Anales PANEL'81/12 JAIIO
Sociedad Argentina de informática
e Investigación Operativa. Buenos Aires, 1981

MUMPS, porque , quando e como usar?

Martín Tornquist
Divisao Acadêmica do CPD - UFRGS
Universidade Federal do Rio Grande do Sul
Avenida Oswaldo Aranha, 99
90000 Porto Alegre, RS, BRASIL

RESUMO

Este trabalho visa apresentar um sistema simples de gerência de banco de dados (SGBD), denominado MUMPS.A sua extrema simplicidade aliada ao seu poderio, faz pleno jus à tradicional citaçao de que "na simplicidade está a virtude" e temos certeza, interessará a muitas pessoas à procura de una ferramenta mais adequada à manipulaçao de suas informaçoes.

E descrita a evoluçao histórica do desenvolvimento de técnicas de programaçao e dos SGBD, caracterizando bem as suas origens e as lacunas que posteriormente vieram a preencher. Segue-se uma análise de beneficios e ou desvantagens obtidas pelo emprego dos mesmos.

Após sao brevemente abordados os temas centralizaçao e distribuiçao de sistemas de informaçao, definido-se o ambiente típico de utilizaçao da linguagem MUMPS. Varias aplicaçoes também sao citadas.

A linguagem MUMPS é apresentada nas suas características e potencialidades, mostrando claramente os benefícios obteníveis pela utilizaçao da mesma. Algunas aplicaçoes em MUMPS, desenvolvidas na UFRGS, sao apresentadas e comentadas.

OBJETIVOS:

Este trabalho visa apresentar um sistema simples de gerência de banco de dados (SGBD), denominado MUMPS.A sua extrema simplicidade aliada ao seu poderio,faz pleno jus à tradicional citaçao de que "na simplicidade está a virtude" e temos certeza, interessará a muitas pessoas à procura de uma ferramenta mais adequada à manipulaçao de suas informaçoes.

Para isso porém, serao necessários primeiro alguns comentários críticos sobre a evoluçao dos SGBD e seu uso.

INTRODUÇÃO

No mundo atual,especificamente na área de sistemas de informação baseados em processamento eletrônico de dados,a situação geral, do ponto de vista da qualidade dos sistemas que produzimos, nos parece bastante caótica. Podemos afirmar que a nossa profissão ainda não alcançou sua plena maturidade,maturidade esta, exigida pela complexidade dos sistemas que criamos.

Perplexos? Irritados? Pois bem,quantos dos nossos leitores se arriscariam a continuar a voar,se soubessem que o nível de qualidade e ou manutenção de uma dada companhia aérea, fosse equivalente ao nível de qualidade e ou confiabilidade dos sistemas por nós produzidos ? Nós ,certamente que não!

Se analisarmos a evolução das técnicas de desenvolvimento de programas nos últimos 30 anos, observaremos alguns fatos insólitos:

- crescimento assustador do Hardware disponível para o desenvolvimento de sistemas.

- proliferação de linguagens de programação, criando uma verdadeira torre de Babel.

- desenvolvimento de um verdadeiro arsenal de programas utilitários para auxiliar em todas as fases do desenvolvimento de sistemas.

- desenvolvimento e uso de um número crescente de técnicas de desenvolvimento de sistemas.O maior enfoque foi concentrado sobre técnicas repressivas e é de se estranhar que , somente nos últimos oito anos,tenham surgido estudos sobre os aspectos psicológicos da tarefa de programação.

Analisando esta evolução, alguns aspectos saltam à vista imediatamente:

- à exceção do aumento da velocidade e capacidade de armazenamento e à exceção da diminuição da largura e altura dos computadores a sua capacidade computacional inerente, continua a mesma. É espantoso que em uma área tão dinâmica e nova,ainda estejamos amarrados a uma mesma arquitetura,datada de 30 anos atrás.Podemos garantir-lhes que não é somente por falta de melhores opções !

- a nossa ferramenta básica para o desenvolvimento de sistemas de informação,a linguagem de programação,também não evolui muito.Para uma grande maioria,terminou com FORTRAN e COBOL!

- uma boa técnica de programação,tem sido tradicionalmente avaliada pelo tempo de codificação e teste dos programas,pela velocidade de execução,pela eficiência do código objeto gerado,pela clareza do

programa,pela facilidade e custo de manutenção e pela clareza da documentação. Poucos ,porém,se preocuparam com a satisfação do usuário do sistema.Vemos o usuário como parte integrante do processo de desenvolvimento de um sistema,e isto, ao nosso ver,tem sido negligenciado.

- como a nossa capacidade computacional não se alterou,continuamos sem poder resolver os mesmos problemas (ditos não computáveis) de 3 décadas atrás.

- o equilíbrio Hardware-Software está definitivamente pendendo para o lado do Software,numa relação que,segundo alguns especialistas,chegará a 9/1 em 1985 em termos de custo dos sistemas.

- o programador como ser humano , foi largamente desprezado,submergindo sob um mar de ferramentas e técnicas ,que teoricamente,pretendiam ser a solução dos seus problemas.Não conseguimos fugir da idéia,que o programador típico de hoje,perde um tempo excessivo com a manipulação e aprendizado de técnicas e linguagens de apoio, quando as maiores soluções para os atuais problemas estão dentro do programador e ou analista. Não podemos esquecer que , a nossa profissão consiste literalmente em montar castelos no ar.Poucas são as profissões que apresentam esta característica de extrema abstração. Sobre este último aspecto, Brooks já dizia : "The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds castles in air, from air, creating by the exertion of the imagination. Few media of creation are so easy to polish and rework, so readily capable of realizing grand conceptual structures. (As we shall see later, this very tractability has its own problems.)" (Brooks,1975) As ferramentas disponíveis hoje,pouco se preocupam com este fato e , por isso mesmo,falham!O maior problema hoje não é a falta de técnicas,é a falta de posicionamento individual,de grupo e de educação. Acreditamos que muitas das atuais técnicas de programação têm nivelado por baixo,restringindo uma característica exclusivamente humana e rara: a criatividade.

EVOLUÇÃO DOS SGBD

Os anos 60 presenciaram à centralização do processamento de dados,ditada por economias de escala.As necessidades dos anos 70,por sua vez ,forçaram a criação de sistema ON-LINE de acesso direto. Linguagens,sistemas operacionais e métodos de acesso às informações , têm profundos efeitos sobre programadores,gerentes e usuários.Gerenciar o trabalho de programaçao em um ambiente de rápidas e constantes mudanças,certamente não é tarefa simples.

Foi neste ambiente e com esta herança que os SGBD surgiram como a resposta a todos os problemas acima citados: centralização,acesso de informações ON-LINE e padronização de estruturas complexas.

Antecipando-nos a sua descrição , MUMPS já nasceu em um ambiente de acesso direto,ON-LINE, a um número reduzido de usuários(2 a 32).MUMPS não foi utilizado em processamento BATCH e as economias de escala não ditavam a sua centralização.

Sobre SGBD , uma palestra do Bachman marcou época:

"Just as the ancient viewed the Earth with the Sun revolving around it , so have the ancients of our information systems viewed a tab machine or computer with a sequential file flowing through it ... Each was an accurate model for its time and place ... This revolution in thinking is changing the programmer from a stationary viewer of objects passing before him in core into a mobile navigator who is able to probe and traverse a data base at will." (Bachman , 1973)

Um típico SGBD (Date , 1973) apresenta cinco componentes principais:

a) uma linguagem hospedeira,tipicamente uma linguagem de uso geral como COBOL ou PL/1.

b) uma linguagem de definição de dados,utilizada para a especificação da forma de armazenamento e do método de acesso aos dados.

c) um modelo de dados,que controla a informação que o usuário vê.

d) um sub-modelo de dados,que controla a parte das informações vista por um usuário individual.

e) um sistema de acesso direto,tipicamente um componente do S.O. empregado.

Date define uma série de vantagens a serem obtidas pelo uso do SGBD:

a) diminuição da redundância dos dados.

b) possível diminuição da inconsistência dos dados armazenados no BD.

c) compartilhamento de dados entre diversas aplicações.

d) facilidades para criar e manter padronizações.

e) facilidades para a aplicação de medidas de segurança.

f) manutenção da integridade física e lógica do BD.

g) independência dos dados.

h) disponibilidade de complexas estruturas de armazenamento e recuperação de dados.

A independência dos dados é citada como uma das maiores realizações dos SGBD. Date a definiu como "imunity of applications to change in storage strucures and access strategy". (Date,1975) A representação física dos dados poderá , portanto , ser modificada,sem influir no Software já desenvolvido sobre os mesmos dados.

Os SGBD por si só ,poderão gerar uma dependência procedural,isto é, a manutenção das hierarquias e redes do SGBD independendentemente de seu conteúdo. Os SGBD substituem dependência de dados por independência de dados.Ainda não sabemos se isto realmente é de real benefício.A maioria dos SGBD existentes,não conseguiram criar uma independência total da base de dados.

CUSTOS DE UM SGBD

Os custos de aquisição ou aluguel dos SGBD no mercado são elevadíssimos.Pior ainda,os custos de manutenção e Hardware do sistema,rapidamente superam os custos do SGBD propriamente dito. A necessidade de maior treinamento dos programadores,o tempo de processamento BATCH diminuído,o aumento dos custos de manutenção de arquivos e a maior complexidade geral do sistema,são outros fatores de custo menos aparentes.

COMPLEXIDADE DE UM SGBD

Um SGBD típico é um pacote de Software extremamente complexo,requerendo normalmente enormes quantidades de memória para uma boa performance. A complexidade de ajustar o Software ao aumento do número de usuários,cresce mais que proporcionalmente. A performance é altamente dependente da atuação do administrador do BD e de sua equipe.

Um erro em um SGBD frequentemente atravessa o domínio de cinco a seis linguagens,descritas em milhares de páginas de documentação.

IMPACTO NA ORGANIZAÇÃO

Muitas companhias instalaram um SGBD na ingênua esperança de grande e gloriosa centralização de informações.Para este estado grandemente contribuíram as tão propaladas SIG (Sistemas de Informação Gerenciais). A consolidação de arquivos e procedimentos de diversos departamentos é uma tarefa hercúlea em qualquer organização estabelecida.Cada departamento possui seus sistemas manuais estabelecidos e procedimentos próprios no uso do computador.Tentar removê-los , dentro de um departamento , é difícil,que dirá tentar integrá-los a nível de organização ! O pior é que , é perfeitamente possível que,junto à entrada em funcionamento do SGBD,surjam novos procedimentos manuais paralelos para atender às omissões e novas necessidades.

Muitos profissionais de PED vêem nos SGBD um fim em si e que a última tecnologia irá trazer a solução para todos os seus problemas criando "bons" sistemas. O SGBD , porém, força um fluxo de informações estranho ao departamento em nome da organização.Isto certamente será duro de engolir para muitos gerentes. Em face destes problemas,o enfoque de distribuição da capacidade computacional ganha novo ímpeto.Cada departamento ,ou departamentos , que dividam as mesmas informações,seguiriam seus próprios caminhos de computação. As prioridades e o andamento dos trabalhos seriam então geridas pelas normas tradicionais da gerência. Isto não é uma volta ao estado caótico de descentralização de 1960 ! As economias de escala oferecidas por computadores de grande porte,além de não serem mais necessariamente verdadeiras,são denegridas pela degradação do tempo de resposta e prontidão do sistema. Novas tecnologias,como MUMPS,se adaptam perfeitamente a este tipo de ambiente.

Apesar de seu custo, o mercado dos SGBD está crescendo,portanto está satisfazendo necessidades do mundo real ! Quais seriam estas?

a) estruturas de acesso direto eficientes e poderosas.

b) capacidade de comunicação ON-LINE.

c) capacidade de controle e segurança dos dados.

d) compatibilidade com o Software e Hardware existentes.

Logo , o sucesso dos SGBD reside basicamente nestas lacunas que preencheu. Conseguir ligar um /360 a vários terminais já foi um façanha em si , pequena , porém , se comparada com a adição das complexas estruturas de acesso de dados que os SGBD oferecem. Muitos usuários de SGBD optaram pelos seus sistemas apenas por estes dois fatores.

CENTRALIZAÇÃO x DISTRIBUIÇÃO

Os SGBD evoluíram das necessidades do processamento centralizado . MUMPS , por sua vez,sempre foi orientado para mini-computadores. Como as diferenças de performance entre os minis , os antigos midis e maxis está diminuindo cada vez mais, nem a economia de escala será mais necessariamente válida. Existe até quem já propôs uma alteração da velha lei de Grosch,afirmado que: "A performance de um computador é inversamente proporcional ao quadrado de seu preço" (Adams , 1962).

O usuário está ficando cada vez mais exigente (com razão) e sofisticado nas suas necessidades, o computador está por sua vez se tornando cada vez mais acessível e a programação menos esotérica. A maioria dos departamentos de uma empresa vai querer ter seu computador. Wagner criou o seu famoso "princípio da descentralização" , enunciado como : "If an organizational group , smaller than 30 people , requires computer assistance , it is better for the total enterprise that those people have exclusive use of their own computer - provided that the computer , big enough to do the job properly , will be loaded to over 10% of its capacity." (Wagner , 1976) . Os usuários gerenciarão os seus próprios recursos,estabelecerão suas próprias prioridades e manipularão suas informações.A maioria das mesmas serão intra-departamentais e o compartilhamento de informações será geralmente a exceção,não a regra. Na próxima década assitiremos à integração cada vez maior do computador nas empresas.

Em suma, os usuários estão sentindo as seguintes possibilidades e ou vantagens no uso de um computador próprio:

a) computação a custo baixo e previsível.

b) liberdade total de desenvolvimento.

c) facilidade de programação.

d) performance relativa melhorada.

e) potencial de crescimento e flexibilidade maiores.

f) ausência de controles e restrições externas.

g) grande motivação do usuário.

h) sistemas mais simples, portanto geralmente mais confiáveis.

i) menor tempo de resposta em aplicações interativas.

MUMPS é adequado a este enfoque,entretanto a parafernália administrativa que , normalmente está associada a um SGBD, não necessariamente se aplica a um ambiente MUMPS. O SGBD tem que ser suficientemente generalizado para atender a uma vasta gama de métodos , estruturas de acesso e interfaces com as linguagens hospedeiras. Muitas destas funções generalizadas dos SGBD são incorporados ao MUMPS sob a forma de funções e comandos simples. O programador tem ao seu dispor um completo conjunto destas funções,de fácil uso e entendimento. Seu trabalho será o de combiná-los adequadamente para uma determinada aplicação.Se necessitar de alguma estrutura de acesso peculiar ou de alguma organização de arquivos ausente naquelas primitivamente definidas no MUMPS, ele as criará fácilmente.

Os SGBD representam a evolução natural dos antigos sistemas em BATCH. Portanto o problema de compatibildade é bastante crítico. MUMPS por sua vez , apenas requer que seja compatível com ele mesmo.

O potencial do MUMPS assustaria a muito gerente de PED.Para esses , o conceito de uma equipe de programação , se restringe a grupos de programadores COBOL,trabalhando em cima de folhas de codificação,protegidos de realizarem grandes estragos pelo S.O. e pelo SGBD. Programadores de sistemas,analistas,operadores e o administrador do banco de dados,todos,têm papéis comuns ao do programador. Para o programador MUMPS, esta estrutura parece opressiva.Ele desempenha o papel de todas as especialidades acima-citadas.Ele pode em questão de horas , realizar tarefas,que levariam semanas em SGBD convencionais.A natureza interativa da linguagem normalmente leva o programador MUMPS a um profundo conhecimento de sua ferramenta. Esta flexibilidade tem seus problemas.Na pressa de terminar o programa , ou por motivos outros , o programador poderá utilizar truques ou simplificações. Os atuais programadores foram treinados a escrever programas "eficientes" . Truques de programação,quanto mais escuros melhor,são fatos corriqueiros no nosso dia a dia. Na verdade é uma característica inerente a todos nós.Qualquer linguagem ou SGBD está sujeita aos seus malefícios. Pensem no político que consegue falar o melhor português por meia hora e não chegar a conclusão alguma. A única maneira de eliminar estes truques é através do esforço consciente do programador.Para isto ele deverá ser exposto às boas técnicas e a boas razões para não utilizar aquelas consideradas negativas.

CARACTERÍSTICAS DO MUMPS

Este trabalho foi introduzido com o intuito de mostrar àqueles que se interessaram pela primeira parte, algumas das principais características do MUMPS, tecendo ainda alguns comentários sobre a nossa experiência com a linguagem.

MUMPS é um acrônimo para Massachussets General Hospital Utility MultiProgramming System. O sistema foi desenvolvido em 1972 e cresceu de baixo para cima. Isto é, trata-se de um sistema criado por usuários, para usuários e portanto de domínio público. Nenhum fabricante foi responsável pela sua criação ! O entusiasmo despertado pela linguagem, fez com que ele rapidamente se espalhasse e fosse oficialmente reconhecido um STANDARD MUMPS, pela ANSI (American National Standards Institute), em 1975. A ANSI já reconheceu quatro linguagens: FORTRAN, COBOL, MUMPS e BASIC (por ordem cronológica) e atualmente está estudando a inclusão da linguagem PASCAL). A linguagem é mantida pela ANSI e por grupos de entusiastas MUMPS chamados MUG's (MUMPS USER GROUP's). No Brasil, o MUG-BRASIL existe desde outubro de 1980. A maioria dos fabricantes (inclusive de computadores de grande porte como o B-6700 e /370) já oferecem versões do MUMPS nos seus equipamentos.

MUMPS tradicionalmente tem sido implementado em mini-computadores dedicados e nestes , age como sistema operacional. Neste papel, MUMPS desempenha as seguintes funções:

a) implementa um sistema de TIME-SHARING, para atendimento de "n" usuários, por terminal.

b) controla todo o tráfego de mensagens entre os terminais-UCP-demais periféricos.

c) implementa rotinas que permitem a criação, deleção, inserção e recuperação de dados em uma base de dados de modelo hierárquico. MUMPS chama esta estrutura de GLOBAL, pois todos os seus usuários poderão acessá-la simultaneamente.

d) implementa um interpretador de uma linguagem também chamada MUMPS.

Pelo acima exposto , fica claro que MUMPS é um sistema dedicado e orientado para aplicações conversacionais (interativas) de entrada e ou recuperação de informações (manipulação de textos), segundo um modelo hierárquico. O modelo de representação de dados hierárquico ,apesar de ser possível mostrar que não se adapta a todas as situações da vida real, atende à grande maioria das mesmas. Basta lembrar que o sistema IMS, que segue um modelo hierárquico, é um dos SGBD mais utilizados no mundo .

A linguagem MUMPS é uma linguagem procedural de alto nível que incorpora os habituais comandos de controle de fluxo de programas das linguagens mais tradicionais. O seu aprendizado é extremamente rápido, pois como a linguagem é interpretativa, a interação do programador com a linguagem pelo terminal, traz excelentes frutos a curto prazo. A nossa experiência tem mostrado que programadores aprendem a linguagem em duas semanas , enquanto que leigos precisam um pouco mais de um mês para dominá-la. Talvez tenha passado despercebido,mas como MUMPS normalmente está implementado em uma máquina dedicada, o programador apenas necessitará aprender UMA única linguagem.Não é por nada que se fala em uma "máquina MUMPS".

A linguagem apenas reconhece um tipo de dado, o "string" de comprimento variável e oferece uma ampla gama de operadores e funções para a sua manipulação,por conteúdo e formato ("pattern matching"). Esta capacidade de manipulação de caracteres , torna MUMPS uma ferramenta ideal para aplicações em escritórios,laboratórios,pequenas e médias empresas,bibliotecas,uso próprio , CAI's e CAD's ...etc. Não é de se estranhar que MUMPS domine completamente o cenário da computação na área médica nos EUA, seu país de origem.

MUMPS incorpora diretamente a capacidade de manipulação de estruturas hierárquicas. Poderão existir quantas arvores o programador desejar e estas serão esparsas , significando que os únicos nodos criados são aqueles aos quais foram explicitamente atribuídos valores , ou aqueles necessários para criar os caminhos de acesso da raíz aos novos nodos. Esta generalidade (que se estende às variáveis subscritas locais) , permite ao programador a utilização desta estrutura com os índices peculiares a uma dada aplicação.Algumas implementações de MUMPS aceitam "strings" como subscrito, permitindo , portanto , acesso por conteúdo. É , porém , importante lembrar , que o grau de generalidade obtido pelo MUMPS na sua estrutura hierárquica é um compromisso entre espaço e tempo, se comparada com um sistema no qual os subscritos são ordenados linearmente. Sendo o MUMPS um sistema de múltiplos usuários,que permite o acesso simultâneo dos mesmos , às estruturas hierárquicas ("Database sharing"), existem formas de garantir o acesso mutuamente exclusivo às mesmas, para evitar a perda de dados e ou abraços mortais ("DEADLOCK's").

A linguagem procura a maior independência dos dados possível e por isso mesmo não possui declarações de tipo algum.A facilidade de uso e conversão de construções que , em outras linguagens seriam representadas por vários tipos diferentes, possivelmente conflitantes e sem regra de conversão definida,realmente liberam o programador MUMPS de quaisquer restrições de máquina,permitindo a sua concentração total na solução do problema que estiver tratando !

Estas características da linguagem trazem consigo duas importantes consequências:

a) livre criatividade do programador, independente da máquina.

b) alta receptividade e entusiasmo pela linguagem.

Estes dois aspectos, aliados às características da linguagem, geram resultados surpreendentes a curto prazo !

Afirmamos , sem sombra de dúvida, que o desenvolvimento de sistemas em MUMPS, se comparados com o desenvolvimento de sistemas similares em outras linguagens, traz consigo as seguintes vantagens:

a) redução do tempo de desenvolvimento , de meses para semanas e de semanas para dias.

b) alta receptividade pelo usuário, devido à flexibilidade e manutentabilidade do sistema.

c) tempo de manutenção drasticamente reduzido (uma das características do MUMPS é a extrema facilidade de alterar programas).

d) maior confiabilidade devido a:

- estrutura modular forçada da linguagem.
- tamanho reduzido de cada rotina (10 a 40 linhas).
- grande motivação pela linguagem, normalmente encontrada nos seus programadores.

e) diminuição da fase de teste e depuração dos sistema, pelas facilidades oferecidas por um interpretador.

Gostaria de mostrar um exemplo da facilidade de teste e depuração que o MUMPS oferece: suponhamos que um programa MUMPS, que já estava processando a 15 minutos, de repente pára por algum motivo ! O programador MUMPS poderá imediatamente visualizar o estado de toda a sua base de dados, corrigí-la se necessário , editar o programa , se necessário , e continuar a execução a partir do ponto de parada. Parece-me que não será necessário comparar esta técnica com a vigente em CPD's que empregam linguagens convencionais !

O sistema MUMPS também apresenta algumas restrições, que deverão ser cuidadosamente analisadas antes de partirmos para a sua utilização.

- a plena capacidade da linguagem só é atingida quando implementada em máquina dedicada , normalmente um mini-computador. Quando sob a égide de um S.O. de uso geral, a linguagem geralmente se apresenta com algumas restrições, principalmente em termos de tempo de resposta.

- por ser uma linguagem interpretativa, o seu calcanhar de Aquilles é um possível tempo de resposta desastroso. Interpretadores já fizeram grande sucesso na década de 60 mas foram abandonados em favor dos compiladores, devido a este problema ! Porém, com os recentes avanços da tecnologia, que anunciam microprocessadores de 32 bits para 1981 a preços irrisórios, a tendência reverterá em benefício dos interpretadores , pois nenhum compilador poderá jamais oferecer a flexibilidade de um interpretador. Cabe aqui lembrar que MUMPS se destina a aplicações "IO-BOUND" e não àquelas que são "CPU-BOUND". Como nas primeiras , o tempo de transmissão e ou recepção e o tempo de interação do operador no terminal são enormes, se comparados com um ciclo da UCP, a relativa lentidão de um interpretador (quando comparada a um compilador) não se deverá fazer sentir.

- a linguagem somente poderá ser aproveitada em todo o seu potencial por bons programadores. MUMPS é uma linguagem elitista até certo ponto. Por não apresentar restrições alguma ao programador, os resultados dependerão essencialmente de sua conscientização da função de programação.

- o uso intenso de programação interativa permite que surja a figura do programador que desenvolve,programa e testa os seus sistemas ON-LINE, de preferência ao mesmo tempo ! Para evitar este tipo de elemento, a gerência da programação deverá tomar medidas cabíveis.

APLICAÇÕES

A UFRGS (Universidade Federal do Rio Grande do SUL) utiliza um sistema MUMPS no seu equipamento B-6700, desde julho de 1979. O seu maior usuário é O Núcleo de Informática Médica,grandemente impulsionado pela utilização do MUMPS.

Lideramos um pequeno grupo de pesquisa que tem desenvolvido algumas aplicações em MUMPS,com o objetivo-mor de testar a sua real capacidade. O primeiro projeto foi o desenvolvimento de um cross-compilador da linguagem PASCAL (BYTE , Setembro , 1978), para o computador HP-2100.Neste projeto foram gastos um total de 270 horas, da concepção à entrega do sistema. O programa é composto de 50 rotinas de em média 8 linhas, num surpreendente total de 480 linhas ! O mesmo projeto foi realizado na linguagem ALGOL, sendo que levou o dobro do tempo para a sua realização e gerou um programa de quase 3000 linhas !

Com o objetivo de testar os efeitos da linguagem com programadores que a desconheciam, foi oferecido um curso de 20 horas sobre MUMPS. No decorrer do curso foi entregue a definição de um sistema de recuperação de informações bibliográficas, para cada um dos integrantes do mesmo. A definição do sistema previa quatro módulos , cadastramento , listagem , atualização e recuperação das seguintes informações: autor,obra,ementa,palavras chave,código da obra e comentários livres associados a cada obra.Foi ainda definida uma linguagem de acesso a estas informações, que permitia o emprego de operadores booleanos e relacionais. Para medir o grau de entusiasmo despertado pela linguagem, o trabalho não foi caracterizado como sendo de entrega obrigatória. Ao cabo de duas semanas após o curso, o primeiro aluno entregou seu trabalho.O sistema completo , funcionando , tinha 130 linhas !! (a listagem ocupava apenas duas folhas). De um total de oito alunos, três alunos terminaram o projeto com resultados similares dentro de um mês após terem aprendido a linguagem.Os demais apenas o concluíram parcialmente, alegando falta de tempo. Consideramos os resultados obtidos como altamente positivos e encorajadores.

O terceiro projeto, de codinome DIÁLOGO , visa estudar as capacidades do MUMPS na criação de diálogos em linguagem semi-natural. Além da entrada e ou recuperação de informações, este sistema se prestará a uma futura inclusão em um sistema de avaliação de alunos por terminal, onde permitiria respostas em aberto e a manutenção de um diálogo com os alunos. Os resultados obtidos até o momento são altamente estimulantes !

CONCLUSÃO:

O MUMPS , contrariamente a muitos sistemas , acredita e investe na capacidade e conscientização de seus programadores . O sistema , apesar de ser voltado para aplicações conversacionais de recuperação de informações , é suficientemente flexível para atender a qualquer aplicação comercial desde que não de cunho essencialmente matemático-estatístico.

MUMPS parece ser a ferramenta ideal para as inúmeras pessoas (pequenas empresas , grupos de pesquisa etc.) que , embora necessitadas de capacidade computacional ON-LINE no seu trabalho do dia a dia , não a obtém devido ao seu elevado custo , sua demasiada complexidade ou simplesmente a dificuldade de acesso aos recursos centralizados . Para estes , uma poderosa "máquina MUMPS" sobre a mesa , de fácil manejo e programação , pode ser a solução ! Para aplicações de maior porte , um mini-computador dedicado ao MUMPS , permitirá a realização de excelentes trabalhos a curto prazo .

É a nossa opinião que as vantagens da utilização de um computador próprio , citadas neste trabalho , poderão ser plenamente aproveitadas utilizando-se o sistema MUMPS.

No barco dos sistemas, o programador será navegador, como sugere Bachman, ou capitão , como defende o MUMPS ? Serão os programadores e ou analistas capazes de elevarem o nível de confiabilidade e manutentabilidade de seus sistemas,ou isto terá que ser atingido por medidas repressivas por parte do S.O., SGBD e a gerência ?

BIBLIOGRAFIA

Adams , C. W. , "Grosch's Law Repealed" , DATAMATION , May 1692

Auerbach , "Problems in Decentralized Computing" , System Developmente Considerations

Auerbach , "Managerial and Economical Issues in Distributed Computing" , General Management , Management Planning

Bachman , C.W. , "The programmer as Navigator" , Communications of the ACM , 16 , 11 , 1973

Brooks , F. P. , Jr. , The Mythical Man-Month , Massachussets , Addisson-Wesley , 1975

Date , C. J. , An Introduction to Database Systems , Massachussets , Adisson-Wesley , 1975

Munnecke , T. H. , "Data Base Managemente Systems - Friend or Foe ? " , Proceedings of the MUMPS Users Group Meeting , Loma Linda , 1975

. . . , "A tiny PASCAL Compiler" , BYTE , Setembro , 1978

O'Neil , J. T. , Editor , MUMPS Language Standard , NBS Handbook 118 , Washington , 1976

Wagner , F. V. ,"Is decentralization inevitable?" , DATAMATION , Nov 1976

Walters , R. F. , Mumps Primer , revised , MUMPS Development Commitee , Bedford , 1979

Wiederhold , G. , Database design , Stanford , McGraw-Hill , 1977

Zimmerman , J. , Editor , "What is MUMPS" , MUMPS Users Group , St. Louis , 1975