Artigo Técnico

Revista de
Informática
Teórica e Aplicada

Modelos de Processamento em Sistemas de Banco de Dados para Aplicações de CAD

Cirano Iochpe
II/CPGCC – Universidade Federal do Rio Grande do Sul

Sumário

Este artigo faz um paralelo entre os sistemas de banco de dados que suportam aplicações comerciais e aqueles que vem sendo desenvolvidos para apoiar aplicações de CAD. As principais características de sistemas de banco de dados para aplicações comerciais são apresentadas e suas dificuldades em apoiar aplicações de CAD de maneira eficiente são apontadas. Num segundo momento, os novos sistemas de banco de dados que vem sendo especialmente desenvolvidos para suportar aplicações de CAD são introduzidos e comentados com ênfase nos modelos de processamento que oferecem. A última parte do artigo apresenta uma técnica de gerência de transações de CAD para sistemas de banco de dados não convencionais.

Abstract

This paper first compares database systems for business-related applications and those which are being developed to support CAD applications. The main characteristics of business-oriented database systems are presented and their difficulties in efficiently supporting CAD applications are pointed out. In a second step, the aspects related to the processing models underlying database systems for CAD applications are discussed. The last part of the paper presents a new technique for managing CAD transactions.

Palavras-chave: sistema de banco de dados, transações de CAD, modelos de processamento.

1. Introdução

Nos últimos anos, a comunidade científica, que investiga sistemas de banco de dados (SBDs), vem investindo muito de seu potencial na pesquisa e desenvolvimento de sistemas que não só suportem as tradicionais aplicações das áreas comercial e afins, mas também possam apoiar com eficiência aplicações ditas não

convencionais, as quais apresentam requisitos novos de modelagem, armazenamento e processamento de dados. Aplicações como projetos da área de engenharia assistidos por computador (CAD), sistemas de manufatura assistidos por computador (CAM), engenharia de software apoiada por computador (CASE) e inteligência artificial são apenas alguns exemplos das muitas aplicações ditas não convencionais para as quais uma nova geração de SBDs está sendo proposta e desenvolvida. Setores da indústria assim como universidades da europa, E.U.A. e Japão já estão testando protótipos de sistemas de banco de dados não convencionais (SBDNCs). A título de exemplo, pode-se citar o sistema POSTGRES [18] da Universidade da Califórnia em Berkeley e o sistema DAMOKLES [17] da Universidade de Karlsruhe na República Federal da Alemanha.

Grande parte das pesquisas desenvolvidas, até agora, na área de SBDNCs concentrou-se na definição, projeto e implementação de modelos de dados adequados para representar e manipular os principais objetos de dado usados por aplicações não convencionais. A investigação de arquiteturas de software e hardware que melhor suportem tais aplicações assim como o desenvolvimento de modelos de processamento que modelem suas características dinâmicas de forma correta e eficiente está, porém, ainda no seu início. Com certeza, os resultados de tais pesquisas serão de grande importância, pois contribuirão para o aumento da eficiência (e, consequentemente, para o aumento da popularidade e utilização) de sistemas de banco de dados que suportem aplicações industriais e da área de engenharia. O presente artigo mostra porque o modelo tradicional de processamento adotado por sistemas de banco de dados para aplicações convencionais como, por exemplo, sistemas bancários e sistemas de reserva de passagens aéreas, não se adequa às caracteristicas dinâmicas de aplicações da área de engenharia, mais precisamente das aplicações de CAD. A partir daí, o artigo descreve três novas classes de modelos de processamento de dados que visam adaptar os serviços do SBD às necessidades de aplicações não convencionais. O último capítulo do artigo apresenta uma técnica de gerência de transações de CAD do tipo "group transactions", as quais representam o esforço de um grupo de projetistas cooperando no desenvolvimento de um projeto comum.

2. Sistemas de Banco de Dados para Aplicações Comerciais

A tecnologia de banco de dados foi criada para suportar aplicações computacionais nas quais os mesmos dados são compartilhados por diversos programas. A idéia de compartilhar dados comuns surgiu em meados da década de 50 através da introdução das chamadas "common areas" na linguagem de programação FORTRAN, as quais permitem a comunicação entre programas. No decorrer dos últimos trinta anos, novas técnicas vem sendo propostas assim como novos mecanismos vem sendo desenvolvidos para melhor suportar o conceito de compartilhamento de dados em sistemas de computação.

Os primeiros a sentirem necessidade de bancos de dados compartilháveis foram os implementadores de sistemas de computação para aplicações comerciais. Este fato pode, em parte, explicar porque a tecnologia de banco de dados evoluiu para o desenvolvimento de sistemas que, basicamente, suportam as necessidades e requisitos de tais aplicações. Pelo fato de representarem a maioria dos sistemas de banco de dados hoje existentes, os SBDs que suportam aplicações comerciais são normalmente denominados de sistemas de banco de dados convencionais. Esta denominação distingue tais sistemas daqueles, ditos não convencionais, que vem sendo construídos nos últimos anos para suportarem outros tipos de aplicações tais como CAD, CAM, inteligência artificial, engenharia de software, reconhecimento de som e imagem e cartografia.

A idéia básica atrás do conceito de banco de dados é capacitar os sistemas de computação para gerenciarem, de maneira confiável, grandes quantidades de dados e permitir que estes dados sejam compartilhados por vários usuários que desenvolvem uma ou mais aplicações de forma concorrente. Há três décadas, pesquisadores da indústria e de universidades vem desenvolvendo conceitos e implementando sistemas para fazer desta idéia uma realidade. Os SBDs atuais baseiam-se em três conceitos principais: *independência de dados, transações* e *independência de localização* [4].

Independência de dados representa a capacidade do sistema de banco de dados de esconder do programador de aplicações (e do usuário final) detalhes de implementação do sistema. A independência de dados garante que mudanças na definição do sistema de banco de dados (ex.: a introdução de um novo tipo de dado, a exclusão de algum caminho de acesso aos dados) não afetam os programas de aplicação já existentes. Em outras palavras, modificações na estrutura e/ou conteúdo do banco de dados não implicam em modificações nos programas de aplicação. Similarmente, alterações em programas de aplicação assim como a criação de novos programas não devem alterar a funcionalidade do SBD.

O conceito de *transação* assegura o isolamento do trabalho concorrente feito sobre o banco de dados assim como a reconstrução da base de dados em caso de falhas tanto do sistema como das aplicações. A transação representa uma unidade lógica de trabalho do usuário e é, tipicamente, formada por um conjunto de instruções de manipulação de dados. A figura 1 apresenta um exemplo de transação que, para melhor entendimento, está escrita em uma pseudo-linguagem de programação de alto nível. As instruções "Início-Transação" e "Fim-Transação" delimitam o trabalho a ser desenvolvido no âmbito do banco de dados para realizar uma tarefa específica da aplicação. A transação da figura 1 debita Cr$100,00 da conta corrente de número 1 e credita este valor na conta de número 2. Esta transação pode ser interpretada como a manipulação de dados necessária para compensar um cheque. A fim de testar se o trabalho executado mantém o banco de dados em um estado consistente, a transação de nosso exemplo verifica se a soma dos saldos de contas correntes feita depois da compensação é igual àquela realizada antes da execução desta operação

bancária. Em caso de erro, a transação aborta, isto é, abre mão de todas as modificações feitas desde seu início até a identificação do erro.

```
Transação1 "Débito/Crédito"
Início_Transação
   Abra       (Arquivo Contas Correntes)
   Some       (Saldo de todas as contas, Result -> Temp1)
   Adicione   (Conta_#1.Saldo, Cr$100,00)
   Subtraia   (Conta_#2.Saldo, Cr$100,00)
   Some       (Saldo de todas as contas, Result -> Temp2)
   Se         (Temp1 diferente Temp2)
   então      Aborte_Transação
Fim_Transação
```

Figura 1: Exemplo de transação que poderia ser executada por uma aplicação bancária

Assumindo que as transações são construídas de maneira a manter a consistência do banco de dados, o SBD assegura três outras propriedades inerentes ao conceito de transação [6]. As transações são *atômicas*. Isto é, o sistema garante que se algum erro ocorrer durante a execução de uma transação, esta será abortada e nenhum de seus efeitos permanecerá no banco de dados. As transações executam em *isolamento*. O SBD garante que cada transação só pode acessar e manipular informações criadas ou atualizadas por transações que já terminaram (com sucesso). Com isso, as transações ficam impedidas de acessar resultados temporários gerados por outras transações que ainda estão em andamento. Por último, o SBD deve garantir a *persistência*, no banco de dados, dos resultados gerados por transações que terminam corretamente ("committed transactions") mesmo na presença de falhas do sistema.

O isolamento garantido na execução de transações concorrentes forma a base para o correto compartilhamento de dados. Se cada transação só pode acessar dados que foram inseridos ou modificados por transações que encerraram sua execução de maneira correta e é sabido que tais transações mantém a consistência do banco de dados, então transações em andamento acessam somente dados consistentes, mesmo que estejam executando em paralelo com outras transações.

As propriedades de atomicidade da transação e de persistência de seus resultados finais formam a base para a manutenção da integridade e da consistência da base de dados em caso de falhas do sistema ou de transações em andamento.

Os sistemas de banco de dados podem ser construídos de forma centralizada ou distribuída. SBDs centralizados executam sobre arquiteturas de computadores monoprocessadas. Já os SBDs distribuídos executam sobre arquiteturas multiprocessadas onde tanto as unidades de processamento quanto partes do

banco de dados podem estar geograficamente distantes umas das outras. Sistemas de banco de dados distribuídos implementam o conceito de *independência de localização*. O fato de que certas informações necessárias ao processamento de determinada transação estão localizadas em outras partes do sistema deve permanecer transparente ao usuário que está executando tal transação.

A atividade das aplicações comerciais sobre o banco de dados é caracterizada por transações de curta duração em sistemas do tipo "batch" e por diálogos curtos entre o usuário e o SBD em sistemas de teleprocessamento. Tanto transações curtas (embutidas em programas de aplicação) como passos de diálogos (em consultas ao banco de dados) são, usualmente, processados em alguns segundos. Tipicamente, os dados acessados por tais transações ficam em seu poder por pouco tempo e logo são devolvidos ao controle do SBD. Pesquisas indicam que a maioria das transações de uma mesma aplicação convencional tendem a manipular uma mesma e relativamente pequena porção da base de dados [5]. Esta alta localidade no acesso aos dados implica em uma maior concorrência por parte de transações que executam em paralelo.

O número de transações que executam concorrentemente em sistemas de banco de dados que suportam aplicações convencionais varia de acordo com a aplicação. Enquanto é normal verificar uma média de menos de uma transação por segundo em sistemas pequenos de teleprocessamento (ex. SBDs conectados a cinco ou dez terminais), existem algumas aplicações (como é o caso de sistemas de reserva de passagens aéreas) que exigem uma média de mais de duzentas transações por segundo.

3. Sistemas de Banco de Dados para Aplicações de CAD

3.1 Algumas características básicas das aplicações de CAD

As aplicações de CAD (computer aided design) se dedicam aos planejamento, especificação e realização de sistemas técnicos. Tais aplicações estão normalmente associadas às atividades da área de engenharia. Exemplos de aplicações de CAD são o desenvolvimento de sistemas de engenharia de software, o projeto de peças mecânicas e o desenvolvimento de circuitos integrados. Devido à crescente complexidade destas aplicações, sistemas de computação vem sendo construídos para apoiá-las. Na sua maioria estes sistemas são compostos de servidores de arquivos e ferramentas de software como editores gráficos e compiladores de máscaras, que ajudam os engenheiros nos projeto, desenvolvimento e teste de seus produtos. No que diz respeito à distribuição de tarefas e sua alocação aos projetistas, as aplicações de CAD geralmente apresentam uma estrutura hierárquica [1]. Normalmente, projetos mais amplos são sucessivamente subdivididos em conjuntos de subprojetos até que cada subprojeto alcance os graus de complexidade e independência esperados. Cada uma das tarefas resultantes

desta subdivisão é, então, executada por um pequeno grupo de projetistas (ex.: cinco a dez pessoas). Nestes grupos, cada projetista desenvolve uma parte do trabalho. Refletindo a organização hierárquica do projeto, projetistas alocados a grupos diferentes, normalmente, manipulam objetos de dado também diferentes. Na maioria dos casos, projetistas de grupos diferentes somente acessam dados comuns para fins de leitura (ex.: acesso a normas gerais de documentação de projeto) [10]. O mesmo tende a ocorrer quando projetistas de um grupo acessam dados criados por outro grupo (ex.: engenheiros projetando uma unidade de fita magnética necessitam de informações relativas às especificações de entrada e saída da memória principal do computador sendo projetado por outro grupo). Projetistas de um mesmo grupo, normalmente, manipulam objetos de dado comuns. Porém, ao contrário de usuários de aplicações convencionais que acessam dados de forma concorrente, projetistas de um mesmo grupo tendem a cooperar uns com os outros (ex.: construir partes distintas de uma mesma peça mecânica, usando uma interface comum). Sendo assim, transações de CAD pertencentes a um mesmo grupo de projetistas podem acessar dados criados (ou atualizados) por outras transações ainda ativas. Como consequência desta característica das aplicações de CAD, os sistemas de banco de dados que as suportam devem relaxar a propriedade de isolamento das transações. Contudo esta modificação do paradigma de transação deve ser feita de maneira controlada para evitar que transações executando em paralelo venham a comprometer a consistência do banco de dados de forma definitiva. Em [13], por exemplo, Korth propõe que o controle de concorrência do SBD seja baseado nos diferentes objetos da base de dados e não mais nas diferentes transações que executam em paralelo no sistema. Esta técnica de controle de concorrência permite que as transações liberem novas versões de objetos de dado mesmo antes de encerrarem sua execução. O único critério a ser seguido é o de que as novas versões não desrespeitem as restrições de integridade que garantem a consistência do banco de dados. No capítulo 5 deste artigo, veremos em mais detalhes como isso pode ser feito.

Geralmente, projetistas de um mesmo grupo trabalham em contiguidade física, isto é, geograficamente próximos uns dos outros (ex.: na mesma sala, no mesmo prédio). Portanto, estes projetistas podem facilmente entrar em contato e discutir problemas comuns. Além disso, são feitas reuniões periódicas para a solução destes problemas. A distribuição física de grupos de projetistas assim como a estrutura hierárquica das aplicações de CAD influenciam a configuração dos sistemas computacionais que as suportam e, consequentemente, a arquitetura dos sistemas de banco de dados que suportam ambientes de desenvolvimento de projetos.

3.2 O Sistema Computacional do Ambiente de Projeto

Os sistemas de computação mais modernos usados no suporte de aplicações de CAD são, geralmente, organizados como uma rede de unidades servidoras (de

dados e programas) e estações de trabalho. A figura 2 ilustra a arquitetura típica de um sistema moderno de CAD. As unidades de processamento são interligadas por uma rede de comunicação rápida (local area network) que permite a comunicação direta entre estas unidades.

As unidades de processamento ditas servidoras (servers), trabalham como gerenciadores de acesso a objetos de dado. Elas controlam tanto o armazenamento quanto o acesso aos dados de projeto e às ferramentas de software globais à aplicação. Para alguns modelos de processamento propostos para sistemas de CAD, estes objetos (dados e programas) constituem o banco de dados público (BDpu) da aplicação. As unidades servidoras podem, por sua vez, também ser implementadas como sistemas distribuídos, mas cada uma delas é tratada pelos outros nodos da rede como uma entidade única.

Figura 2: Arquitetura de um sistema de CAD moderno

Cada estação de trabalho (workstation) é alocada a um ou mais projetistas para o desenvolvimento de uma ou mais tarefas específicas. De um modo geral, estas estações de trabalho são, então, colocadas nos ambientes de trabalho dos projetistas. A construção das estações é baseada em processadores relativamente eficientes (ex.: microprocessadores de 32 bits). As estações são equipadas com memórias de vários megabytes e monitores de alta resolução para processamento gráfico. A grande maioria das estações de trabalho oferecidas no mercado já vem equipada com unidade de disco fixo própria. Estes discos são utilizados no armazenamento dos objetos de dado privados de cada projetista, isto é, dos dados sendo criados ou atualizados por transações ainda em andamento (dados não liberados a outros projetistas). Para alguns dos novos modelos de processamento que estão sendo propostos para representar as características dinâmicas das aplicações de CAD, os resultados temporários armazenados no disco da estação de trabalho representam o banco de dados privado (BDpr) do projetista. Como foi colocado acima, já existem vários sistemas de suporte a aplicações de CAD sendo oferecidos no mercado. A maioria deles, porém, é baseada em gerenciadores de arquivos comuns que não oferecem as vantagens apresentadas por sistemas de banco de dados, isto é, independência de dados,

controle de concorrência, recuperação da base de dados em casos de falha do sistema e independência de localização de dados. Outros sistemas de CAD são baseados em SBDs convencionais. Apesar de estes sistemas apresentarem as propriedades citadas acima, eles implementam modelos de dados (ex.: modelo hierárquico, modelo relacional) que nem sempre atendem bem às necessidades de modelagem e manipulação de dados apresentadas por aplicações de CAD. A maioria destas aplicações, lidam com objetos de dado muito mais complexos e estruturados do que aqueles passíveis de serem representados diretamente com base nos modelos de dados convencionais (ex.: peças mecânicas constituídas de muitas subpeças também complexas). Além disso, os SBDs convencionais implementam o conceito tradicional de transações, o qual nem sempre supre as necessidades de cooperação de transações presentes em ambientes de projeto. Por estes motivos, pesquisadores de sistemas de banco de dados em todo o mundo estão investigando novas tecnologias que melhor suportem ambientes de CAD. Estas novas tecnologias estão sendo integradas para formar os chamados sistemas de banco de dados não convencionais. Seja qual for o SBD sendo integrado ao ambiente de projeto, este deve, necessariamente, ser um sistema de banco de dados distribuído. As unidades servidoras são controladas pelo SBD público (SBDpu) que gerencia os objetos pertencentes ao projeto global sendo executado. Estes objetos são as ferramentas de projeto (programas de aplicação), versões de dados liberadas pelas transações que as criaram e informações de interesse comum dos subprojetos (ex.: normas de documentação, standards a serem seguidos, catálogos de peças compradas prontas). Cada estação de trabalho tem seu próprio sistema de banco de dados privado (SBDpr). Este sistema gerencia o acesso ao BDpr e controla o trabalho concorrente sendo executado na estação. Este trabalho concorrente é, normalmente, iniciado pelo projetista que atualmente ocupa a estação de trabalho. Através da rede de comunicações, os diversos sistemas de banco de dados podem copiar e/ou transferir objetos de um banco de dados a outro.

Os novos modelos de dados sendo propostos para SBDs não convencionais suportam a definição e manipulação de objetos de dado altamente estruturados (ex.: objetos complexos e moléculas como estes estão definidos em [8] e [2], respectivamente). SBDs para aplicações de CAD provavelmente usarão estes objetos como unidades de comunicação entre os diversos nodos da rede de computadores a ser implantada.

Ao contrário dos SBDs convencionais distribuídos, os sistemas de banco de dados para CAD não irão processar os dados exclusivamente nos nodos onde estes estão armazenados. Já que os projetistas podem precisar processar certos objetos por períodos de tempo longos (ex.: dias, semanas), o desempenho do SBD pode melhorar consideravelmente, se este transferir objetos das unidades servidoras para as estações de trabalho e permitir que estes objetos sejam processados nestas últimas.

Os SBDs privados deverão copiar objetos do banco de dados público para os bancos de dados privados através das chamadas operações de CHECKOUT.

Estas operações serão iniciadas nas estações de trabalho e executadas nas unidades servidoras pelo SBD público. Para realizar a operação inversa, isto é, copiar objetos dos bancos de dados privados para o público, os SBDs privados deverão iniciar operações de CHECKIN. Através das operações de CHECKOUT e CHECKIN, o SBDpu pode controlar o acesso concorrente a objetos do BDpu, mesmo que o processamento destes objetos não ocorra na unidade servidora.

4. Modelos de Processamento para Aplicações de CAD

Os modelos de processamento propostos para aplicações de CAD tentam capturar as principais características dinâmicas do ambiente real de projeto e usá-las para adaptar o conceito tradicional de transação às necessidades das atividades de projeto.

Vários modelos de processamento para aplicações de CAD tem sido propostos recentemente na literatura de banco de dados. Em [9], alguns dos modelos mais conhecidos são analisados e comparados entre si. A partir daí, estes modelos são classificados em três grupos distintos. As principais características de cada um destes grupos são apresentadas a seguir. A figura 3 relaciona as três classes de modelos de processamento à hierarquia de unidades de trabalho do ambiente de projeto. Estas unidades de trabalho podem ser usadas para representar diferentes tipos de transação no sistema de banco de dados. A hierarquia de transações resultante desta abstração pode ser realizada no SBD como uma transação aninhada. O conceito de transações aninhadas (nested transactions) é apresentado e formalmente tratado em [15]. A transação aninhada é uma extensão do conceito tradicional de transação e serve de base para a implementação correta de mecanismos de gerenciamento de transações em sistemas de banco de dados distribuídos.

Os modelos de processamento para aplicações de CAD deveriam representar tanto as atividades de projeto realizadas nas estações de trabalho quanto aquelas executadas nas unidades servidoras. Como nem todos os modelos propostos fazem isto na prática, o modo como o trabalho de projeto deve evoluir na estação de trabalho foi generalizado em [9] para todos os modelos estudados.

Pode-se imaginar que, independentemente da classe a que pertencem, todos os modelos representam o esforço do projetista na estação de trabalho da mesma maneira. O conjunto de atividades desenvolvidas pelo projetista em sua estação de trabalho é modelado como uma transação (aninhada), a qual recebe o nome de *transação de projeto* (design transaction). Esta transação é, normalmente, subdividida em um conjunto de transações menores (isto é, de menor duração), as quais se convencionou chamar de *transações de recuperação* (recovery transactions). A figura 4 localiza, na arquitetura do sistema de CAD, a estrutura em árvore formada pela transação de projeto e suas subtransações.

Nota: MGi lê-se modelo genérico i

Figura 3: Relacionando novos modelos de processamento de dados às unidades de trabalho das aplicações de CAD

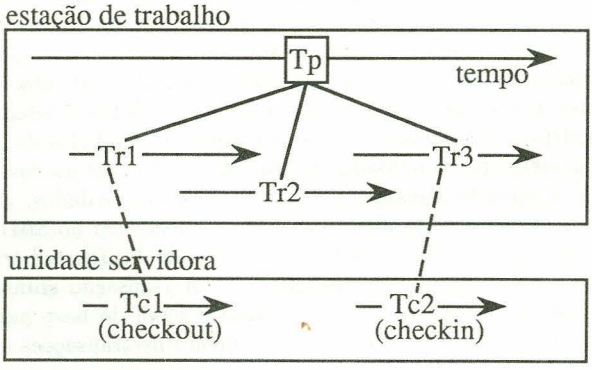


Figura 4: A estrutura em árvore da transação de projeto e suas subtransações

As transações de recuperação representam as várias atividades desenvolvidas pelo projetista durante a evolução de seu projeto individual. Na área de engenharia de software, exemplos de tais atividades seriam a edição de um módulo de programa, sua compilação e o posterior teste do módulo criado; na área de projeto de sistemas digitais, o desenho de um circuito eletrônico com a ajuda de um editor gráfico e a posterior simulação deste circuito, com base em uma ferramenta de software criada para tal fim. A transação de recuperação faz

parte de um esforço mais amplo do projetista que é representado pela transação de projeto. Dependendo da atividade sendo desenvolvida na estação de trabalho, duas ou mais transações de recuperação podem estar sendo executadas em paralelo. Como estas transações são de curta duração, o SBDpr pode tratá-las como transações convencionais, isto é, o sistema de banco de dados da estação de trabalho garante que cada transação de recuperação é executada atomicamente, de maneira isolada e que seus resultados persistem no BDpr mesmo em caso de falhas do sistema. Ao contrário do modelo tradicional de transação, porém, os efeitos de transações de recuperação que terminaram com sucesso são anulados no banco de dados privado, caso o projetista exija que sua transação de projeto seja desfeita (abortada). Além das características comuns relativas às atividades na estação de trabalho, todos os modelos de processamento para CAD representam as operações remotas (aquelas que aparecem tracejadas na figura 4 na forma de transações de curta duração (ou *transações curtas*). Por operações remotas, entende-se aquelas que são iniciadas na estação de trabalho e executadas nas unidades servidoras pelo SBDpu. Exemplos de tais operações são as atividades de CHECKOUT e CHECKIN. As transações curtas executadas nas unidades servidoras apresentam as mesmas propriedades das transações de recuperação que são processadas nas estações de trabalho. Os nomes diferentes servem somente para identificar onde cada transação é executada.

A seguir, as três classes de modelos de processamento indicadas na figura 3 são apresentadas em mais detalhe. Com isso, o leitor poderá identificar as principais diferenças entre elas.

4.1 As Atividades de Projeto no Modelo Genérico MG1

O primeiro modelo genérico de processamento para aplicações de CAD em sistemas de banco de dados é também o mais simples da classificação. MG1 representa modelos de processamento como os apresentados em [8] e em [17].

MG1 não modela completamente a transação de projeto na unidade servidora. Este modelo, representa o processamento (feito na mesma estação de trabalho) de objetos diferentes como conjuntos independentes de atividades. O SBDpu garante a propriedade de isolamento para cada conjunto de atividades em separado. Isto é, o processamento de cada objeto é tratado como uma transação de projeto independente. Sendo assim, com relação ao banco de dados público, o projetista não pode representar o processamento de dois objetos distintos (levados à estação de trabalho através de duas operações de CHECKOUT) como passos na execução de uma única transação de projeto.

No início de suas atividades, o projetista cria sua transação de projeto na estação de trabalho através do SBDpr. Quando esta transação executa sua primeira operação de CHECKOUT na unidade servidora, o SBDpu dá-se conta de sua existência e passa a controlá-la no que diz respeito a acessos que ela faz ao BDpu. O único objetivo do controle da transação de projeto na unidade servidora é o de gerenciar os locks (permissões de acesso a dados) associados

aos objetos copiados do BDpu para o BDpr da estação de trabalho. Estes locks são requisitados ao SBDpu pelas transações curtas que executam operações de CHECKOUT por ordem da transação de projeto. Por meio destes locks, o SBDpu regulamenta o acesso de outros projetistas aos objetos copiados para o BDpr. Os locks associados a operações de CHECKOUT são liberados na unidade servidora assim que as respectivas operações de CHECKIN são executadas no SBDpu (e os respectivos objetos são devolvidos ao BDpu). Nestas ocasiões, o sistema público atualiza o registro da transação de projeto mantido na unidade servidora. De acordo com MG1, a transação de projeto pode iniciar operações de CHECKOUT/CHECKIN a qualquer momento e em qualquer ordem (em se tratando de objetos diferentes). Portanto o sistema de banco de dados não pode assegurar a propriedade de isolamento para a transação de projeto como um todo. Em consequência disso, o sistema não pode garantir a serializabilidade [3] do conjunto de transações de projeto que acessam o BDpu de forma concorrente. Portanto, pelo menos parte da responsabilidade sobre a manutenção da consistência do BDpu é transferida para os projetistas.

Quando o projetista decide encerrar sua transação de projeto, o SBDpr deve consultar o SBDpu. Este último só permite o término da transação de projeto, se esta não está mais associada a quaisquer locks adquiridos durante operações de CHECKOUT. Isto é, MG1 garante que a transação de projeto só será encerrada quando todos os objetos copiados para o BDpr já tiverem sido devolvidos ao BDpu.

4.2 As Atividades de Projeto no Modelo Genérico MG2

Ao contrário de MG1, o modelo de processamento MG2 regulamenta as atividades de CHECKOUT e CHECKIN de maneira rígida para garantir o isolamento completo da transação de projeto. MG2 representa modelos de processamento como o proposto em [7]. MG2 impõe um protocolo de duas fases (strict two-phase locking [3]) para a transferência de objetos entre os bancos de dado público e privado. Durante a execução da transação de projeto, objetos podem ser copiados (através de operações de CHECKOUT) do BDpu para o BDpr, mas nenhum objeto pode ser devolvido ao banco de dados público na unidade servidora. Isto é, operações de CHECKIN não são permitidas durante a execução da transação de projeto. Sendo assim, nenhum outro projetista pode ter acesso aos resultados parciais produzidos pela transação de projeto em questão. No final da transação, todos os objetos copiados do BDpu para o BDpr devem ser copiados de volta ao banco de dados público de maneira atômica. Isto é, ou todos os objetos são transferidos para a unidade servidora corretamente, ou toda a operação de cópia (CHECKIN conjunto [7]) deve ser reiniciada. Enquanto o protocolo para operações de CHECKOUT e CHECKIN garante, no BDpu, o isolamento completo da transação de projeto em relação a outras transações do mesmo tipo, a operação de CHECKIN conjunto garante a atomicidade daquela transação em relação àquele banco de dados.

Além das propriedades descritas acima, MG2 introduz o conceito de pontos de retorno (save points) para tornar mais flexível a atividade do projetista na estação de trabalho. Conforme o trabalho sendo desenvolvido pelo projetista vai evoluindo, este pode definir pontos de retorno para onde a transação de projeto poderá retroceder, caso ele assim o deseje.

```
Transação2: "Edição de Subrotina"
Início_Transação
   CHECKOUT (objeto O1) (* algoritmo *)
   CHECKOUT (objeto O2) (* ADT pilha *)
   CHECKOUT (objeto O3) (* ADT fila *)
   Defina      (ponto de retorno = PR1)
   Crie        (objeto O4 = integra(O1,O2))
   Se          (O4 não é eficiente)
   então       início_faça
                  Estado_Transação = PR1
                  Crie (objeto O4 = integra (O1,O3))
               fim_faça
   CHECKIN     (objectos: O1, O2, O3, O4)
Fim_Transação
```

Figura 5: Exemplo de transação de projeto baseada no modelo MG2

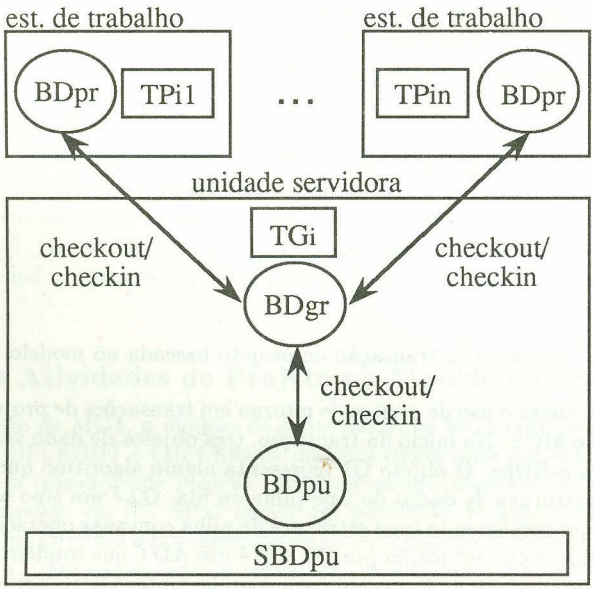
A figura 5 ilustra o uso de pontos de retorno em transações de projeto que seguem o modelo MG2. No início da transação, três objetos de dado são copiados do BDpu para o BDpr. O objeto O1 representa algum algoritmo que deve manipular uma estrutura de dados do tipo pilha ou fila. O2 é um tipo abstrato de dado (ADT) que implementa uma estrutura de pilha com suas operações básicas e pode ser usado como subrotina por O1. O3 é um ADT que implementa o tipo de dado fila e também pode ser usado para complementar O1. O projetista integra O2 a O1 e testa a eficiência do algoritmo resultante (O4). Se o desempenho de O4 não é aceitável, o projetista restaura o estado da transação de projeto para o ponto de retorno PR1 (o qual representa um estado da transação anterior à integração de O2 a O1) e constrói uma nova versão do objeto O4, usando agora os objetos O1 e O3. No final da transação, todos os objetos copiados do BDpu mais o que foi criado são transferidos para lá através de uma única operação de CHECKIN (que deve ser executada de forma atômica).

4.3 MG3: Uma Classe de Modelos para Representar as Atividades de Grupos de Projetistas

A classe MG3 engloba aqueles modelos de processamento que representam não só o esforço individual de um projetista mas todo o trabalho de um grupo de

projetistas cooperando no desenvolvimento de um projeto comum (ex.: [11], [12], [13] e [16]). Este trabalho de cooperação entre projetistas é modelado através do conceito de transação de grupo (group transaction).

A figura 6 apresenta a estrutura da transação de grupo e sua relação com as transações de projeto. Se imaginarmos que cada uma das transações de projeto ilustradas na figura 6 tem a estrutura mostrada na figura 4, então pode-se concluir que a transação de grupo complementa e, ao mesmo tempo, é a raiz da hierarquia de transações aninhadas que suportam e controlam o processamento de dados em sistemas de banco de dados voltados para aplicações de CAD.



TP -> transação de projeto
TG -> transação de grupo
SBDpu -> sist. de banco de dados público
BDpr -> banco de dados privado
BDgr -> banco de dados de grupo
BDpu -> banco de dados público

Figura 6: Estrutura de árvore da transação de grupo no modelo MG3

Além de introduzir um tipo novo de transação, os modelos da classe MG3 também introduzem um novo elemento na hierarquia de bancos de dados do SBD. Assim como cada transação de projeto está associada a um banco de dados privado, cada transação de grupo está associada a um banco de dados de grupo

(BDgr). Bancos de dado de grupo podem ser implementados como porções do banco de dados público. Como cada grupo de projetistas está associado a um, e somente um BDgr, estes bancos de dado delimitam a área do BDpu que cada grupo de projetistas pode acessar.

Com base na hierarquia de bancos de dados proposta, os modelos pertencentes à classe MG3 impõem o seguinte protocolo para acesso a dados em aplicações de CAD. Quando, por exemplo, uma transação de recuperação Tr1 necessita manipular um objeto de dado O1, ela consulta, através do SBDpr de sua estação de trabalho, se o objeto em questão encontra-se no banco de dados privado. Em caso positivo, Tr1 solicita ao SBDpr para acessar O1. De acordo com a situação de outras transações de recuperação que executam em paralelo com Tr1, o sistema de banco de dados privado vai conceder ou negar a esta transação o acesso a O1.

Se, ao contrário, O1 não estiver armazenado no BDpr da estação, então a transação de projeto Tp1, que controla Tr1, é informada e inicia uma operação de CHECKOUT(O1) no banco de dados do grupo ao qual pertence. Assumindo que BDgr é gerenciado pelo SBDpu na unidade servidora, a operação de CHECKOUT iniciada por Tp1 na estação de trabalho é transformada em uma transação curta (Tc1), a qual atua sobre o banco de dados do grupo que controla Tp1. Caso O1 está armazenado em BDgr e não está emprestado a outra transação de projeto do mesmo grupo (em modo conflitante de acesso), SBDpu reserva O1 para Tp1 e copia este objeto para o BDpr da estação onde Tp1 está sendo executada. Tp1, então, transfere o controle de acesso sobre O1 para Tr1.

Se O1 também não está armazenado no BDgr associado a Tp1, o SBDpu tenta copiar este objeto do banco de dados público para o banco de dados do grupo em questão. Isto só é possível, se O1 não estiver alocado a outro grupo de projetistas e, portanto, não estiver armazenado em outro banco de dados de grupo ao qual Tp1 não tem acesso. Com os conceitos de transação de grupo e banco de dados de grupo, os modelos de processamento da classe MG3 objetivam modelar a troca de resultados parciais entre transações de projeto concorrentes. Isto é, liberando (através de operações de CHECKIN) novos objetos de dado ou novas versões de objetos no banco de dados do grupo, ainda antes de encerrar sua transação de projeto, o projetista tem certeza de que somente os projetistas de seu grupo terão acesso a estes resultados que, de certa forma, ainda são temporários (quem pode garantir neste momento que a transação que está liberando os objetos não irá abortar no futuro?). Objetos novos ou atualizados só são transferidos do BDgr para o banco de dados público quando o grupo de projetistas chega a conclusão de que estes objetos mantém a consistência do BDpu e suas versões estão suficientemente maduras para serem acessadas por outros grupos.

A classe de modelos MG3 permite que transações de projeto de um mesmo grupo troquem resultados parciais através do banco de dados do grupo. Similar a MG1 e oposta a MG2, a classe MG3 permite que as transações de projeto executem operações de CHECKOUT e CHECKIN a qualquer momento e em qual-

quer ordem. O que diferencia as transações em MG3 daquelas em MG1 é que as primeiras executam estas operações no banco de dados de grupo enquanto as últimas o fazem diretamente no banco de dados público. Uma única restrição imposta por alguns modelos da classe MG3 (ex.: [13]) à transação de projeto é a de que esta transação não pode copiar um objeto do BDgr para o BDpr depois de já tê-lo devolvido ao BDgr uma vez. Isto garante a serializabilidade do processamento ao nível dos objetos de dado.

Como em MG2, o projetista também pode gerar pontos de retorno (save points) para a sua transação de projeto em alguns modelos da classe MG3 (ex.: [16]). O projetista pode, então, descartar opções de projeto de duas maneiras: retrocedendo o estado da transação de projeto para um ponto de retorno previamente fixado ou abortando a transação por completo.

5. Gerenciando a execução de Transações de Grupo no Banco de Dados

A certeza de que sistemas de banco de dados, que suportam o conceito tradicional de transações, garantem a integridade e a consistência do banco de dados repousa nas propriedades de isolamento, atomicidade e persistência que o paradigma da transação apresenta. Para atender a requisitos importantes das aplicações de CAD, os modelos de processamento da classe MG3 relaxaram, porém, algumas destas propriedades. Com o objetivo de permitir uma maior cooperação entre projetistas, os modelos da classe MG3 permitem que as transações de projeto de um mesmo grupo possam acessar objetos criados ou atualizados por transações ainda ativas. Alguns destes modelos de processamento vão ainda mais longe e permitem que as transações de projeto executem de forma não atômica (ex.: [16]). Isto é, os objetos de dado liberados no BDgr através de operações de CHECKIN não são, necessariamente, invalidados caso a transação que os criou seja abortada por seu projetista.

Ao suportar o isolamento na execução de transações, os SBDs convencionais garantem a serializabilidade (serializability) da execução do conjunto de transações que é processado ao longo do tempo. O processamento deste conjunto de transações, ao qual dá-se o nome de *schedule*, é dito "serializável" quando as transações que o compõem produzem, ainda que executadas concorrentemente, os mesmos resultados que produziriam se fossem executadas seqüencialmente (em alguma ordem). Ao relaxar a propriedade de isolamento das transações de projeto, os modelos da classe MG3 abrem mão da serializabilidade como critério de correção que garante a consistência do banco de dados em um ambiente de execução de transações concorrentes.

Em [13], a serializabilidade da transformação de objetos da base de dados é proposta para substituir a serializabilidade do schedule como critério de correção em sistemas de banco de dados que suportam transações de grupo.

Para entender este novo critério de correçao, é necessário aceitar que a consistência do banco de dados (público) possa ser expressa por uma única (e abrangente) restrição de integridade **I**. Esta restrição de integridade (que nada mais é do que um predicado sobre o estado do banco de dados) poderia, então, ser expressa de forma conjuntiva, isto é, como um conjunto de subexpressões lógicas, onde a estrutura interna de cada subexpressão não apresenta nenhuma ocorrência do conetivo & (and) e onde estas subexpressões estão ligadas entre si, exclusivamente, por conetivos deste tipo. Subexpressões que seguem esta regra são denominadas conjunções.

Se a restrição de integridade **I** do banco de dados for detalhada a tal ponto que cada conjunção da expressão resultante estiver associada a um, e somente um, objeto do banco de dados, então uma operação sobre um objeto Oi qualquer estará correta se, e somente se, a conjunção Ci associada a este objeto for satisfeita (mantida verdadeira) pela operação que manipula Oi. Diz-se, então, que Ci é a invariante que deve ser respeitada pelas operações que atualizam Oi para que a consistência de Oi seja mantida. Se o SBD permite a execução somente daquelas operações que garantem as invariantes associadas aos objetos que manipulam, este estará, indiretamente, garantindo toda a restrição de integridade **I** e, com isto, a consistência de todo o banco de dados.

Ao contrário do conceito de serializabilidade do schedule, este novo critério de correção do SBD não está baseado na execução de transações completas, mas na execução de grupos de operações dentro da transação que manipulam o mesmo objeto. Sendo assim, o critério de serializabilidade na transformação de objetos suporta o relaxamento da propriedade de isolamento das transações, desde que o SBD garanta que transações ativas só possam liberar objetos novos ou atualizados que satisfaçam o critério de consistência do banco de dados que é representado pela restrição de integridade **I**.

5.1 Suporte à Atomicidade e a Pontos de Retorno em Transações de Grupo

Supondo que o SBD realiza um modelo de processamento do tipo MG3 e controla a consistência do banco de dados com base no critério de serializabilidade na transformação de objetos, pergunta-se como o SBD vai suportar a atomicidade e os pontos de retorno de transações de projeto que liberam resultados no banco de dados de grupo, ainda antes de encerrarem sua execução.

Em um ambiente em que as transações apresentam a propriedade de isolamento, fica fácil para o sistema de banco de dados retroceder o estado de uma transação para um ponto de retorno previamente definido ou para seu início (caso a transação seja abortada). O SBD só precisa se preocupar em desfazer modificações causadas no banco de dados pela transação sendo retrocedida ou abortada. Mas como fica o caso em que outras transações já acessaram um ou mais objetos modificados pela transação sendo retrocedida?

A seguir, apresentamos uma técnica para a gerência de transações de grupo

que suporta a definição e o uso de pontos de retorno em transações de projeto assim como transfere ao projetista a decisão de manter ou não a atomicidade de sua transação. Por questões de espaço, aqui só será fornecida uma visão geral desta técnica. Em [9], o leitor interessado encontra sua descrição completa. As transações de grupo como um todo são gerenciadas de forma convencional, isto é, o SBDpu garante as propriedades de isolamento, atomicidade e persistência para as transações de grupo e controla seu acesso concorrente ao BDpu através de um mecanismo de bloqueio (locking) para operações de CHECKOUT. A aquisição de locks pelas transações de grupo segue um protocolo de duas fases semelhante àquele apresentado para as transações de projeto da classe de modelos MG2.

Dentro de cada transação de grupo, as transações de projeto são gerenciadas de outra forma. A execução destas transações é controlada com base em um grafo direcionado (G) que, a cada momento, descreve o estado de cada transação e das comunicações existentes entre elas. Cada transação de projeto (Tp_i) da transação de grupo (Tg) é representada como um nodo em G. Tp_i é inserida em G no momento em que é iniciada por seu projetista e retirada do grafo quando termina com sucesso ou quando é abortada.

Os arcos de G representam as atividades de CHECKOUT e CHECKIN sobre o BDgr e indicam a direção das trocas de objetos entre as diversas transações de projeto que executam em paralelo (e/ou concorrentemente). Uma troca de objeto representa um par de operações. A transação que criou ou atualizou o objeto a ser trocado transfere-o para o BDgr através de uma operação de CHECKIN. A transação que necessita deste objeto copia-o do BDgr para seu banco de dados privado através de uma operação de CHECKOUT. Cada operação de troca de objeto pode, portanto, ser representada por uma tripla da forma (Tp_x, Tp_y, Oi_v). Tp_x identifica a transação que criou ou atualizou o objeto Oi que está sendo transferido. Tp_y identifica a transação que está recebendo este objeto. O índice v do objeto Oi indica a versão em que se encontra Oi no momento da transferência.

Na realidade, o arco que liga o nodo Tp_x ao nodo Tp_y em G representa o conjunto de todas as trocas de objetos da transação Tp_x para a transação Tp_y, e pode ser representado pela tripla (x, y, s_{xy}) onde s_{xy} é um conjunto de identificadores do tipo Oi_v (com $1 \leq i \leq n$ e $1 \leq v \leq m$).

A figura 7 mostra um possível estado do grafo G durante a execução de uma transação de grupo Tg que engloba três transações de projeto (Tp_1, Tp_2 e Tp_3). Supondo que o estado inicial do banco de dados de grupo é BDgr=O_{11}, O_{21}, O_{31}, a transação Tp_1 copia O_{11} para seu BDpr, atualiza-o e insere a versão O_{12} em BDgr por meio de um CHECKIN. Tp_3 copia O_{12} para seu BDpr e passa a processá-lo. Ao mesmo tempo, Tp_2 cria a versão O_{22} a partir de O_{21}, libera-a no banco de dados de grupo e define um ponto de retorno (PR_{tp2}) para si mesma. Depois disso, Tp_2 executa uma operação de CHECKOUT em O_{31} e passa a atualizá-lo. Após Tp_2 ter inserido a versão O_{32} no BDgr, Tp_3 copia-a para seu BDpr e passa a processá-la.

Analisando o estado de Tg descrito acima, chega-se a conclusão de que a

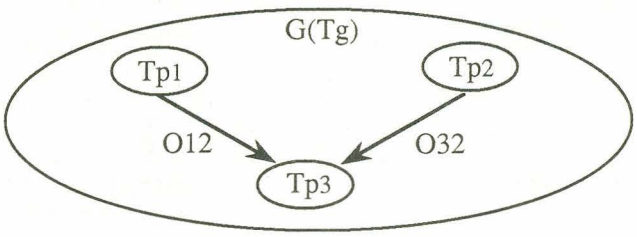


Figura 7: Possível estado do grafo G que representa a execução da transação Tg

continuação da execução de Tp_3 depende intimamente das execuções de Tp_1 e Tp_2. Se o SBD garante a atomicidade de transações de projeto, então Tp_3 terá que ser abortada pelo sistema caso Tp_1 ou Tp_2 aborte. O mesmo acontecerá se Tp_2 retroceder ao ponto de retorno PR_{tp2}. Nestes caso, todo o trabalho desenvolvido pelo projetista responsável por Tp_3 seria perdido. Existe, porém, uma situação ainda mais complicada. O que aconteceria se Tp_3 fosse encerrada (com sucesso) por seu projetista ainda antes de Tp_1 ou Tp_2 resolver abortar ou Tp_2 decidir retroceder a PR_{tp2}?

A técnica de gerência de transações de grupo proposta em [9] resolve estes e outros casos, envolvendo a execução de duas ou mais transações de projeto, a partir da análise do grafo G e através da introdução de um novo estado possível para estas últimas transações. A figura 8 ilustra os estados possíveis de serem assumidos por transações de projeto no ambiente gerenciado pela nova técnica. Logo que a transação é iniciada, ela entra no estado ativo. No caso de a transação ser de longa duração, o projetista pode suspendê-la (ex.: para ir almoçar) e reativá-la tantas vezes quantas achar necessário. O projetista pode também decidir abortar sua transação. Ele só não pode encerrá-la com sucesso diretamente (isto é, em um passo). Ao invés disso, o projetista declara sua transação como pronta a encerrar (ready) e passa, ao SBDpu, o controle sobre ela. Somente após o SBDpu ter encerrado, definitivamente, todas as transações que geraram algum objeto acessado pela transação pronta para encerrar é que esta será realmente encerrada. Impondo este diagrama de transição de estados às transações de projeto de um mesmo grupo, o SBDpu pode garantir a recuperabilidade do banco de dados de grupo com base em técnicas de "recovery" que suportam o fenômeno conhecido como "cascading aborts". Detalhes sobre tais técnicas são encontrados em [3].

Quando alguma transação de projeto retrocede para algum ponto de retorno ou é abortada, o SBDpu analisa o grafo G para identificar aquelas transações que, direta ou indiretamente, acessaram objetos gerados pela transação em questão. Cada uma destas transações (ou dos projetistas que as controlam) tem, no mínimo, três alternativas a seguir. Ou a transação é abortada ou ela

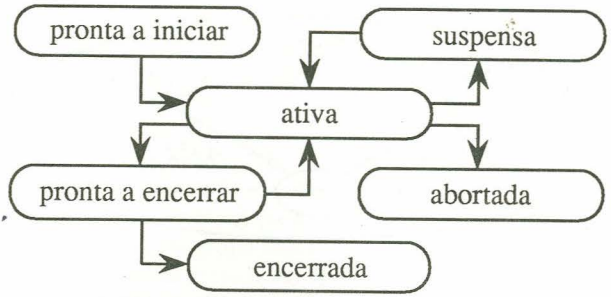


Figura 8: Conjunto de estados e transições possíveis para transações de projeto em MG3

despreza aquela parte de seu trabalho que está relacionada aos objetos herdados da transação que está abortando ou retrocedendo. Uma terceira alternativa seria a transação assumir, como suas, as versões de objetos que recebeu da transação que está abortando ou retrocedendo.

6. Conclusão

O presente artigo descreve novas técnicas de gerênciamento de transações em sistemas de banco de dados para aplicações de CAD. Para justificar a necessidade e a importância de tais técnicas, o artigo contrasta as principais características de sistemas de banco de dados para aplicações comerciais com os requisitos de processamento de dados das aplicações da área de engenharia, mais precisamente, das aplicações de CAD.

Após descrever três classes distintas de modelos de processamento para sistemas de banco de dados que visam suportar aplicações da área de engenharia, o artigo propõe uma nova técnica para o controle de concorrência e "recovery" de transações de projeto que executam em ambiente cooperativo. Este tipo de ambiente, modelado através da transação de grupo, representa o esforço de um grupo de projetistas no desenvolvimento de um projeto comum. As transações de grupo permitem que projetistas desenvolvendo subprojetos distintos cooperem entre si, através da troca de resultados parciais de seu trabalho. A nova técnica evita que esta cooperação prejudique a consistência do banco de dados enquanto protege o trabalho desenvolvido em conjunto pelos diversos projetistas que participam da transação de grupo.

Referências

[1] BANCILHON, F.; KIM, W.; KORTH, H. F.: A model of CAD transacti-

ons. In: 11th Int. Conf. on Very Large Data Bases, Stockholm, 1985.

[2] BATORY, D. S.; BUCHMANN, A.P.: Molecular objects, abstract data types, and data models: a framework. In: 10th Int. Conf. on Very Large Data Bases, Singapore, 1984.

[3] BERNSTEIN, P. A.; HADZILACOS, V.; GOODMAN, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley Series in Computer Sciences, Addison-Wesley Publishing Company, 1987.

[4] BRODIE, M. L.; MANOLA, F.: Database management: a survey. In: Fundamentals of Knowledge Base Management Systems, J.W. Schmidt and C. Thanos (editores), Springer Verlag, 1988.

[5] EFFELSBERG, W.; HÄRDER, T.: Principles of Database Buffer Management. ACM Transactions on Database Systems, Vol. 9, No. 4, 1984.

[6] HÄRDER, T.; REUTER, A.: Principles of Transaction-Oriented Database Recovery. ACM Computing Surveys, Vol. 15, No. 4, dezembro 1983.

[7] HÄRDER, T.; HÜBEL, Ch.; MAYER-WEGENER, K.; MITSCHANG, B.: Processing and Transaction Concepts for Cooperation of Engineering Workstations and a Database Server. In: Data & Knowledge Engineering, North Holland, No. 3, 1988.

[8] HASKIN, R. L.; LORIE, R. A.: On extending the functions of a relational database system. Research Report RJ3182, IBM Research Laboratory, San José (CA), E.U.A. 1981.

[9] IOCHPE, C.: Database recovery in the design environment: requirements analysis and performance evaluation. Tese de doutorado, Faculdade de Informática, Universidade de Karlsruhe, República Federal da Alemanha, dezembro 1989 (em inglês).

[10] KELTER, U.: Transaction concepts for non-standard database systems. Informationstechnik, No. 30, janeiro 1988 (em alemão).

[11] KIM, W.; LORIE, R. A.; McNABB, D.; PLOUFFE, W.: A Transaction Mechanism for Engineering Design Databases. Proc. 1oth Int. Conf. on Very Large Data Bases, Singapore, 1984.

[12] KLAHOLD, P.; SCHLAGETER, G.; UNLAND, R.; WILKES, W.: A transaction model supporting complex applications in integrated information systems. In: ACM SIGMOD Int. Conf. on Management of Data, Austin (Texas), E.U.A., 1985.

[13] KORTH, H. F.; KIM, W.; BANCILHON, F.: On long-duration CAD transactions. Information Sciences, 1987.

[14] LORIE, R. A.; PLOUFFE, W.: Complex objects and their use in design transactions. In: Engineering Design Applications, Proc. of the Annual Meeting: Database Week, San Jose (CA), E.U.A., maio 1983.

[15] MOSS, J. E.: Nested transactions: an approach to reliable distributed computing. Ph.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, 1981.

[16] OLIVEIRA NETO, M. M. de: Gerência de Transações Longas no Sistema AMPLO. Trabalho Individual, CPGCC-UFRGS, 1989.

[17] REHM, S. et al.: Support for Design Process in a Structurally Object-Oriented Database System. In: Proc. 2nd Int. Workshop on Object-Oriented Database Systems, Bad Münster am Stein-Eberburg (Alemanha), setembro 1988.

[18] STONEBRAKER, M.; ROWE, L. A.: The Design of POSTGRES. ACM SIGMOD Int. Conf. on Management of Data, Washington, maio 1986.

Curriculum vitae (Cirano Iochpe)

Doutor em Informática pela Universidade de Karlsruhe, Alemanha, 1989. Professor Adjunto do Instituto de Informática e Professor Orientador do CPGCC, ambos na UFRGS. Áreas de interesse: implementação de sistemas de banco de dados (SBDs), modelos de transação em SBDs (convencionais ou não), SBDs para aplicações de engenharia e indústria, investigação sobre cooperação em atividades de projeto e produção.

Endereço para contato:
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Caixa Postal 1501
90001 – Porto Alegre –RS
Brasil
E-mail: iochpe@sbu.ufrgs.anrs.br

Artigo submetido em outubro 1990. Aceito na forma final em março 1991. Convertido e formatado para o estilo da Revista por Raul F. Weber a partir do texto fornecido em MS Word (versão Macintosh) pelo autor. Figuras fornecidas pelo autor em MacDraw II.