje jedno nebo více logicky spojených datových polí.Segment je základní datový element,který je zpracováván a předáván mezi aplikačním programem a DL/I.Segment je také základní element pro stanovení citlivosti.

b/ Rekord /věta/ logické databáze - je to skupina hierarchicky spojených segmentů o pevné délce,která obsahuje jeden nebo více typů segmentů.Každý typ segmentu má svou délku a formát.Z hlediska aplikačního programu je rekord logické databáze vždy hierarchická stromová struktura segmentů.

c/ Logická databáze - je to množina rekordů logické databáze,která používá organizaci dat jazyka DL/I.Pro přístup do databáze je možno použít jednu nebo více přístupových metod jazyka DL/I.Databázi tvoří obyčejně jeden nebo více klasických souborů.

Programovací jazyk DL/I používá dvě základní organizace dat: hierarchickou sekvenční organizaci a hierarchickou přímou organizaci.Pro přístup do těchto organizací dat se používají čtyři přístupové metody:

hierarchická sekvenční přístupová metoda /HSAM/
hierarchická index-sekvenční přístupová metoda /HISAM/
hierarchická přímá přístupová metoda /HDAM/
hierarchická indexně přímá přístupová metoda /HIDAM/.

Spojení aplikačního programu a uvedenými organizacemi a

přístupovými metodami je pouze symbolické. Aplikační program je nezávislý na použité organizaci a přístupové metodě.

Aby tato nezávislost byla umožněna,musí uživatel ještě před používáním databáze definovat

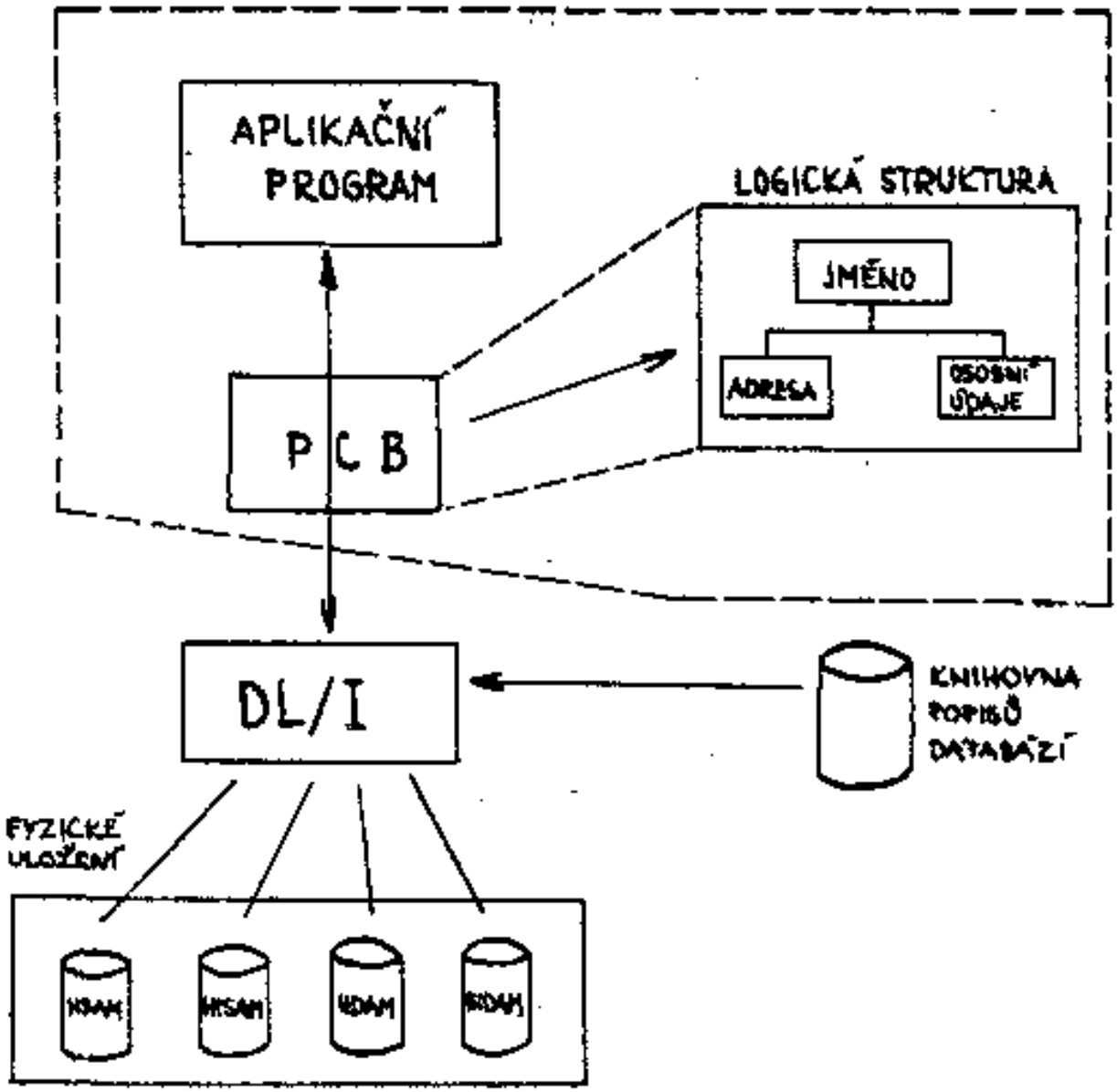
a/ segmenty v rekordu logické databáze,ke kterým má být program citlivý
b/ strukturu logického rekordu databáze
c/ organizaci databáze a přístupové metody.

Aplikační programy,které pracují za řízení IMS,operují s logickými strukturami dat.Logickou strukturou se rozumí způsob,jak aplikační program "vidí" data.Logická struktura je vždy hierarchická struktura segmentů.Programy,které spracovávají logické struktury dat,mohou být nezávislé na fyzické struktuře dat.Fyzickou strukturou se rozumí způsob, jak jsou data uložena na magnetické pásce nebo na zařízení s přímým přístupem.Aplikační program nikdy nepracuje přímo s fyzickou strukturou dat.

Pravidla jazyka DL/I pro práci s databázemi:

Databáze může mít max. 255 typů segmentů.
Databáze může mít max. 15 hierarchických úrovní.
Rekord databáze může mít pouze jeden kořenový segment /root segment/.
Rodičovský segment může mít n závislých segmentů.
Každý segment může mít v rekordu databáze n výskytů.
Databáze se skládá z 1 až n rekordů databáze.

Obrázek 1 ukazuje "pohled" aplikačního programu pomocí komunikačního bloku PCB /Program Communication Block/ na logickou datovou strukturu.Obrázek dále ukazuje spojení jazyka DL/I s fyzickým uložením dat pomocí různých přístupových metod.Přitom je možné,že každý segment definovaný v logické struktuře je fyzicky uložen v jiné databázi.Tímto způsobem získává IMS nezávislost na datech.



Obr. 1

3. Spojení aplikačního programu s IMS

V systému IMS může být aplikační program nastartován dvěma způsoby: buď přijde zpráva z terminálu, která má jako první položku tranzakční kód a tento tranzakční kód vyvolá příslušný aplikační program /to je případ režimu on-line/ nebo aplikační program nahrajeme do systému pomocí JCL /t.j. jazyk pro řízení prací/ a pak tento program pracuje jako normální program v dávkovém zpracování. Ovšem i

tento program má přístup do databází a front systému IMS.

Když aplikační program pracuje s databází - nezáleží na tom, jakým způsobem byl nastertován - musí obsahovat masku pro tuto databázi a pomocí této masky "vidí" logickou strukturu dat. V programovacím jazyce PL/I vypadá tato maska takto:

```
DECLARE 1 PCBNAME,
          2 DBDNAME CHAR(8),
          2 SEGLEVEL CHAR(2),
          2 STATUSCODE CHAR(2),
          2 PROCOPTIONS CHAR(4),
          2 RESERVEDLI FIXED BIN(31,0),
          2 SEGNAMEFB CHAR(8),
          2 LENGTHFBKEY FIXED BIN(31,0),
          2 NUMBSENSSEG FIXED BIN(31,0),
          2 KEYFBAREA CHAR(45);
```

V okamžiku zpracování vkládá jazyk DL/I do této masky údaje, které říkají aplikačnímu programu, jaký byl výsledek použitého příkazu CALL /dále jen CALL/. Stručně uvedu význam jednotlivých položek struktury:

1. Jméno PCB - toto jméno se používá v příkazech jazyka DL/I a nepředstavuje žádnou položku.
2. Jméno popisu databáze - udává jméno členu ze speciální knihovny, který popisuje databázi. Každá databáze má v této knihovně jeden člen. Jméno je dlouhé 8 znaků.
3. Indikátor úrovně segmentu - DL/I dává do této položky číslo úrovně posledního nalezeného segmentu, který vyhovoval podmínkám CALLu. Tato položka je 2 znaky dlouhá a obsahuje numerické znaky.
4. Kód stavu DL/I - kód stavu, který indikuje s jakým výsledkem byl proveden DL/I CALL, je umístěn do tohoto pole a zůstává zde tak dlouho, dokud další DL/I CALL nepoužije toto PCB. Tato položka je 2 znaky dlouhá. Jestliže CALL byl úspěšný, tato položka je prázdná /blank/ nebo obsahuje pouze kód, který znamená jistá upozornění.

5. Volby zpracování DL/I - tato položka obsahuje kód,který říká jazyku DL/I,které typy zpracování mohou být použity programem při práci s databází.Položka je 4 znaky dlouhá.Nejdůležitější volby pro zpracování jsou:

   G - GET/přečti/
   I - INSERT/vlož/
   R - REPLACE/nahraď/
   D - DELETE/vypusť/
   A - ALL/zahrnuje všechny výše uvedené funkce/
   E - EXCLUSIVE/ výlučné použití databáze nebo segmentu,t.j. v době,kdy program pracuje s databází,žádný jiný program nemá do ní přístup/.

6. Rezervovaná položka pro DL/I - DL/I používá tuto položku pro své vlastní spojení s aplikačním programem.

7. Jméno segmentu - DL/I do této položky dává jméno posledního segmentu,který vyhovoval podmínkám CALLu.Jestliže požadovaný segment byl nalezen,pak jeho jméno je umístěno zde.Jestliže nebyl nalezen,pak zde bude uvedeno jméno posledního segmentu na hierarchické cestě,který vyhovoval podmínkám CALLu.Toto pole je dlouhé 8 znaků.

8. Délka klíče - tato položka udává délku klíče segmentu popsaného v bodě 7.

9. Počet citlivých segmentů - tato položka udává počet segmentů databáze,ke kterým je aplikační program citlivý.

10. Klíč segmentu - do této položky dává DL/I zřetězený klíč segmentu z bodu 7. Jestliže požadovaný segment byl nalezen,pak do této položky je uložen zřetězený klíč požadovaného segmentu a klíč každého segmentu hierarchické cesty směrem k požadovanému segmentu. Klíče jsou umístěny zleva doprava a to počínaje kořenovým segmentem.Jestliže požadovaný segment nebyl na-

lezen,pak je uložen zřetězený klíč - podobně jako v předcházejícím případě - posledního segmentu,který vyhovoval podmínkám CALLu.

Aplikační program obsahuje pouze masku pro PCB.Skutečné PCB je uloženo mimo aplikační program.Všechna PCB,která jsou používána aplikačním programem,jsou obsažena ve specifickém bloku PSB - Program Specification Block.Toto PSB je přidruženo k aplikačnímu programu.Vytvoření PSB provádí firemní program a ukládá nové PSB do knihovny PSB.

Když řídicí modul DMS předá řízení aplikačnímu programu,pak první příkaz programu musí specifikovat všechna PCB,se kterými program pracuje.Příkaz v jazyce PL/I vypadá takto:

```
PROGRAM:PROCEDURE(PCBNAME1,...,PCBNAMEn)OPTIONS(MAIN);
```

4. Příkaz CALL při operacích na databázi

Jazyk DL/I může spolupracovat pomocí CALLu s jazyky Cobol,Assembler a PL/I.Forma příkazu CALL v PL/I je následující:

```
CALL PLITDLI(param,funkce,jméno PCB,oblast,SSA1,...,SSAn);
```

kde param je jméno slova,které obsahuje počet parametrů tohoto CALLu.Máme-li jen jedno SSA,pak v tomto slově musí být hodnota 5,

funkce udává jméno 4-znakové položky,která popisuje požadovanou vstupní nebo výstupní operaci.Tyto funkce budou popsány dále,

jméno PCB je třetím parametrem tohoto CALLu.Je to jméno tabulky,která přiřazuje aplikačnímu programu jistou logickou strukturu dat.Data,ke kterým bude mít aplikační program přístup,jsou definována v PCB,

oblast je čtvrtým parametrem CALLu.Udává pracovní oblast v aplikačním programu,do které DL/I uloží požadovaný segment nebo ze které DL/I vezme vložený segment;

tato oblast musí být tak velká,aby do ní mohl být vložen i nejdelší segment,který bude zpracován. Segmenty do této oblasti jsou vždy ukládány zleva. V jazyce PL/I musí být tato oblast definována jako CHARACTER,

SSA /zkratka z anglického Segment Search Argument/ je pátým parametrem CALLu a slouží pro vyhledávání segmentů.Tento parametr je popsán dále.

Když aplikační program požaduje na DL/I aby provedl jistou funkci,často musí určit jméno segmentu a jména všech rodičovských segmentů,které na hierarchické cestě vedou k požadovanému segmentu.Tato jména se přímo neobjevují v příkaze CALL.Místo SSA je uděno jméno pole,které obsahuje daný parametr pro vyhledávání segmentů.Toto pole musí mít určitý formát.
Omezení: V CALLu může být uvedeno maximálně 15 parametrů SSA.

Parametr SSA se může skládat z těchto prvků:

jména segmentu
povelového kódu a
kvalifikačního příkazu.

Povelové kódy jsou volitelné a specifikují funkci,která se týká CALLu nebo kvalifikace segmentu nebo ustavení rodičovského segmentu.

Kvalifikační příkaz je také volitelný a obsahuje informaci,kterou používá DL/I pro test klíče segmentu nebo pro test některé položky segmentu.Je-li test splněn,pak příslušný segment je vyvolán.Pomocí této techniky DL/I prohledává segmenty databáze a aplikační program pak zpracovává pouze ty segmenty,které potřebuje.

Každý kvalifikační příkaz se skládá se tří částí: z názvu pole,relačního operátoru a srovnávací hodnoty.Je možné provádět také booleovské operace s kvalifikačními příkazy, t.j. je možné je spojovat pomocí logického "a" a "nebo". Celý příkaz pro kvalifikaci segmentu musí být uzavřen v

závorkách.V jednom SSA může být maximálně 8 kvalifikačních příkazů spojených booleovskými operacemi.

Stručně uvedu význam některých povelových kódů a kvalifikačních příkazů.

Povelové kódy jsou volitelné a dělí se do třech kategorií:

1. kódy,které modifikují 2.parametr CALLu,totiž funkci:
   L - najdi poslední výskyt segmentu žádaného typu pod rodičovským segmentem,který vyhovuje kvalifikačnímu příkazu.Jde-li o nekvalifikovaný příkaz,najdi poslední výskyt segmentu žádaného typu pod rodičovským segmentem
   D - pro CALLy,které vyvolávají segment,ulož segment, vyhovující podmínkám CALLu do oblasti.Tento kód umožňuje vyvolání více segmentů podle hierarchické cesty v jednom CALLu.Tento typ CALLu se nazývá CALL sledující hierarchickou cestu.Segmenty z hierarchické cesty jsou zřetězeny a umístěny do oblasti aplikačního programu.

2. kódy,které provádějí kvalifikaci segmentu:
   C - hodnota uvedená v závorkách představuje zřetězený klíč jmenovaného segmentu
   U - CALL musí použít segment,na kterém bylo provedeno poslední nastavení v databázi.

3. kódy pro nastavení rodičovského segmentu
   P - na této úrovni vytvoř ze segmentu rodičovský segment.Rodičovství bude zrušeno,jestliže pak použijeme CALL s funkcí GU nebo GN - viz dále.

Zásady pro použití kvalifikačních příkazů:

Pro každý segment je možno specifikovat až 8 kvalifikačních příkazů.Kvalifikační příkazy spojené logickým "a" jsou považovány za množinu kvalifikačních příkazů.Logické "nebo" mezi dvěma kvalifikačními příkazy začíná novou množinu kvalifikačních příkazů.Množina může obsahovat jeden

nebo více příkazů.Aby bylo vyhověno SSA,segment musí vyhovovat alespoň jedné množině kvalifikačních příkazů.

V kvalifikačních příkazech je dovoleno použít tyto operátory:

=	roven
>=	větší nebo roven
<=	menší nebo roven
>	větší
<	menší
¬=	není roven

Příklad kvalifikačního příkazu:

SEGMENTA(FIELDAAA)099 & FIELDAAA<201|FIELDBBB=0

/ & představuje logické "a", | představuje logické "nebo"/.

Podmínkám SSA budou vyhovovat segmenty SEGMENTA,jejichž položka FIELDAAA je v rozmezí od 100 do 200 nebo jejichž položka FIELDBBB je rovna nule.

Funkce pro vstupní a výstupní operace příkazu CALL

Funkce pro vstupní a výstupní operace je druhým parametrem CALLu.Funkce,které můžeme použít,představují služby,které poskytuje jazyk DL/I uživateli.

Pro informaci uvádím formu CALLů i pro zbývající jazyky:

Cobol:

```
CALL 'CBLTDLI' USING [parem,]funkce,jméno PCB,
     oblast,SSA....
```

Assembler:

```
CALL  ASMTDLI,(parem,funkce,(R),...) ,
```

kde R je registr,který udává adresu PCB.
Význam ostatních parametrů je stejný jako u CALLu v jazyce PL/I.

Funkce pro vstupní a výstupní operace rozdělujeme do

tří skupin: vyvolání segmentu
vložení segmentu
vypuštění a nahrazení segmentu.

Vyvolání segmentu:

1. GET UNIQUE /GU/ - vyvolání segmentu bez ohledu na stávající pozici v databázi.Tato funkce se používá,když vyvolávéme segmenty,které jsou umístěny v databázi na různých místech anebo pro nastavení pozice v databázi s následujícím sekvenčním zpracováním.Tato funkce se dá také použít pro čtení zpráv z front systému IMS.

2. GET NEXT /GN/ - vyvolání segmentu nebo segmentů z hierarchické cesty od nastavené pozice v databázi. /To je také hlavní rozdíl mezi touto a předcházející funkcí.GET UNIQUE hledá kořenový segment,který vyhovuje podmínkám CALLu./ Jestliže není uveden parametr SSA v CALLu,pak bude vyvolán segment,který je hned za nastavenou pozicí v databázi.Pokud budeme tento příkaz opakovat,pak tímto způsobem lze přečíst sekvenčně všechny segmenty v databázi.
Použijeme-li pro tuto funkci CALL s parametrem SSA,pak dostaneme segment žádaného typu,který bude nalezen první za nastavenou pozicí v databázi.Budeme-li tento příkaz opakovat,dostaneme všechny segmenty databáze žádaného typu.Tato funkce se dá také použít pro čtení zpráv z front systému IMS.

3. GET NEXT WITHIN PARENT /GNP/ - vyvolání segmentu pouze nižší úrovně vzhledem k podřízenému segmentu.

Chceme-li změnit obsah segmentu v databázi pomocí funkce vypuštění a nahrazení,pak program musí segment nejdříve získat.Pak změníme jeho obsah a pomocí DL/I jej vložíme zpět do databáze.Má-li být segment změněn,potom tato skutečnost musí být sdělena DL/I již v okamžiku,kdy je segment získán.Děje se to pomocí funkce GET HOLD. Funkční

kódy jsou stejné jako u právě popsaných funkcí bodů 1.-3., jen za písmeno G je vloženo H.Máme tedy k dispozici další tři funkce,t.j. GHU,GHN,GHNP.Tyto funkce se řídí stejnými pravidly jako první tři funkce.V segmentu lze měnit kterékoliv pole kromě klíčového pole.

Vložení segmentu:

INSERT /ISRT/ - tato funkce se používá pro dva účely. Pro úvodní nahrání segmentů při vytváření databáze a při vložení již existujícího typu segmentu,má-li databáze orgenizaci HISAM,HDAM nebo HIDAM.Pole volby zpracování masky PCB říká,jde-li o vytvoření databáze nebo o přidání segmentu do již existující databáze. Formát příkazu CALL je stejný pro oba příkazy.

Vypuštění a nahrazení segmentu:

1. DELETE /DLET/ - vypuštění segmentu z databáze.Před tímto CALLem musí být proveden příkaz typu GET HOLD.Mezi příkazem GET HOLD a příkazem DELETE nesmí být proveden žádný jiný příkaz pro uvedenou databázi.Vypuštěný segment nemusí být fyzicky zrušen,může být označen jako neexistující segment.Závisí to na typu orgenizace databáze.
2. REPLACE /REPL/ - nahrazení segmentu v databázi. Před tímto CALLem musí být proveden příkaz typu GET HOLD. Mezi příkazem GET HOLD a příkazem REPLACE nesmí být proveden žádný jiný příkaz pro databázi,které se příkaz REPLACE týká.Při změně segmentu není povoleno změnit klíč segmentu.Pokud modifikujeme klíč segmentu, pak tento CALL jazyk DL/I neprovede.

5. Příkaz CALL při práci s terminály

Při práci s databází byl aplikační program spojen s databází pomocí masky PCB.Při práci s terminály existuje po-

dobné spojení.Aplikační program "vidí" zařízení dálkového přenosu dat jako logický terminál pomocí masky LTPCB.

Princip logického terminálu vytváří nezávislost aplikačního programu na fyzickém zařízení.Logický terminál je jméno,které je přidělenо fyzickému terminálu.Fyzickému terminálu může být přidělenо více logických terminálů. Jméno logického terminálu je použito při vytváření a přenosu zpráv.Aplikační program nikdy nepracuje s adresou fyzického terminálu.Jestliže fyzický terminál je v poruše,pak logický terminál,který byl přidělen tomuto fyzickému terminálu,lze přiřadit jinému fyzickému terminálu.

Maska logického terminálu vypadá v jazyce PL/I takto:

```
DECLARE  1 INPUTPCB,
           2 IOTERMINAL CHAR(8),
           2 IORESERVE BIT(16),
           2 IOSTATUS CHAR(2),
           2 INPREFIX,
             3 PREDATE FIXED DECIMAL(7,0),
             3 PRETIME FIXED DECIMAL(7,0),
             3 PREMSGCOUNT FIXED BINARY(31,0),
           2 MODNAME CHAR(8);
```

Význam jednotlivých položek této masky:

1. Jméno PCB /Program Communication Block/ logického terminálu - užívá se v příkazech jazyka DL/I.
2. Jméno logického terminálu - může být maximálně 8 znaků dlouhé.
3. Rezervovaná oblast jazyka DL/I - obsahuje speciální informace pro DL/I.
4. Kód stavu - výsledek CALLu,který zpracoval zprávu,je uložen do této položky.Jestliže byl CALL úspěšný,bude tato položka prázdná nebo bude obsahovat kód,který znamená jisté upozornění.
5. Vstupní prefix - je požadován jen pro PCB vstupního

logického terminálu.Délka této položky je 12 byte a její struktura je tato:

4 byte - datum vstupní zprávy ve tvaru YYDDD /např. 75032/,kde DDD je pořadové číslo dne v rámci roku,YY je běžný rok,

4 byte - čas,kdy vstupní zpráva vstoupila do systému,ve tvaru HHMMSS.S, kde HH - hodina,MM - minuta,SS.S - vteřina.desetina vteřiny,

2 byte - rezervováno pro jazyk DL/I

2 byte - pořadové číslo zprávy.

6. Jméno popisu výstupní zprávy - používá se jen v případě,když aplikační program komunikuje s terminálem,který používá služební /firemní/ program pro formátování zpráv.Tímto terminálem je v systému IMS displej.

Při zpracování segmentu vstupní či výstupní zprávy je možno použít těchto funkcí: GU,GN,ISRT.Tyto funkce byly popsány v kapitole 4.

Formát příkazu CALL pro zpracování vstupní a výstupní zprávy je jednodušší,protože zde nepracujeme s žádnou hierarchickou strukturou.Parametr SSA se nesmí použít. Formát tohoto příkazu CALL vypadá v jazyce PL/I takto:

```
CALL PLITDLI(param,funkce,jméno LTPCB,oblast);.
```

Význam jednotlivých parametrů je stejný jako v CALLu pro operace na databázi.

Funkce GU,GN se používají pro čtení vstupních zpráv. Funkce ISRT se používá pro vyslání zprávy na terminál nebo k vyslání zprávy do jiného aplikačního programu.

IMS tedy pracuje se 3 druhy zpráv.Jsou to vstupní zprávy ,výstupní zprávy a zprávy,které jdou z programu do programu.

6. Závěr

Cílem tohoto příspěvku bylo podat stručnou informaci o základních funkcích jazyka DL/I a naznačit možnosti použití tohoto jazyka v systému IMS. Podrobnou informaci o tomto jazyce lze nalézt ve firemních publikacích firmy IBM o systému IMS. Nezmínil jsem se záměrně o celé koncepci systému IMS,protože to nebylo tematem tohoto příspěvku.Domnívám se však,že dobrá znalost systému IMS a jeho možností je nutná pro aplikačního programátora, který používá jazyk DL/I.

Jazyk DL/I mohou používat jen uživatelé IMS a proto není zatím příliš rozšířen. Soudím však,že ostatní jazyky či procedury,které budou pracovat s databázemi a bankami dat,budou používat podobné principy a zásady jako programovací jazyk DL/I.