Okč

KATEDRA SOFTWAROVÉHO INŽENÝRSTVÍ

vypisuje seminář

M technologie

Vedoucí : Pavel Bezstarosti
Rozsah : 0/2 Z

M technologie (dříve MUMPS) je komplexní ANSI a ISO standardizovaný systém založený na vlastním datovém modelu dovolující efektivně vyvíjet vysoce funkční aplikace. M technologie obsahuje mimo jiné procedurální programovací jazyk, víceuživatelský a vícedimenzionální databázový subsystém a řadu doplňujících standardů a propojení.

M jazyk

Flexibilní a výkonný programovací jazyk navržený zejména pro manipulaci s řetězci a databázemi. Je velice jednoduchý, kompaktní, snadný k naučení s jasnými syntaktickými i sémantickými pravidly.

M databáze

Zatímco relační systémy pracují s dvourozměrnými relacemi, jsou datové struktury M vícedimenzionální. To činí M velice flexibilní při popisu reálného světa. M používá jediný datový typ - řetězec s proměnlivou délkou. Veškerá data v M jsou uložena v proměnných, které mohou být libovolně indexované, nemusí se definovat a systém pro ně přiděluje zdroje dynamicky. Výhodou M je stejná logická struktura proměnných lokálních (patřících samotnému procesu) a globálních (sdílených všemi procesy).

Program semináře

1. Úvod, historie a principy M
2. Datové struktury M, základní příkazy, proměnné, funkce
3. Dotazy v M
4. Návrhy síťových aplikací
5. Návrhy vícejazyčných aplikací
6. Návrhy temporálních datových struktur
7. Datový slovník a propojení s SQL
8. Propojení s vizuálními nástroji Visual M, Delphi

Podmínkou pro získání zápočtu bude pravidelná účast a vypracování jednoduché příkladu. Úmluva a první cvičení proběhne ve čtvrtek 20.2.1997 v 18:00 v posluchárně S7 v budově na Malé Straně. Připomínky k termínu nebo dotazy volejte telefonicky a případně nechte vzkaz.

Pavel Bezstarosti
E-mail : pbez0262@comenius.mff.cuni.cz
telefon : 02/2213 6512 (IDEA s.r.o.)

M technologie

Pavel Bezstarosti

Motto : M technologie je podobná hře v šachy. Je velice snadné se jí naučit, ale i po letech intenzivní práce člověk stále nachází nové a nové fascinující možnosti.

1. Úvod

Hlavním cílem tohoto příspěvku je představit M a ukázat, že tato technologie nabízí originální systém manipulace s daty, založený na logických a přehledných stromových strukturách. Ty jsou podpořeny jednoduchým, zároveň však velmi silným programovacím jazykem, řadou navazujících standardů, protokolů a možných propojení.

M technologie (dříve MUMPS) je komplexní systém dovolující velice efektivně vyvíjet vysoce funkční aplikace. Vývojáři přitom těží zejména z vysoké produktivity týmů, přenositelnosti aplikací a jejich vysoké výkonnosti.

V tomto článku se budeme věnovat následujícím tématům :

- Datový model M technologie
- Vývoj aplikací a hlavní rysy programovacího jazyka M
- Hlavní výhody a nevýhody M technologie

2. Datový model M technologie

M technologie je SŘDB založený na vlastním datovém modelu práce se stromovou databází, jejíž struktura se velice blíží síťové struktuře reálných ekonomických, výrobních či biologických systémů. Tyto síťové struktury dovolují naprostou volnost v návrhu dat, přičemž uživatel může lehce využívat nesporných výhod všech existujících databázových modelů ať už se jedná o hierarchické, síťové, relační či objektové přístupy. Celá M databáze je schopna poskytovat běžné relační služby pomocí standardního SQL2, datové struktury jsou velice podobné objektům a ze struktury databáze je možné vypozorovat i prvky síťového modelu. Navigace po takovéto struktuře je v přirozeném směru mnohem efektivnější než u relačního modelu.

Veškerá data M technologie jsou uložena v proměnných, které se dělí na lokální a globální. Globální proměnné (nazývané též globály) obsahují persistentní data, jsou uloženy na disku, jsou přístupné všem uživatelům a jsou jimi sdíleny. Obsah lokálních proměnných je uložen v paměti, tyto proměnné jsou přístupné pouze jednomu procesu (uživateli) a po skončení běhu programu se ztrácejí. Struktura lokálních a globálních proměnných je naprosto stejná.

Zkusíme-li u pracovníka evidovat i všechny kursy, které kdy absolvoval, můžeme dostat následující strukturu :

```
^ZAM
├── (1)="Josef\Novák\12.4.1965\6500"
│   └── ,"KURS")
│       └── ,"SVAREC")="Svářečský kurs\15.3.1993\15.3.1996"
├── (2)="Petr\Macháček\18.12.1958\7600"
│   └── ,"KURS")
│       ├── ,1)="Kurs angličtiny\12.10.1994"
│       └── ,2)="Kurs první pomoci\10.1.1995\10.1.1998"
└── (40)="Jana\Boušková\8.3.1973\5000"
```

Je třeba ještě dodat, že každý kurs je identifikován libovolným identifikátorem (jednoznačným v rámci jednoho člověka) a záznam obsahuje popis kursu, datum absolvování a eventuelně i dobu platnosti osvědčení z tohoto kursu. V tomto příkladu není zohledněna možnost efektivního zadání téhož kursu více zaměstnancům. Ta by spočívala ve vytvoření pomocného číselníku kursů a uvedením odkazu na něj u zaměstnance.

Na tomto příkladu vidíme další typické vlastnosti M :

1. Údaje o kursu jsou vloženy přímo do struktury seznamu zaměstnanců, což dosti připomíná objektové struktury. V klasickém relačním modelu bychom museli vytvořit novou tabulku kursů s cizím klíčem osobní číslo (což je samozřejmě možné i v M). Zde použitá konstrukce je vlastně zárodkem navigace a jednoduchou realizací operace spojení v relačním modelu (JOIN v SQL).

2. Celá M technologie je založena na zpracování řetězců proměnlivé délky. Není zde kladen žádný důraz na typovost údajů, což dovoluje označovat jednou identifikátor kursu znakově, podruhé číselně. Stejně tak je výhodné používat M v případě evidence dat, která obsahují často prázdné hodnoty. V tomto okamžiku M obsadí pouze tolik místa na disku, kolik záznam skutečně vyžaduje. Pokud má tabulka často hodně prázdných atributů (nedefinovaných, NULL), pak se takováto data neumisťují příliš dobře do struktur relačního modelu. Řídké hierarchické struktury polí používané v M jsou naopak pro takováto data přirozená a efektivní.

3. V tomto návrhu, který představuje vlastně strom se třemi úrovněmi, se také objevil uzel "KURS", který na rozdíl od uzlů na první a třetí úrovni nenese žádná data. Jedná se o takzvaný fiktivní uzel. Tyto fiktivní uzly se využívají pro přehledné rozčlenění datové struktury.

4. Z příkladu je též pěkně vidět maximální arita některých atributů. K údaji o jménu jednoho zaměstnance se dostaneme pouze po jedné větvi stromu, zatímco k popisu jeho kursů po

několika. Z toho plyne, že zaměstnanec může mít maximálně jedno jméno a libovolně mnoho kursů, které jsou vlastně zavěšeny v jeho podstromu. Této názornosti lze s úspěchem využít v době analýzy a návrhu systému, kdy i běžný uživatel je schopen diskutovat a pracovat nad těmito schématy. Zde je vhodné upozornit na podobnost s asociativním datovým modelem, který používá podobné mechanismy pro zápis výsledků datové analýzy.

Dalším rozšířením našeho příkladu může být evidence jednotlivých

spojení na daného zaměstnance, ať už se jedná o spojení telefonické, faxové, E-mailové nebo o spojení adresou nebo bankovním účtem :

```
^ZAM
└──── (2)="Petr\Macháček\18.12.1958\7600"
        ├──── ,"KURS")
        │       ├──── ,1)="Kurs angličtiny\12.10.1994"
        │       └──── ,2)="Kurs první pomoci\10.1.1995\10.1.1998"
        └──── ,"SPOJ")
                ├──── ,1)="TEL\02-4056849\domů"
                ├──── ,2)="TEL\02-3453345\do práce od 7:00 do 15:00"
                ├──── ,3)="ADR\Václavské náměstí 1\Praha 1\111 50"
                └──── ,4)="MAIL\machacek@idea.cz"
```

Zde se dostáváme do oblasti řešitelné v relačním modelu množstvím pomocných relací eventuelně ztrátou 1.NF. Projevuje se tu totiž variantnost záznamu, v tomto příkladu v závislosti na prvním atributu. Tato variantnost není ovšem v praxi příliš výjimečná. Vezměme si například evidenci diagnóz u pacienta v nemocničním informačním systému. V principu každé onemocnění vyžaduje evidenci výsledků jiných laboratorních testů, potřebu mnoha různých atributů v různých tabulkách. Obdobným případem v běžném ekonomickém SW je existence různých typů řádků na faktuře, které mohou být například za zboží, za práci, poznámkové, součtové, daňové a podobně. Zde se ukazuje výhoda a praktičnost možnosti navrhovat datová schémata bez nutnosti jejich deklarace a s možností pozdějších úprav. Vůbec úpravy databázového schématu za běhu aplikace jsou velkou předností M technologie, která nejenomže dovoluje přidat další uzel (resp. změnit význam a interpretaci některého již existujícího), ale nevyžaduje přitom žádnou reorganizaci existujících dat a s tím spojené odstavení systému.

Pokud se v našem příkladu se zaměstnanci rozhodneme navíc ještě evidovat též seznam dětí zaměstnance, stačí pouze upravit potřebné obslužné programy a případnou reorganizaci dat si provede databázový systém sám až v okamžiku vložení prvního záznamu o dítěti do databáze.

Posledním rozšířením našeho minipříkladu může být evidence úkolů, jejich dílčích termínů a evidence zaměstnanců, kteří na nich pracují :

```
^UKOLY
└──── (95001)="Zpracování projektu XYZ\Zákazník ABC"
        ├──── ,"PRACUJI")
        │       ├── ,2)="šéf projektu"
        │       └── ,40)="asistentka"
        └──── ,"TERMINY")="Zahájení 1.1.1995\Ukončení 31.3.1995"
                ├── ,950110)="Úvodní studie\hotovo"
                ├── ,950131)="Hrubé zpracování\rozpracováno"
                ├── ,950228)="Detailní zpracování"
                ├── ,950315)="Předání"
                └── ,950320)="Ukončení a vyhodnocení"
```

Při pohledu na variabilitu zachycených dat v jedné struktuře je důležité povšimnout si rozměru zachycených dat. Zatímco relační databáze pracují s dvourozměrnými relacemi (tabulkami), pracuje M se strukturami vícerozměrnými. To je způsobeno tím, že každý uzel v globálové struktuře může nést (ale nemusí, viz fiktivní uzly) jinou informaci. Toto hnízdění relací do sebe zatím běžně používané relační databáze neposkytují.

Toto byl tedy jednoduchý příklad demonstrující možný způsob návrhu datových struktur. Další část referátu bude určena popisu programovacího jazyka M technologie.

3. Vývoj aplikací a hlavní rysy programovacího jazyka M

Programovací jazyk je velice jednoduchý, stručný, kompaktní, snadný k naučení, s jasnými syntaktickými i sémantickými pravidly. Byl navržen zejména pro efektivní manipulaci s databázemi obsahujícími řetězce proměnlivé délky i struktury.

ANSI norma jazyka M poskytuje základní příkazy, rozšiřující příkazy, zabudované funkce a standardní proměnné. Základních příkazů je zhruba tolik, kolik je písmen anglické abecedy. Všechny jména příkazů, funkcí a standardních proměnných se dají při zápisu zkrátit na 1 znak, což vede k velice kompaktnímu a krátkému zápisu programových konstrukcí. Jazyk M bývá často přirovnáván k RISC procesorům, zejména proto, že obsahuje pouze několik základních a jasných příkazů, konstrukcí a pravidel, která jsou však mimořádně rychlá a efektivní.

Programy v M se skládají z posloupnosti řádků. Každý řádek může volitelně obsahovat návěští, které je možné využít jako vstupní bod do programu (vyvolat program od zvoleného návěští). Za návěštím pak jsou jednotlivé příkazy mezi sebou oddělené mezerou, přičemž vykonání každého příkazu v M lze podmínit. Nejlepší bude opět ukázat hlavní jádro programovacího jazyka na několika příkladech :

Tento zásah může vést například k tomuto :

```
^ZAM
└──── (2)="Petr\Macháček\18.12.1958\7600"
        ├──── ,"KURS")
        │       ├──── ,1)="Kurs angličtiny\12.10.1994"
        │       └──── ,2)="Kurs první pomoci\10.1.1995\10.1.1998"
        ├──── ,"SPOJ")
        │       ├──── ,1)="TEL\02-4056849\domů"
        │       ├──── ,2)="TEL\02-3453345\do práce od 7:00 do 15:00"
        │       ├──── ,3)="ADR\Václavské náměstí 1\Praha 1\111 50"
        │       └──── ,4)="MAIL\machacek@idea.cz"
        └──── ,"DETI")
                ├──── ,"800414/1234")="Petr"
                └──── ,"845609/3214")="Alena"
```

Naprosto zadarmo získáváme v takových strukturách takzvané kaskádovité rušení. Zrušíme-li uzel s osobním číslem zaměstnance příkazem KILL ^ZAM(2) (nebo uzel s jeho dětmi příkazem KILL ^ZAM(2,"DETI")), automaticky ztrácíme pod ním zavěšenou větev a tím i cestu ke všem informacím o zaměstnanci (nebo o jeho dětech). Zde je třeba upozornit na fakt, že v tomto návrhu nemůže existovat záznam o dítěti bez jeho rodiče. V případě potřeby je možné navrhnout globál dětí ^DETI s indexem rodné číslo, který by v datech obsahoval osobní čísla rodičů.

Pro zopakování máme tedy v případě návrhu M struktur dvě základní možnosti postupu :

1. výše popsaným postupem, kdy vlastně z datového hlediska budujeme hierarchii objektu (v našem případě objektu zaměstnanec)

2. klasickým postupem návrhu relací, kdy výše uvedený příklad bude implementován jako skupina vzájemně provázaných relací zaměstnanců, kursů, spojení a dětí umístěných například v globálech ^ZAM, ^KURSY, ^SPOJENI a ^DETI, které koneckonců budou mít obdobnou strukturu jako nahoře.

Zde je třeba upozornit na existenci standardu propojení M a SQL obsažený v normě SQL2. Možnost propojení M s jinými prostředími na úrovni SQL příkazu je tedy možná a je implementovaná již všemi výrobci M systémů.

A. Základní příkazy

S	SET	nastavení hodnoty proměnné
K	KILL	zrušení proměnné a všech jejích synů
R	READ	přečtení hodnoty z aktuálního zařízení
W	WRITE	výpis hodnoty na aktuální zařízení
D	DO	provedení procedury začínající na daném návěští
Q	QUIT	ukončení procedury
F	FOR	příkaz cyklu
G	GOTO	skok na dané návěští
X	XECUTE	provedení příkazu daného parametrem

B. Standardní funkce

$D	$DATA	testuje existenci proměnné
$P	$PIECE	vrací zvolený podřetězec podle oddělovače
$E	$EXTRACT	vrací zvolený podřetězec podle pozic
$S	$SELECT	alternativní výběr hodnoty
$O	$ORDER	vrací následníka zvoleného uzlu

C. Některé základní obraty

```
START WRITE !,"Zadej osobni číslo : " READ OSC QUIT:OSC="K"
      SET EXIST=$DATA(^ZAM(OSC))
      D VYPIS:EXIST,NENI:'EXIST
      GO START
VYPIS FOR A="Osobni číslo "_OSC,1:1:4 DO
      . WRITE:+A !,$PIECE(^ZAM(OSC),"\",A)
      . WRITE:'A !,A
      QUIT
NENI  WRITE !,"Toto osobní číslo neexistuje"
      QUIT
TISK  SET OSC=""
      FOR SET OSC=$ORDER(^ZAM(OSC)) QUIT:OSC="" D VYPIS
      QUIT
```

Příklad dialogu :

Zadej osobní číslo : 2

Osobni číslo 2

Petr

Macháček

18.12.1958

7600

Zadej osobní číslo : K

Tento fragment programu ukazuje, jak se dotazovat na data o zaměstnanci.

1. Program se zeptá na osobní číslo a pokud uživatel zadá hodnotu "K", pak skončí.

2. Podle toho, zda údaje o daném pracovníkovi existují se buď skočí na podprogram vypisující data nebo na podprogram oznamující neúspěch. V podprogramu VYPIS je vidět podmíněné vykonávání příkazu WRITE. Podmínka +A znamená test, zda A obsahuje číslo a podmínka 'A (znamená NOT A) test, zda se jedná o řetězec (jinak řečeno, zda číselné vyjadření obsahu proměnné A je rovno 0, po negaci 1)

3. Poté se pokračuje opět od začátku.

Návěští TISK je připraveno pro případný výpis dat o všech pracovnících ve formátu daném podprogramem VYPIS.

Zajímavá jsou zde dvě různá použití příkazu FOR. První ukazuje postupné dosazování hodnot do proměnné A, která tak nabývá postupně hodnoty "Osobní číslo 2",1,2,3,4. Druhý příklad na řádku TISK+1 ukazuje na použití nekonečného cyklu, který je ukončen až příkazem QUIT v případě nalezení konce seznamu. Příkaz FOR je vlastně jedinou konstrukcí cyklu v M (pokud pomineme příkaz GO), ale svoji variabilitou předčí všechny konstrukce známé z jazyků C nebo Pascal (FOR, WHILE, REPEAT, ..)

Příklad vyhodnocení některých výrazů :

```
> SET A="3 jablka",B="2 hrušky"   ; nastaví proměnné A a B na řetězcové hodnoty
> WRITE +A                        ; vynutí výpis číselné reprezentace
 3
> WRITE A+B                       ; vypíše součet číselných reprezentací A a B
 5
> WRITE A_" a "_B                 ; vypíše zřetězení výrazů A, " a ", B
 3 jablka a 2 hrušky
```

D. Některé triky

```
PRIKLAD     SET A="WRITE 1+1"
            SET X="S X=A"
            XECUTE X XECUTE X
            QUIT
```

Na návěští PRIKLAD je vidět malá hříčka s příkazem XECUTE pro okamžité provedení příkazu uvedeného v řetězcovém parametru. V tomto případě se do proměnných A a X nastaví řetězce představující M příkazy pro výpis výsledku výrazu 1+1 a nastavení hodnoty proměnné A do proměnné B. Provedení prvního příkazu XECUTE způsobí nastavení proměnné X na "WRITE 1+1", druhý XECUTE zařídí výpis hodnoty 2 na aktuální výstupní zařízení. Představíme-li si však rozumnou aplikaci, kde potřebujeme v závislosti na okolnostech sestavit různé příkazy (typicky se může jednat o sestavování SQL příkazů), pak se ukáže obrovská síla příkazu XECUTE.

Poznámka : M programátor by příkazy XECUTE X XECUTE X napsal s využitím zkrácení jako X X X X, kde první a třetí X znamená zkratku za příkaz XECUTE a druhé a čtvrté X odkaz na proměnnou X.

```
SPOJENI     READ "Zadej druh spojení : ",DRUH
            DO @("SPOJ"_DRUH)
            WRITE SPOJENI
            QUIT
SPOJTEL     READ "Zadej telefon : ",TELEFON
            READ "Kam to je ? ",KAM
            SET SPOJENI=TELEFON_"\"_KAM
            QUIT
SPOJADR     READ "Zadej město : ",MESTO
            READ "Zadej ulici : ",ULICE
            READ "Zadej PSČ : ",PSC
            SET SPOJENI=MESTO_"\"_ULICE_"\"_PSC
            QUIT
```

V souvislosti s příkazem XECUTE a příkladem, kde se uváděla možnost evidence různých druhů spojení na jednoho člověka, je na rutině SPOJENI ukázána možnost jejich zpracování pomocí mechanismu nepřímé adresace. Zde v závislosti na zadaném druhu spojení příkaz DO @("SPOJ"_DRUH) předá řízení na odpovídajícímu návěští.

S tím souvisí i interpretační charakter M technologie. Skutečná kompilace M programů do strojového kódu není uskutečnitelná vzhledem k následujícím charakteristickým rysům M jazyka :

- příkaz XECUTE dovoluje vykonání libovolného řetězce vzniklého za běhu M programu, obdobně nepřímá adresace dovoluje nahrazovat za běhu odkazy na proměnné v rámci prováděného kódu. Proto by stejně musel být v době běhu programu přítomen kompletní překladač M.

- funkce $TEXT dovoluje zpřístupnit v době vykonávání programu části zdrojového kódu. To v principu znamená, že celý tento zdrojový text by musel být držen společně s kompilovaným programem. Pomocí této funkce lze realizovat vzájemnou převeditelnost programů a dat.

Většina současných verzí MUMPSu má proto předkompilátory. Předkompilovaná verze programů je uložena na disku odděleně a samozřejmě i jejich běh je rychlejší než u interpretů.

Tím, že nedochází k úplné kompilaci a spojování přeložených souborů, se ukazuje i další výhoda M. Ta spočívá v možnosti libovolného volání programů mezi sebou a spouštění jejich částí od zvolených návěští, přičemž se využívá malé velikosti programů v M (typicky 2-4KB) a vzájemné převeditelnosti dat a programů. To vede společně s příkazem XECUTE a nepřímou adresací k naplnění mechanismu nazývaného "late-binding".

4. Hlavní výhody M technologie

4.1. Přenositelnost

M je ANSI standardizovaný systém dostupný na všech hlavních HW platformách i operačních systémech, přičemž původní MUMPS byl standardizován touto normou jako třetí programovací jazyk v pořadí po Cobolu a Fortranu v roce 1977. Později byl ANSI standard M převzat i jako mezinárodní standard (ISO/IEC 11756). Tento standard obsahuje celou kapitolu týkající se přenositelnosti. Ta specifikuje nezbytné minimum požadavků, které musí všichni producenti splnit a které všichni aplikační programátoři mohou použít s jistotou, že takto napsané aplikace budou přenositelné mezi všemi implementacemi M. Implementace M jsou nyní dostupné na všech hlavních platformách (HW/OS) včetně OpenVMS, UNIX, MS-DOS, Windows, VM, Macintosh a dalších.

4.2. Produktivita a výkonnost

M bývá pro svou jednoduchost a přímočarost přirovnáván k RISC procesorům. Zdrojové kódy programů M technologie zabírají při porovnání s ostatními programovacími a databázovými systémy typicky 10-20% jejich velikosti při shodné funkčnosti. S malou velikostí programů souvisí též jejich snadná údržba a vysoká přehlednost.

M systémy jsou charakteristické svým vynikajícím poměrem mezi cenou a výkonem. Tento poměr je dán především mimořádně nízkými nároky na HW zdroje.

4.3. Použitelnost a pokrokovost

M je široce použitelný prostředek pro vývoj komplexních systémů použitelných v rozsahu od samostatného PC až po rozsáhlou síť s klient/server architekturou. V poslední době se svět informačních technologií velice rychle vyvíjí. Ani M nezůstává pozadu a neustále se doplňuje původní standard MUMPSu o další rozšíření a rozhraní jako jsou :

- M Windowing API - pro vytváření aplikací nezávislých, na konkrétním systému definované mimo jiné i pro MS Windows, X Windows, Macintosh.

- Open API - pro sdílení dat i aplikací s jinými systémy

- SQL2 - definovaný standard pro SQL volání M systémů

- Internationalization - pro podporu aplikací v japonštině, čínštině, arabštině,...

- Object Oriented Programming - pro zabudování principů OOP do standardu M technologie

4.4. Životnost

MUMPS se vyvíjel od počátku pod striktním dohledem samotných uživatelů v komisi definující standardy. Tím se zajistil konzistentní vývoj MUMPSu, který v konečném důsledku vede k obrovské životnosti M aplikací. Mnohé aplikace, které se dodnes používají, byly vyvinuty již před více než 20 lety. Typické M aplikace byly vyvinuty na PDP, poté se beze změny provozovaly na DEC/VAX a nyní pracují na DEC/Alpha procesoru a to vše bez jediné změny v kódu.

4.5. Datové struktury a programovací jazyk

Vlastní datový model M technologie dovoluje velice přirozeně modelovat reálný svět, programovací jazyk umožňuje programátorům jasně a jednoduše přepisovat myšlenky do programového kódu. Nejvýznačnějšími vlastnostmi jsou netypované proměnné (konvertované automaticky mezi numerickými a textovými hodnotami), vícerozměrná asociativní pole, persistentní proměnné (globály), dobré možnosti práce s řetězci, nepřímá adresace (dovolující využít řetězec spočítaný za běhu programu jako část M programu), zabudovaná podpora pro multiuser/multitasking.

5. Hlavní nevýhody M technologie

5.1. Datový slovník

Každá dobrá věc musí mít i svoji stinnou stránku. M technologie nabízí datový model dovolující navrhovat prakticky libovolné struktury, ovšem bez podpory centrálního datového slovníku. Ten je sice možné používat při propojení M s SQL, ale není součástí základu M technologie. Proto vzniklo mnoho nadstaveb pro popis globálových struktur a jejich vazeb, které tento problém řeší.

5.2. Disciplína v programování

Jednoduchost a stručnost ve výrazových prostředcích programovacího jazyka M vede mnoho programátorů k používání nepříliš mnemotechnických názvů proměnných, rutin a návěští.

Proto je nutné v každé firmě definovat jejich standardní názvy. Stejně tak je potřebné formalizovat a sjednotit i některé programové obraty, které lze vyjádřit několika alternativními zápisy. Pokud se tyto standardy ve firmě zavedou, mění se zmiňovaná nevýhoda v obrovskou výhodu vzhledem k jednoduchému přebírání projektů mezi programátory.

5.3. Utajení

Gartner Group označila v roce 1993 M technologii jako nejlépe utajenou informační technologii na světě. M technologie není středem zájmu sdělovacích prostředků, výuky na školách, mnoho manažerů o M nikdy neslyšelo. Důvodem bude zřejmě obrovská oblíbenost relačních SŘBD, nástup objektově orientovaných SŘBD či malá marketingová podpora MTA (M Technology Association).

6. Fakta

6.1. Standardy M technologie

ISO	ISO standard (ISO/IEC 11756)
ANSI	ANSI standard (ANSI X11.1-1990)
SQL	SQL binding (ISO/IEC 9075:1992 SQL)
USA	Federal Information Processing Standard (FIPS 125-1)

6.2. Hlavní implementace M technologie

DSM - Digital Standard M, DTM - Data Tree Mumps, M/SQL a Visual M -implementace fy Intersystems, MSM - Micronetics Standard Mumps, GSM -Greystone Mumps

6.3. Literatura

Lewkowicz John: THE COMPLETE MUMPS (MTA item #2030)
Prentice-Hall, Englewood Cliff, N.J.
ISBN 0-13-162141-6

Walters Richard F.: THE ABCs OF MUMPS (MTA item #2034)
Digital Press. ISBN 1-55558-017-3.

Krieg Arthur F., Miller David H., Bressler Gregory L.: COMPUTER PROGRAMMING IN STANDARD MUMPS (MTA item #2028)

Zimmerman Joan: INTRODUCTION TO STANDARD MUMPS (MTA item #2012)

Achtenberg Joel: MUMPS POCKET GUIDE - 1990 (MTA item #2018)

Manuály a propagační materiály firem DEC, Intersystems, Micronetics

Tesla Eltos s.p. : Programovací jazyk MUMPS, Praha 1990

Základní školení programátora IDEA s.r.o.

Celková doba zaškolení : 38 dnů

1. Základní pojmy
2. Uživatelské seznámení s SDP systémem
3. Ovládání editorů, programátorský režim
4. M technologie - datové struktury, příkazy, systémové proměnné
5. Služební programy DSM a DTM
6. SDP systém
7. Best'a CASE
8. Ostatní nástroje SDP systému
9. Základy OpenVMS

Základní pojmy **Trvání : 2 dny**

- literatura : osobní sdělení
- informace o firmách IDEA, SDP, DEC, InterSystems, Xyplex, Micronetics, Inseko (WRQ)
- pojmy MicroVax, Alpha, terminály VT, tiskové a terminálové servery, OpenVMS, LAT
- pojmy SDP systém, Reflection
- M technologie - Digital Standard M, Data Tree M, Micronetics Standard M, UCI
- seznam zákazníků a aplikací firmy IDEA

Uživatelské seznámení s SDP systémem **Trvání : 4 dny**

- literatura : Systém SDP - uživatelská příručka
- přihlášení do systému, odhlášení
- identifikace
- menu
- uživatelské helpy
- aplikační programy
- hesla
- datumové vstupy
- hledání ve find-oknech
- vícenásobné vstupy
- směrování tisku
- textové okno, wordprocesor
- ZOOM funkce

SYS
GW-bla

Ovládání editorů **Trvání : 1 den**

- literatura : osobní sdělení
- programátorský režim v DSM i DTM
- příkazy ZL, ZS, ZR, ZI, ZP
- LINE editor
    - vyvolání nápovědy - ?
    - příkazy pro hledání v textu (; a ;S)
    - kopírování části jiných rutin - ;C
    - uložení rutiny - /
    - uložení rutiny s kontrolou syntaxe- //
    - hledání v helpech SDP systému
- WP editor
    - vyvolání z LINE editoru - ;W nebo ;W120
    - odlišení programového řádku od řádku ve WP

M technologie

- literatura **Trvání : 10 dnů**
  - článek M technologie od PB na DATASEM 95
  - česky ze žluté příručky
  - anglicky z manuálu DSM - Language reference manual a Introduction to DSM
  - anglicky z manuálu DTM - M language
- datové struktury M
  - proměnné - globální vs. lokální, jednoduché vs. indexované
  - techniky indexování
  - collating sequence
  - dočasné globály
- příkazy jazyka M
  - pojmy program, návěští, funkce, parametry
  - standardní obraty - průchod globálem, tvorba uživatelských funkcí, procedury
  - předávání parametrů - hodnotou, odkazem, jménem proměnné, sdílenou proměnnou
- systémové proměnné

Služební programy DSM a DTM

- literatura : DSM - Programmer's guide chapter 3, 7 **Trvání : 1 den**
- programy na rutiny - výpis rutin, prohledávání, globální změny, ukládání a rozbalování %RD, %RSE, %RCE, %RS, %RR, %FL
- programy na globály - výpis globálů, prohlížení globálu - %GD, %G, %GTO, %GTI
- help DSM

SDP systém

- literatura : Systém SDP - Příručka programátora, vzorové programy %CSVZ*, **Trvání : 10 dnů**
- standardy - návěští, proměnné, typy programů
- knihovny programů
- řešené problémy - viz pracovní dokumentace

Besťa CASE

- literatura : osobní sdělení, %CSKEYS, %CSHELP **Trvání : 4 dny**
- generování programů - princip kreslení a předloh
- ovládání kreslení programů - %CSKEYS, findy, texty, vstupní objekty
- úpravy vygenerovaných programů
- další funkce CASE - viz %CSHELP, %CSS
- nabídka CS v menu %ZMENU5

Ostatní nástroje SDP systému

- literatura : osobní sdělení, Systém SDP - Příručka programátora **Trvání : 4 dny**
- SDP generátory
  - screen generátor
  - read generátor
  - find generátor
  - output generátor
- definování ZOOMů
- nástroje z programátorského menu %ZMENU5

Základy OpenVMS

- literatura : help VMS **Trvání : 2 dny**
- adresářové struktury , DIR, TYPE, COPY, BACKUP, INIT, MOUNT, PCDISK
- SET DEF, SET PROT
- SHOW TERM, SHOW SYS, MONITOR, STOP/ID
- logické proměnné, symboly
- editor TPU

Plán zaškolení - pokročilý

1. DECNET - NCP
2. Ovládání DECServer a Xyplex (nastvení parametrů)
3. Ovládání modemů
    - HAYES kódy
    - Penril konfigurace
    - známé potíže (E1,Q0 -› naplnění mailboxu)
4. Napojení přes modem na jiný počítač
    - konfigurace Terminál - modem - modem - počítač
    - konfigurace Terminál - počítač- modem - modem - počítač
5. Kermit - ovladání
    - poslání souboru
    - známé potíže (nelze příjmout JB.DAT)

Plán zaškolení - specialista

1. Konfigurace DSM
2. Konfigurace DTM
3. Konfigurace VMS
4. Konfigurace MS-DOS
5. Ovládání PATHWORKS v. 4x a v. 5x
6. Konfigurace PATHWORKS v. 4x a v. 5x
7. Instalace a konfigurace modelu klient - server v M
    - DTM-DTM
    - DTM-DSM
    - MSM-DSM

Plán zaškolení M - technologie :

1) Struktura dat, proměnné
   - stromová struktura, řídká pole, ukládání jen skutečné délky
   - lokální, globální, přístup ke globálům
   - jeden typ = řetězec, není potřeba deklarace, numerická hodnota
   - jména proměnných
   - holé odkazy
   - oddělovač (\)
   - collating sequence
2) Operátory
   - unární (+ - ')
   - aritmetické (+ - / # \ < > =) '>, '<, '=
   - řetězcové ([ ] ? _)
   - logické (& !)
   - priorita při vyhodnocování výrazů
3) Příkazy
   - standardní a rozšířené (S,D,F,I,K,G,O,U,C,X,Q,ZR,ZS,ZP) QUIT
   - zkracování
   - podmiňování W, R
   - nepřímá adresace
4) Funkce
   - standardní a rozšířené ($P,$D,$E,$L,$O,$S,$TR)
5) Speciální proměnné
   - standardní a rozšířené ($T,$X,$Y,$ZR)
6) Programování
   - jména návěští
   - line a full screen editor programů
   - příkazový režim
   - spuštění programu (od začátku, od náveští)
   - tisk programu
   - editor globálů (program %GM)
7) Help DSM

Literatura : - příspěvek PB na Datasem "M technologie"
- Intoduction to DSM, Language syntax
- česká příručka MUMPS
(Language Reference DTM)

Úkol : Vytvoření vlastních jednoduchých programů na testování DSM (zápis do databáze, čtení uložených dat)

IDEA spol.s r.o.