SIDNEY VICENTE MENDES

UMA METODOLOGIA PARA DESENVOLVIMENTO DE AUDITORIA INDEPENDENTE DE SISTEMAS DE INFORMAÇÃO

FLORIANÓPOLIS – SC

2003

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

SIDNEY VICENTE MENDES

UMA METODOLOGIA PARA DESENVOLVIMENTO DE AUDITORIA INDEPENDENTE DE SISTEMAS DE INFORMAÇÃO

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Murilo S. de Camargo
(Orientador)

Florianópolis, 5 de Fevereiro de 2003

UMA METODOLOGIA PARA DESENVOLVIMENTO DE AUDITORIA INDEPENDENTE DE SISTEMAS DE INFORMAÇÃO

SIDNEY VICENTE MENDES

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação em Sistemas da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação

_____

Prof. Dr. Fernando A. Ostuni Gauthier
(Coordenador do Curso)

Banca Examinadora:

_____

Prof. Dr. Murilo S. de Camargo (Orientador)

_____

Prof. Dr. Rosvelter João Coelho da Costa

_____

Prof. Dr. Vitório Bruno Mazzola

À minha filha Isabela

AGRADECIMENTOS

Nos últimos meses algumas batalhas foram travadas em meu universo pessoal. Essa empreitada foi uma delas. Somente com muita fé em Deus e o apoio de pessoas amigas pude superar tudo isso. Por isso quero agradecer primeiramente a Deus, pois somente Ele poderia me dar forças que considerava não possuir para lutar em tantas frentes.

Agradeço à minha mãe, que em sua pouca instrução, mas grande sabedoria, soube me incentivar para o estudo, e espero repassar isso para minha filha.

Agradeço também às pessoas que se mostraram realmente amigas e estiveram sempre presentes me incentivando. Agradeço a Fausto e Sônia, Rosângela e Osório, Rosana, Marlene e Beto.

Agradeço aos companheiros de mestrado Josué, Gilson e Guilherme, que souberam compartilhar suas motivações durante todo o curso. Agradeço ao meu orientador Prof. Murilo, e aos Profs. Ricardo e Vitório que se mostraram interessados no meu sucesso.

RESUMO

A informação assumiu importância vital para a manutenção dos negócios devido ao dinamismo da economia globalizada e *on-line*. O comprometimento do sistema de informações, por problemas de segurança, pode causar grandes prejuízos à organização. Diversos tipos de incidentes podem ocorrer a qualquer momento, podendo atingir a informação em sua confidencialidade, integridade e disponibilidade. A auditoria serve para avaliar os controles internos em um sistema de informação automatizado e verificar as fases e resultados do processamento de sistemas. Incompreensivelmente, a grande maioria dos sistemas de informação existentes tem sido implementada sem provisões para auditorias futuras.

É proposta uma metodologia que permita explorar a auditoria de sistemas de informação por uma abordagem independente da aplicação, registrando evidências sobre as ações da aplicação sobre os dados armazenados na base de dados relacional, sem que nenhum esforço durante as etapas de projeto e desenvolvimento do sistema sejam direcionadas para a disposição de um ambiente de auditoria de sistemas.

A metodologia especifica a origem, a modelagem das estruturas, e o tratamento dos dados, provendo uma base de dados para consultas futuras pelo auditor de sistemas, e permitindo a detecção de abusos cometidos sobre os sistemas de informação.

Serão também estabelecidos comparativos entre essa e outras abordagens tradicionais, amplamente utilizadas pelas *software houses*, caracterizando potenciais vantagens e mesmo desvantagens entre elas.

ABSTRACT

Information has got vital importance for business survival, due to the dynamism of global and on-line economy. Information systems damages, due to security faults, can cause large losses to the organization. Many kinds of incidents can occur at any time, affecting information in its confidentiality, integrity and availability. Auditing evaluates internal controls in computer-based information systems and verifies systems processing phases and results. Incomprehensibly, most existent information systems have been developed without provisions for future auditing.

A methodology is proposed here to permit the exploration of information systems auditing by an independent application approach, recording evidences of the application actions over the data stored in the relational database, with no effort during design and development stages directed for providing a systems auditing environment.

The methodology specifies origin, structure modeling, and treatment of the data, providing a database for future system auditor queries, and permitting information systems misuse detection.

Comparisons will be established between this and other traditional auditing approaches, broadly applied by the software houses, identifying potential advantages and even disadvantages between them.

SUMÁRIO

ÍNDICE DE FIGURAS

1 INTRODUÇÃO

A crescente utilização de soluções informatizadas nas diversas áreas de serviços exige níveis de segurança adequados à maior exposição dos valores e informações. A evolução da tecnologia de informação, migrando de um ambiente centralizado para um ambiente distribuído, interligando redes internas e externas, somada a revolução da Internet, mudaram a forma de se fazer negócios. Isso fez com que as empresas se preocupassem com quem pode ter acesso às suas informações e como elas devem ficar protegidas dos ataques internos e externos.

A segurança da informação tornou-se estratégica, pois interfere na capacidade das organizações de realizarem negócios e no valor de seus produtos no mercado [6]. Em tempos de economia nervosa e racionalização de investimentos, a utilização de recursos deve estar focada naquilo que mais agrega valor ao negócio [2].

Visando minimizar essas ameaças, a ISO (*International Standardization Organization*), e a ABNT (Associação Brasileira de Normas Técnicas), em sintonia com a ISO, publicaram uma norma internacional para garantir a segurança das informações nas empresas. Mesmo antes da publicação do documento oficial, as empresas brasileiras se anteciparam no sentido de preparar suas estruturas para implementação dos itens que compõem a norma. Isso se justifica pelo fato das empresas estarem mudando seu comportamento quanto à questão da segurança da informação, pressionadas pelo risco crescente de roubos e ataques a seus sistemas. As normas ISO e ABNT são o resultado de um esforço internacional que consumiu anos de pesquisa e desenvolvimento para se obter um modelo de segurança eficiente e universal [18].

Segurança da informação, conforme a norma ISO/IEC 17799:2000, é a proteção contra um grande número de ameaças às informações, objetivando assegurar a continuidade do negócio, minimizando danos comerciais e maximizando o retorno de investimentos e oportunidades. Caracteriza-se por preservar:

- Confidencialidade: segurança de que a informação pode ser acessada apenas por quem tem autorização;
- Integridade: certeza da precisão e do completismo da informação;

- Disponibilidade: garantia de que os usuários autorizados tenham acesso à informação e aos recursos associados, quando necessário;

As empresas estão constantemente expostas às ameaças e diversos tipos de incidentes podem ocorrer a qualquer momento, podendo atingir a informação em sua confidencialidade, integridade e disponibilidade [2]. A preservação desses atributos constitui o paradigma básico da norma internacional e de toda a ciência da Segurança da Informação [18].

Problemas de quebra de confidencialidade, por vazamento ou roubo de informações sigilosas, podem expor para o mercado ou concorrência as estratégias ou tecnologias da organização, eliminando um diferencial competitivo, comprometendo mercado e margens de lucro.

Problemas de disponibilidade podem impactar diretamente sobre o faturamento, deixar a organização sem matéria-prima ou suprimentos importantes ou impedir de honrar compromissos com clientes, prejudicando a imagem perante eles e gerando problemas com custos. Novamente a margem de lucro fica comprometida.

Problemas de integridade, devidos à invasão ou causas técnicas, sobre dados sensíveis, sem a sua imediata percepção, impactam sobre as tomadas de decisões. Decisões erradas fatalmente reduzirão faturamento ou aumentarão custos, afetando novamente a margem de lucros.

A invasão do *site* da empresa, com modificação de conteúdo, revela a negligência com a segurança da informação e a vulnerabilidade à perda de rentabilidade aos investidores. As ações da empresa perdem valor, e os acionistas, dinheiro.

Conclui-se que elementos fundamentais para a sobrevivência das empresas estão relacionados com segurança de informação, a qual contribui grandemente para a sua lucratividade e avaliação, ou seja, agrega valor ao negócio e garante o retorno sobre o investimento que o acionista deseja.

Entre as decisões importantes que devem ser tomadas, surge um novo fator para as empresas: a gestão segura da informação, que apesar de sempre presente, não era devidamente conhecida, entendida e praticada. Isso significa levar em conta muitas variáveis, como o próprio *core business*, o risco relacionado aos seus processos de

negócio, o fator humano e a organização, o orçamento, os custos diretos e indiretos, a aplicação de uma política de segurança, ou seja, como tornar tudo isto eficaz dentro e fora da organização. O negócio, que é o motivo de existência da empresa, corre riscos iminentes estando vulnerável e suscetível à indisponibilidade de serviços, perda de confidencialidade das informações e integridade dos dados [16].

A necessidade evidente da implantação de políticas de segurança não pode ser efetivada da noite para o dia. Além de decidir por um modelo de negócios adequado às vulnerabilidades dos sistemas, as corporações precisam saber e entender o que querem e o que é vital para o *core business*. Um plano diretor de segurança deve ser dinâmico e flexível para suportar as necessidades que surgem em virtude da velocidade com que fatores físicos, tecnológicos e humanos mudam [13].

Tradicionalmente, as organizações dedicam grande atenção para com seus ativos tangíveis físicos e financeiros, mas relativamente pouca atenção aos ativos de informação. Elas devem preocupar-se com o trato de certas informações, especialmente as de valor estratégico e comercial [18]. A adoção de políticas de segurança é prática, principalmente entre as grandes empresas [14]. Elas representam um conjunto formado por diretrizes, normas, procedimentos e instruções que irá nortear os usuários quanto ao uso adequado dos recursos a eles disponibilizados, e definir regras, comportamentos, proibições e até punições por má utilização [17].

A gestão de segurança de uma organização deve se preocupar com problemas criados por um estranho em sua rede, mas não deve ignorar as ações de seus usuários internos [6].

Em pesquisa realizada por INFORMATIONWEEK em 1999, com o apoio técnico da empresa de consultoria e auditoria PricewaterhouseCoopers, em 49 países e com respostas dadas por 2,7 mil profissionais ligados à área de tecnologia de informações, identificou-se como principal ameaça à segurança dos dados das empresas os funcionários e ex-funcionários, responsáveis por pelo menos 73% dos ataques aos sistemas [5].

Em 2001, a Módulo Security Solutions S.A. fez uma análise do cenário de segurança no Brasil. Esse estudo é considerado referência e principal fonte de consulta sobre o assunto no país. Aproximadamente 53% das empresas apontaram os

funcionários insatisfeitos como a maior ameaça à segurança da informação (figura 1.1), e 35% das empresas que sofreram invasões reconhecem os empregados como os principais responsáveis [1].

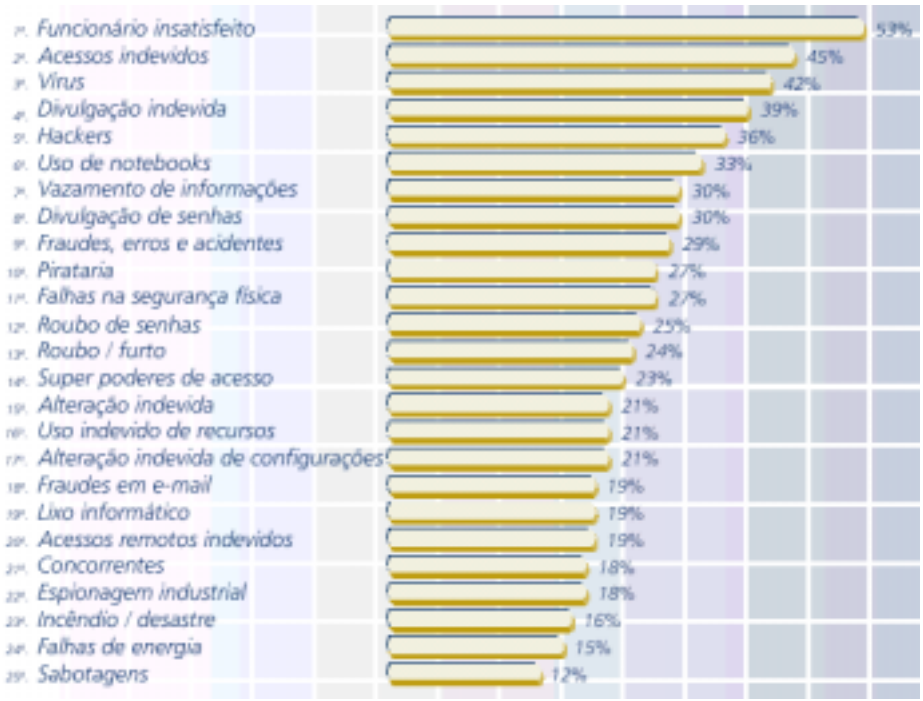


Figura 1.1: Principais ameaças às informações da empresa

Nas empresas que conseguiram identificar as causas dos problemas de segurança, verificou-se em 43% dos casos que os responsáveis estavam dentro da empresa (figura 1.2) [1].

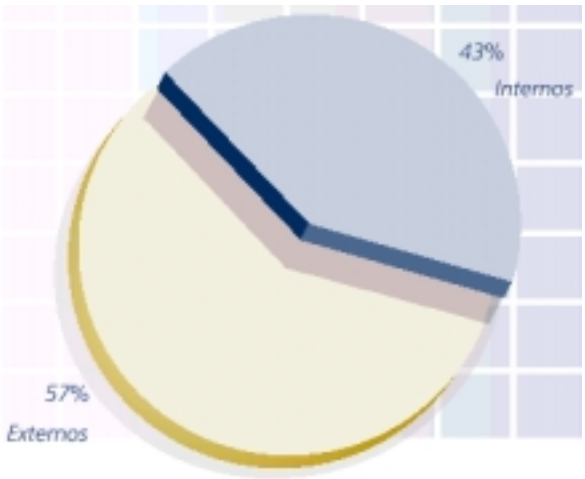


Figura 1.2: Responsáveis por problemas de segurança registrados

Apesar de “problemas internos” ser considerada a principal ameaça, faltam investimentos em capacitação e treinamento, além de procedimentos de classificação de segurança [1].

A definição de segurança de informação não inclui somente proteção de fontes internas ou externas de abuso intencional ou danos ao sistema, mas também a integridade do sistema em si, garantindo que o sistema faça o que se espera, e como é esperado, e que suas entradas e saídas sejam válidas e precisas com respeito a tudo. A auditoria do sistema de aplicação provê essa garantia [12].

A necessidade de procedimentos de auditoria completos pode não ser prontamente aparente para alguém que assuma que, uma vez que o sistema de informação tenha sido completamente testado, ele desde então funcionará perfeitamente. Isso é o ideal, porém, infelizmente, não é a realidade. Nenhum sistema é 100% seguro, e todo sistema tem mudanças de manutenção de programas. A auditoria serve para avaliar os controles internos em um sistema de informação automatizado e verifica as fases e resultados do processamento de sistemas. Ela detecta e, espera-se, elimina erros de processamento, sejam intencionais, acidentais, mecânicos ou humanos. Os objetivos da auditoria são identificar as vulnerabilidades do sistema, onde exista, avaliar os controles internos, verificar a implementação, e prover conclusões para a administração [12].

É importante não assumir que o sistema de segurança é perfeito. Em situações em que os dados sejam suficientemente sensíveis, ou onde o processamento executado sobre os dados seja suficientemente crítico, um sistema de auditoria é uma necessidade. Se discrepâncias das informações levam a suspeitas de adulterações, pode-se examinar o que ocorreu e verificar que tudo esteja sob controle – ou se não estiver, apontar o responsável [3].

Nas últimas décadas, o rápido crescimento e o avanço da tecnologia, com processamento distribuído e sistemas de banco de dados, criou-se um enorme desafio para a profissão de auditoria. Para lidar com a complexidade natural dos controles nos sistemas mais sofisticados, pode ser necessário descartar procedimentos tradicionais e desenvolver técnicas aperfeiçoadas [11].

A maioria das corporações há muito tempo não possui mais equipes de desenvolvimento de aplicativos em seus departamentos de T.I., optando por comprar sistemas de terceiros. Os sistemas de gestão de negócios (ERP) dos grandes fornecedores estão preparados para fornecer informações sobre as operações realizadas pelos usuários. As abordagens utilizadas para suprir tais informações variam e o nível de detalhamento também. O crescente papel central desempenhado pelos grandes sistemas de *software* na sociedade moderna levanta a questão de sua confiabilidade. Por que alguém deveria confiar em um sistema que ninguém consegue compreender completamente e que tende a evoluir? Mesmo uma pequena mudança evolutiva na estrutura do *software* pode produzir mudanças drásticas em seu comportamento [4].

Porém, incompreensivelmente, a grande maioria dos sistemas de informação existentes tem sido implementada sem provisões para auditorias futuras. Isso não implica que esses sistemas não possam ser auditados, mas particularmente que a auditoria não será tão efetiva como teria sido se os padrões de auditoria tivessem sido aplicados [12].

Surge dentro das empresas uma preocupação maior das pessoas envolvidas, no sentido de quais serão as diretrizes de segurança para alcançar, coletar, acessar, trocar, manipular um de seus principais patrimônios: as informações que deverão circular na instituição. Com isso pretende-se atingir objetivos de redução de riscos de vazamento, fraudes, erros, uso indevido, sabotagem, roubo de informações e outros problemas [18].

Situações como a informação incorreta de parâmetros para processamento pelo usuário e a ocultação desse fato quando percebido, geram suspeitas indevidas sobre a qualidade do sistema de informação e investigações penosas, já que se estará procurando por um erro que não existe. Também questões sobre autoria e genuinidade de operações suspeitas necessitam ser analisadas sob a luz de evidências concretas.

As corporações não podem ficar sem a capacidade de identificar o que se passa com suas informações, devido ao risco que isso pode acarretar. De alguma forma elas devem obter a resposta que lhes gere a certeza sobre a integridade de suas informações, e conseqüentemente, que oriente seus negócios [10]. Se o sistema de informação não foi implementado prevendo alguma forma de provisão de auditoria, tal necessidade deverá ser providenciada de alguma maneira, independente do aplicativo.

Controles de acesso aos sistemas operacionais e às aplicações, prevenindo acesso não autorizado à informação contida nos sistemas de informação, são contabilizados e registrados pela maioria dos sistemas operacionais [19]. Tanto a rotina, quanto o correspondente arquivo gerado pela rotina de contabilização, foram desenvolvidos para serem usados pelo pessoal de computação, mas representam também excelente ferramenta para a auditoria de sistemas [22].

Todas transações de negócios e os dados derivados dessas transações também deverão ser contabilizados e registrados [20]. A partir da década de 80, e devido ao barateamento das plataformas de *hardware/software* para executar SGBD (sistema gerenciador de banco de dados) relacional, este SGBD passou a dominar o mercado e converteu-se em padrão internacional, sendo os sistemas de informação atualmente quase que exclusivamente desenvolvidos com o uso de SGBD relacional [21].

A estrutura mais usada para registrar as modificações no banco de dados é o *log*. O *log* é uma seqüência de registros que mantém, em uma área separada e segura, um arquivo atualizado das atividades em um banco de dados. Há vários tipos de registros de *log*. Um registro de atualização de *log* descreve uma única escrita do banco de dados e contém um identificador da transação, um identificador do item de dado, o valor antigo e o valor novo [23]. A ação de término de uma transação grava os novos valores de objetos modificados no *log*, antes de efetiva-los na base de dados [24]. Há outros registros de *log* para arquivar eventos significativos durante o processamento de uma transação, como o início da transação, sua efetivação ou aborto [23].

A maioria dos sistemas gerenciadores de banco de dados mantém esse arquivo de *log* para recuperação, mas esse *log* geralmente é inadequado para os auditores investigarem quem é o responsável pelas mudanças feitas na base de dados, em termos de descrição e conteúdo, e em que data e horário as mudanças foram feitas [7].

Percebe-se que as evidências disponíveis para identificar as ações executadas nos sistemas de informação, quando não houve a implementação de provisão para auditoria futura, obtidas de registros do sistema operacional e *log* de banco de dados relacional, não são adequadas para análise dos auditores de sistemas de informação.

O objetivo desse trabalho é apresentar uma metodologia para o desenvolvimento de auditoria independente de sistemas de informação, com foco na integridade das

informações. Sua abordagem baseia-se nos controles internos do sistema operacional e do gerenciador de banco de dados, disponibilizando evidências para rastrear as atividades dos usuários de sistemas de informação sobre as informações. A complexidade de análise dos registros de controle é bastante reduzida através do processamento e inter-relacionamento dos dados e da provisão de uma interface simplificada para os auditores.

2 CONTROLE E AUDITORIA DE SISTEMAS DE INFORMAÇÃO

Enquanto há 50 anos atrás, a maioria das nossas necessidades de processamento de dados era totalmente suprida manualmente, hoje os computadores executam muito dessas necessidades nos setores público e privado de nossa economia, exigindo a manutenção da integridade dos dados processados. Muitas pessoas temem que as crescentes capacidades de processamento não são bem controladas [8].

O uso descontrolado dos computadores pode ter um amplo impacto em uma sociedade. Informação imprecisa pode causar alocação errônea de recursos dentro da economia, e fraudes podem ocorrer devido a controles de sistemas inadequados. Nesse capítulo objetiva-se justificar a necessidade do processo de auditoria em sistemas de informação, relacionar os fatores que a influenciam, apresentar uma definição para o processo de auditoria, e fazer considerações sobre o processo e seus fundamentos.

2.1 NECESSIDADE DE CONTROLE E AUDITORIA DE COMPUTADORES

Os computadores são utilizados extensivamente para processar dados e prover informação para a tomada de decisão, situação favorecida pela atual disponibilidade de microcomputadores poderosos associados a pacotes de *software*, e pela intensa competição no mercado de tecnologia de informação.

Devido à intensa participação em nos assistir no processamento de dados e na tomada de decisão, é importante que o uso de computadores seja controlado. A figura 2.1 mostra sete razões para estabelecer uma função para examinar controles sobre processamento de dados baseado em computadores.

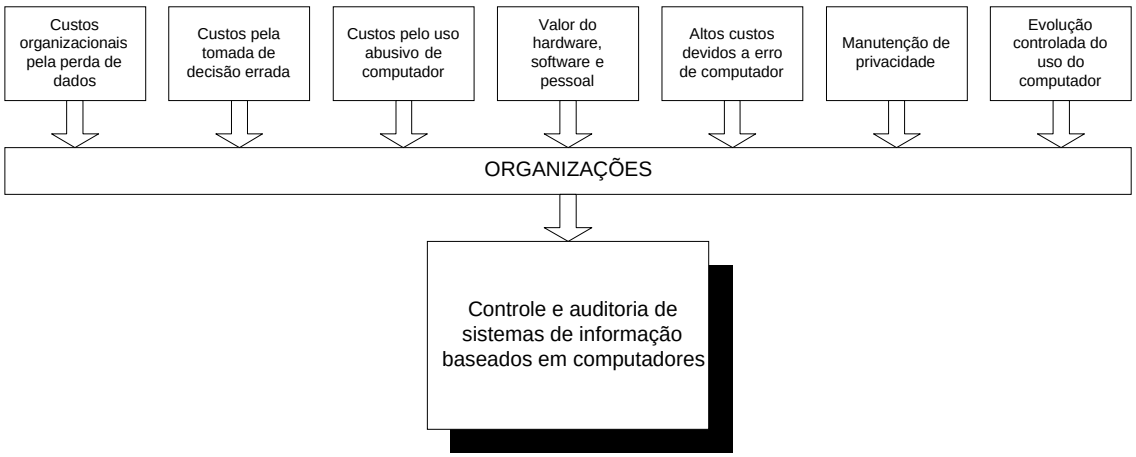


Figura 2.1: Fatores influenciando na auditoria de computadores

2.1.1 Custos Organizacionais pela Perda de Dados

Os dados compõem um recurso crítico necessário para as operações contínuas de uma organização. Eles provêm para a organização uma imagem de si mesma, seu ambiente, sua história e seu futuro. Se essa imagem é precisa, a organização aumenta suas habilidades de adaptação e sobrevivência em um ambiente de mudança. Se essa imagem é imprecisa ou perdida, a organização pode sofrer perdas substanciais.

Perdas de dados podem ocorrer quando os controles existentes sobre os computadores são relaxados. Além dos prejuízos diretos ao negócio, a imagem da empresa e a competência da administração podem ser questionadas.

2.1.2 Tomada de Decisão Incorreta

Tomar decisões de alta qualidade depende em parte da qualidade dos dados e da qualidade das regras de decisão existentes nos sistemas de informação baseados em computador.

A importância de dados precisos depende do tipo de decisão. Decisões estratégicas toleram algum erro nos dados, dada à natureza de longo prazo e a incerteza que envolve esse tipo de decisão. Decisões de controle administrativo e operacional exigem dados precisos para tomada de decisão. Esses tipos de decisão envolvem detecção, investigação e correção de processos fora do controle. Dados imprecisos podem ocasionar investigações desnecessárias e custosas, ou permitir que processos

fora do controle não sejam detectados. Além disso, dados imprecisos podem ter impacto em outras partes que tenham interesse em uma organização, tais como investidores, por exemplo.

A importância de ter regras de decisão precisas também depende do tipo de decisão sendo tomada, podendo variar de pequenas a grandes perdas monetárias, até tragédias, no caso de um sistema especialista para diagnóstico médico, por exemplo.

2.1.3 Custo do Uso Abusivo de Computador

O maior estímulo para o desenvolvimento da função de auditoria de sistemas de informação, dentro das organizações, geralmente parece estar relacionado com o uso abusivo de computador, que é caracterizado como um incidente associado com a tecnologia de computador onde a vítima sofreu ou poderia ter sofrido alguma perda, e o criminoso intencionalmente obteve ou poderia ter obtido algum ganho.

Alguns dos principais tipos de uso abusivo de computador que uma organização poderia encontrar incluem:

- *Hacking*;
- Vírus;
- Acesso físico ilegal;
- Abuso de privilégios;
- Destruição de ativos;
- Roubo de ativos;
- Modificação de ativos;
- Violação de privacidade;
- Interrupção de operações;
- Uso não autorizado de ativos;
- Danos pessoais.

O número e tipos de ameaças que levam ao uso abusivo de computador estão aumentando. Vírus mais letais e complexos continuam a aparecer. Similarmente, o rápido crescimento da Internet tem exposto as organizações, com segurança inadequada, a partes hostis externas, que antes não as afetariam. Outra informação preocupante é que cerca de 80% dos casos de uso abusivo de computador são cometidos por empregados internos, mas somente 20% das organizações fazem revisões de segurança de empregados potenciais.

2.1.4 Valor de *Hardware*, *Software* e Pessoal de Informática

Em adição aos dados, *hardware*, *software* e pessoal de informática são recursos organizacionais críticos. Algumas organizações têm investimentos milionários em *hardware*, e mesmo devidamente assegurados, uma perda intencional ou não, pode causar uma interrupção considerável. *Software* também constitui um investimento considerável que, se corrompido ou destruído, pode impedir que se continuem as operações se não puder ser recuperado prontamente. O roubo de *software* pode permitir que informações confidenciais possam ser reveladas a concorrentes. Pessoal qualificado é sempre um recurso precioso, particularmente à luz da escassez de alguns países.

2.1.5 Alto Custo de Erro de Computador

Computadores atualmente executam muitas funções críticas dentro de nossa sociedade. Conseqüentemente, o custo de erro de computador em termos de perdas de vidas, privação de liberdade, ou danos ao meio ambiente pode ser alto. Crescentemente parece que as organizações serão responsabilizadas pelos danos que ocorram como um resultado de erros de projeto, implementação, ou operação de seus sistemas de computadores.

2.1.6 Manutenção de Privacidade

Muitos dados são coletados atualmente sobre cada indivíduo: impostos, crédito, saúde, escolaridade, emprego, residência, etc. A capacidade de processamento dos computadores, particularmente sua alta produtividade, integração e volume de consultas, causa nas pessoas e organizações a dúvida se a privacidade individual não

ruiu a níveis inaceitáveis. Ativistas de direito civil sustentam preocupações substanciais sobre o uso de sistemas de computadores para propósitos de identificação de perfis pessoais.

Muitas nações consideram a privacidade um direito humano, e consideram responsabilidade das pessoas preocupadas com o processamento de dados por computador garantir que dados pessoais não sejam disponibilizados facilmente, e que sejam somente utilizados para os devidos fins.

2.1.7 Evolução Controlada do Uso de Computador

De tempos em tempos surgem discussões sobre o uso da tecnologia de computador na sociedade. Muitos cientistas argumentam que o uso de computadores para suportar o comando de armas nucleares e sistemas de controle não pode ser garantido, devido à complexidade dos sistemas de computador, e as conseqüências podem ser catastróficas.

Outra preocupação comum das pessoas é sobre os efeitos que o uso de computadores terá sobre sua vida profissional. A tecnologia de computador causará desemprego ou embrutecerá o serviço? Que efeitos terá sobre o bem estar físico e mental de seus usuários?

Pode-se argumentar que a tecnologia é neutra — não é boa, nem má. O uso da tecnologia pode, entretanto, produzir grandes problemas sociais.

2.2 DEFINIÇÃO DE AUDITORIA DE SISTEMAS DE INFORMAÇÃO

Auditoria de sistemas de informação é o processo de coleta e avaliação de evidências para determinar se um sistema computadorizado protege ativos, mantém a integridade dos dados, permite que as metas organizacionais sejam efetivamente atingidas, e utiliza os recursos eficientemente. Dessa forma suporta objetivos tradicionais de auditoria: objetivos certificadores (auditoria externa) com foco na proteção de ativos e integridade de dados, e objetivos administrativos (auditoria interna) com foco na efetividade e eficiência.

Entende-se auditoria de sistemas de informação como sendo uma força que permite às organizações melhor atingir quatro objetivos, conforme a figura 2.2:

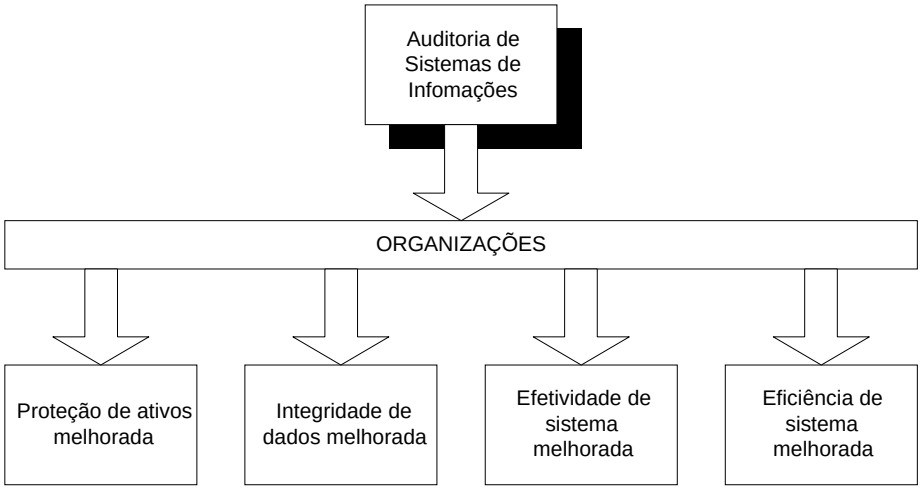


Figura 2.2: Objetivos da auditoria de sistemas de informação

2.2.1 Objetivos de Proteção de Ativos

Os ativos de um sistema de informação de uma organização incluem *hardware*, *software*, equipamentos, pessoas (conhecimento), arquivos de dados, documentação de sistemas e suprimentos. Como todos ativos, devem ser protegidos por um sistema de controle interno. *Hardware* pode ser danificado intencionalmente. *Software* proprietário ou conteúdo de arquivos de dados podem ser roubados ou destruídos. Suprimentos podem ser utilizados para propósitos não autorizados. Esses ativos estão geralmente concentrados em pequeno número de locais, tornando a proteção de ativos um objetivo especialmente importante de ser atingido em muitas organizações.

2.2.2 Objetivos de Integridade de Dados

Integridade de dados é um conceito fundamental na auditoria de sistemas de informação. É um estado implicando que os dados tenham certos atributos: completeza, validade, pureza, e veracidade. Se a integridade dos dados não é mantida, uma organização não tem mais uma representação verdadeira de si mesma ou dos eventos, e pode perder vantagem competitiva. Apesar de tudo, a manutenção da integridade de

dados pode ser atingida somente a um custo. Os benefícios obtidos deveriam exceder os custos dos procedimentos de controle necessários.

Três fatores principais afetam o valor de um item de dado para uma organização e assim a importância de manter a integridade daquele item de dado:

1. O valor do conteúdo informativo do item de dado para os tomadores de decisão individuais;
2. A extensão de compartilhamento do item de dado entre tomadores de decisão;
3. O valor do item de dado para os concorrentes.

2.2.3 Objetivos de Efetividade de Sistema

Um sistema de informação efetivo efetua seus objetivos. Para avaliar se um sistema reporta informação de uma forma que facilite a tomada de decisão por seus usuários, os auditores devem saber as características dos usuários e do ambiente de tomada de decisão.

Auditoria de efetividade geralmente ocorre após um sistema ter executado por algum tempo. A administração requer uma pós-auditoria para determinar se o sistema está atingindo os objetivos estabelecidos.

Se um sistema é complexo e custoso para implementar, a administração poderia querer que os auditores executassem uma avaliação independente para determinar se o projeto atenderá provavelmente as necessidades dos usuários.

2.2.4 Objetivos de Eficiência de Sistema

Um sistema de informação eficiente utiliza o mínimo de recursos para atingir seus objetivos exigidos. Geralmente não há uma noção nítida disso e não se pode considerar um sistema isolado dos outros.

A eficiência torna-se especialmente importante quando um computador não tem mais excesso de capacidade. O desempenho de sistemas de aplicativos individuais degrada, e usuários podem tornar-se crescentemente frustrados. A administração deve

então decidir se a eficiência pode ser melhorada ou se recursos extras devem ser comprados.

2.3 EFEITOS DOS COMPUTADORES NOS CONTROLES INTERNOS

As metas de proteção de ativos, integridade de dados, efetividade de sistema, e eficiência de sistema podem ser atingidas somente se a administração de uma organização ativar um sistema de controle interno. Em sistemas de computador os componentes dos sistemas de controle interno tradicionais foram adotados e adaptados para ajustarem-se ao ambiente computacional (figura 2.3).

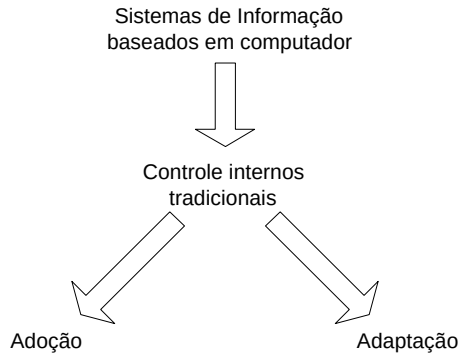


Figura 2.3: Efeitos dos sistemas de informação nos controles internos tradicionais

2.3.1 Separação de Obrigações

Diferente de um sistema manual, onde pessoas separadas são responsáveis por iniciar transações, registrar transações e manter custódia de ativos, em um sistema de computador todas essas funções, consideradas incompatíveis, podem estar juntas em um único programa, e poderia ser ineficiente separa-las em programas diferentes. A separação de obrigações deve existir de uma forma diferente. Quando tiver sido determinado que o programa executa corretamente, a capacidade de executar o programa em modo de produção e a capacidade de modificar o programa devem ser separadas.

2.3.2 Delegação de Responsabilidade e Autoridade

Num sistema de computador, delegar autoridade e responsabilidade em uma maneira não ambígua pode ser difícil, porque alguns recursos são compartilhados entre vários usuários, e não é fácil identificar quem cometeu uma violação e quem é responsável por identificar e resolver o problema.

Linhas de autoridade e responsabilidade perdem a clareza devido ao rápido crescimento das ferramentas de usuário final. Mais usuários estão desenvolvendo, modificando, operando e mantendo seus próprios sistemas de aplicações, em vez de ter esse trabalho executado por profissionais de sistemas de informação. Embora isso seja um benefício substancial para a organização, infelizmente aumenta em muito os problemas de exercer controle geral sobre o uso de computador.

2.3.3 Pessoal Competente e Confiável

Poder substancial é geralmente investido em pessoas responsáveis pelos sistemas de informação baseados em computador desenvolvidos, implementados, operados e mantidos dentro das organizações. Esse poder geralmente supera o poder investido no pessoal responsável pelos sistemas manuais.

Garantir que uma organização tenha pessoal de sistemas de informação competente e confiável é uma tarefa difícil. Em muitos países e por muitos anos, tem havido deficiência em prover pessoal competente e bem treinado. Isso tem levado as organizações a comprometer sua seleção da equipe. Pior ainda, tem sido difícil para as organizações avaliar a competência e a integridade de suas equipes de sistemas de informação, devido à troca constante de profissionais. A rápida evolução da tecnologia também inibe a capacidade dos administradores de avaliar a experiência de seus empregados. Alguns profissionais também parecem não apresentar um senso de ética bem desenvolvido ou se divertem subvertendo controles.

2.3.4 Sistema de Autorizações

A administração emite dois tipos de autorização para execução de transações. Primeiro, autorizações gerais que estabelecem políticas para a organização seguir.

Segundo, autorizações específicas aplicadas a transações individuais. Geralmente isso está embutido dentro de um programa de computador. Ao avaliar a adequação dos procedimentos de autorização, os auditores devem examinar não somente o trabalho dos empregados, mas também a veracidade do processamento do programa.

Também é mais difícil avaliar se a autorização designada a um indivíduo é consistente com o desejo do administrador. Por exemplo, pode ser difícil determinar exatamente que dados o usuário pode ter acesso, quando ele é guarnecido de uma linguagem de recuperação de dados genérica, tendo a possibilidade de violar a privacidade de certas informações.

2.3.5 Documentos e Registros Adequados

Devido à característica *on-line* dos sistemas de informação atuais, documentos podem não ser exigidos para suportar o início, execução e registro de algumas transações. Similarmente, algumas transações podem ser ativadas automaticamente por um sistema de computador. Assim, nenhuma trilha visível de auditoria ou gerenciamento estaria disponível para rastrear a transação.

Isso não é problema para os auditores, desde que o sistema de informação tenha sido projetado para manter um registro de todos eventos e que o registro possa ser facilmente recuperado. Em sistemas de computador bem projetados, trilhas de auditoria são geralmente mais abrangentes do que aquelas mantidas em sistemas manuais. Infelizmente, nem todos os sistemas de computador são bem projetados. Quando essa situação é somada a uma reduzida habilidade de separar funções incompatíveis, sérios problemas podem surgir.

2.3.6 Controle Físico sobre Ativos e Registros

Controle físico sobre acesso a ativos e registros é crítico tanto em sistemas manuais, quanto em sistemas de computador. Entretanto, sistemas de computador diferem no fato de concentrar os ativos e registros dos sistemas de informação da organização. Isso elimina a necessidade de deslocamento físico para ter acesso aos ativos e registros, e aumenta as perdas que podem surgir do uso abusivo do computador

ou desastres, podendo cessar as operações da organização, se esta não estiver adequadamente preparada.

2.3.7 Supervisão Administrativa Adequada

Em sistemas manuais a supervisão administrativa das atividades dos empregados é relativamente simples, porque gerentes e empregados estão geralmente na mesma localização física. Em sistemas de computador, entretanto, as facilidades de comunicação de dados permitem a realização de tarefas remotamente, e controles devem ser construídos nos sistemas de computador para compensar os controles geralmente exercidos pela observação e investigação.

Como muitas atividades são realizadas eletronicamente em sistemas de computador, os administradores devem periodicamente ter acesso às trilhas de auditoria das atividades do empregado e examina-las por ações não autorizadas. Assim como nos controles, a efetividade da observação e investigação diminui.

2.3.8 Verificação Independente de Desempenho

Em sistemas manuais, verificações independentes são realizadas porque os empregados costumam esquecer procedimentos, cometer erros genuínos, tornar-se descuidados, ou falham intencionalmente em seguir procedimentos prescritos. Verificações por pessoas independentes ajudam a detectar quaisquer erros ou irregularidades. Se o código do programa em um sistema de computador está autorizado, preciso, e completo, o sistema seguirá sempre os procedimentos designados na ausência de algum outro tipo de falha, como falha de *hardware* ou *software*. Assim a verificação independente de desempenho de programas tem geralmente pouco valor. A ênfase do controle desvia para garantir a veracidade do código do programa. Os auditores devem avaliar os controles estabelecidos para o desenvolvimento, modificação, operação, e manutenção do programa.

2.3.9 Comparação das Informações de Controle Registradas com os Ativos

Os dados e os ativos que os dados tem em vista representar, deveriam ser periodicamente comparados para determinar se os dados estão incompletos ou

imprecisos, ou se faltas ou excessos de ativos tem ocorrido. Em sistemas de computador, algum *software* é utilizado para preparar os dados para essa verificação. Se modificações não autorizadas ocorrerem no programa ou arquivos de dados que o programa utiliza, uma irregularidade poderia não ser descoberta. Novamente controles internos devem ser implementados para garantir a veracidade do código do programa, porque a separação de obrigações não mais se aplica aos dados sendo preparados com propósito de comparação.

2.4 EFEITOS DOS COMPUTADORES NA AUDITORIA

Quando os sistemas de computador surgiram, muitos auditores estavam preocupados que a natureza fundamental da auditoria poderia ter que mudar para lidar com a nova tecnologia. Agora está claro que não é esse o caso. Os auditores devem ainda prover uma avaliação competente e independente indicando se um conjunto de atividades econômicas tem sido registrado e informado de acordo com padrões e critérios estabelecidos. Todavia, sistemas de computador têm afetado como os auditores realizam suas duas funções básicas: coleta e avaliação de evidências.

2.4.1 Mudanças na Coleta de Evidências

Coletar evidências sobre a segurança em um sistema de computador é geralmente mais complexo do que em um sistema manual, devido à diversidade e complexidade da tecnologia de controle interno. Os auditores devem entender esses controles, de forma a ter capacidade de coletar evidências competentemente.

Hardware e *software* evoluem rapidamente, o que torna o entendimento da tecnologia de controle difícil, ocorrendo intervalos entre o surgimento e entendimento da tecnologia. Também se torna mais difícil coletar evidências sobre a segurança dos controles. Em alguns casos não é possível ao auditor coletar evidência de forma manual. Assim eles mesmos necessitam de sistemas de computador para coletar a evidência necessária. Novas ferramentas de auditoria podem ser requeridas devido à evolução da tecnologia, e infelizmente também aqui ocorre um intervalo para seu desenvolvimento. Nesse ínterim, os auditores são geralmente forçados a comprometer de alguma forma quando realizando a função de coleta de evidência.

2.4.2 Mudanças na Avaliação de Evidência

Devido à crescente complexidade dos sistemas de computador e da tecnologia de controle interno, também está mais difícil avaliar as conseqüências das forças e fraquezas dos controles para a segurança geral dos sistemas. Primeiro, os auditores devem entender quando um controle está agindo de forma segura ou não. Depois, eles devem ser capazes de rastrear as conseqüências da força ou fraqueza do controle através do sistema. Em um ambiente compartilhado isso é particularmente difícil. Uma única transação poderia modificar múltiplos itens de dados que são utilizados por diversos usuários fisicamente dispersos.

Erros em sistemas de computadores tendem a ser deterministas: um programa errado sempre executará incorretamente. Além disso, erros são gerados em grande velocidade, e o custo para corrigir e executar novamente os programas pode ser alto, envolvendo correções de projeto e programação. Assim, controles internos que garantam que sistemas de computador de alta qualidade sejam projetados, implementados, operados, e mantidos são críticos.

O ônus sobre os auditores é garantir que esses controles são suficientes para manter a proteção dos ativos, integridade dos dados, efetividade dos sistemas, e eficiência dos sistemas, e que eles estejam disponíveis e operando de forma segura.

2.5 FUNDAMENTOS DA AUDITORIA DE SISTEMAS DE INFORMAÇÃO

Auditoria de sistemas de informação não é apenas uma extensão da auditoria tradicional. O reconhecimento da necessidade de uma função de auditoria de sistemas de informação veio de duas direções. Primeiro, os auditores perceberam que os computadores afetaram sua habilidade de executar a função de validação. Segundo, tanto administradores corporativos, quanto de sistemas de informação, reconheceram que os computadores eram recursos valiosos que necessitavam controle como qualquer outro recurso chave dentro da organização.

A disciplina de auditoria de sistemas de informações foi modelada pelo conhecimento obtido de outras quatro disciplinas (figura 2.4): auditoria tradicional,

gerenciamento de sistemas de informação, ciência comportamental, e ciência da computação.

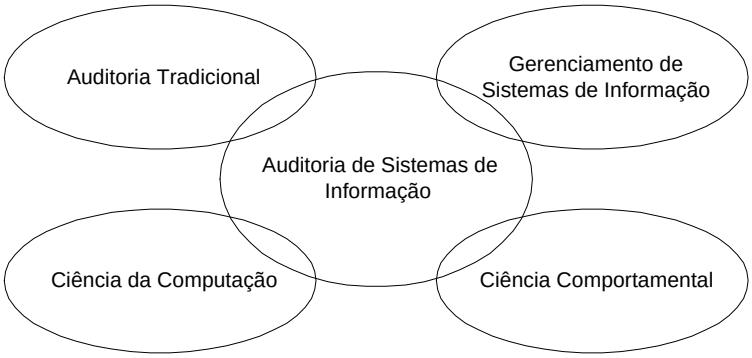


Figura 2.4: Auditoria de sistemas de informação como interseção de outras disciplinas

2.5.1 Auditoria Tradicional

A auditoria tradicional trouxe para a auditoria de sistemas de informação uma riqueza de conhecimentos e experiências com técnicas de controle interno, o que teve um impacto no projeto tanto dos componentes manuais, quanto de máquina de um sistema de informação.

Similarmente, conceitos de auditoria tradicionais, como totais de controle, são também relevantes na atualização e manutenção de arquivos por programas de computador, os quais devem garantir que todos os dados da transação são processados e processados corretamente.

A longa evolução e experiência abrangente da auditoria tradicional destacam a importância crítica de ter objetivo, evidência verificável e uma avaliação independente dos sistemas de informação.

Talvez mais importante, a auditoria tradicional trouxe para a auditoria de sistemas de informação uma filosofia de controle, a qual envolve examinar os sistemas de informação com uma mente crítica, sempre com uma visão questionando a capacidade de um sistema de informação de proteger ativos, manter a integridade dos dados, e atingir seus objetivos efetiva e eficientemente.

2.5.2 Gerenciamento de Sistemas de Informação

O início da história dos sistemas de informação baseados em computador mostra alguns desastres espetaculares. Ocorreram custos excessivos massivos, e muitos sistemas falharam em atingir seus objetivos estabelecidos. Como resultado, por muitos anos os pesquisadores preocuparam-se em identificar maneiras melhores de gerenciar o desenvolvimento e implementação de sistemas de informação.

Alguns avanços importantes foram feitos. Técnicas de gerenciamento de projeto foram trazidas para a área de sistemas de informação com sucesso considerável. Documentação, padrões, orçamentos, e investigação de variação são agora enfatizados. Maneiras melhores de desenvolver e implementar sistemas foram desenvolvidas, como análise, projeto e programação orientados a objetos permitiram aos programadores desenvolver *software* mais rápido, com menos erros e característica de manutenção mais fácil. Esses avanços afetam a auditoria de sistemas de informação porque eles, no final das contas, afetam os objetivos de proteção de ativos, integridade de dados, efetividade e eficiência de sistemas.

2.5.3 Ciência Comportamental

Sistemas de computador falham algumas vezes porque seus projetistas não avaliam as difíceis questões humanas que estão geralmente associadas com o desenvolvimento e a implementação de um sistema, tais como resistência, sabotagem ou evasão dos controles. Similarmente, projetistas e usuários poderiam ter dificuldades de comunicação entre si devido às diferentes concepções de um propósito inerente em um domínio de aplicação.

Auditores devem entender as condições que podem levar a problemas comportamentais, e conseqüentemente a possíveis falhas de sistema. Cientistas comportamentais, especialmente teóricos organizacionais, contribuíram muito para o entendimento dos problemas pessoais que podem surgir dentro das organizações, e essas descobertas têm sido aplicadas no desenvolvimento e implementação de sistemas de informação.

2.5.4 Ciência da computação

Cientistas da computação também têm se preocupado em como os objetivos de proteção de ativos, integridade de dados, efetividade e eficiência de sistemas poderiam ser mais bem atingidos. Eles têm conduzido pesquisas em como provar a exatidão do *software* formalmente, construir sistemas de computação tolerantes à falha, projetar sistemas operacionais seguros, e como transmitir dados seguramente através de um meio de comunicação.

Os avanços da ciência da computação proveram tanto benefícios, quanto problemas para o trabalho dos auditores. De um lado, eles agora podem estar menos preocupados com a segurança de certos componentes em um sistema de computador. De outro lado, se essas inovações forem usadas impropriamente, eles podem ter dificuldades em detectar o abuso.

2.6 RESUMO

Nesse capítulo foi apresentada justificativa para a necessidade de auditoria de sistemas de informação, sendo o uso abusivo do computador o maior estímulo para sua implementação. Foi definida como um processo de coleta e avaliação de evidências para determinar se um sistema computadorizado protege ativos, mantém a integridade dos dados, permite que as metas organizacionais sejam efetivamente atingidas, e utiliza os recursos eficientemente. Para isso os componentes dos sistemas de controle interno tradicionais foram adotados e adaptados para se ajustarem ao ambiente computacional. Percebeu-se que os computadores eram recursos valiosos que necessitavam controle como qualquer outro recurso chave dentro da organização e que afetaram a habilidade dos auditores de executar a função de validação. A auditoria de sistemas de informação foi modelada a partir da interseção de disciplinas tais como a auditoria tradicional, gerenciamento de sistemas de informação, ciência comportamental, e ciência da computação. Essa base servirá de fundamento para o próximo capítulo, o qual apresentará algumas abordagens para a implementação de um processo de auditoria em sistemas de informação.

3 CONDUÇÃO DE UMA AUDITORIA DE SISTEMAS DE INFORMAÇÃO

É uma grande responsabilidade realizar a auditoria de sistemas de informações de uma organização com centenas de programadores e analistas, muitos computadores e milhares de arquivos de dados. Obviamente nem todas organizações são desse tamanho. Entretanto, excetuando-se as pequenas organizações, não se pode fazer uma verificação detalhada de todos os dados processados pelos sistemas de informação. Em vez disso, deve-se contar com uma amostra dos dados para determinar se os objetivos da auditoria de sistemas de informação estão sendo atingidos.

Para realizar a auditoria de sistemas de informação de forma a garantir segurança razoável de que uma organização protege seus ativos de processamento de dados, mantém a integridade dos dados, e obtém efetividade e eficiência dos sistemas, serão apresentadas visões gerais de algumas abordagens que podem ser usadas com esse fim.

A natureza dos controles será examinada e serão apresentadas algumas técnicas para simplificar e prover ordem à complexidade de fazer julgamentos de avaliação de sistemas de informação baseados em computador, e serão considerados os passos básicos a serem seguidos na condução de uma auditoria de sistemas de informação.

3.1 NATUREZA DOS CONTROLES

Os auditores de sistemas de informações ultimamente estão preocupados com a avaliação da confiança, ou efetividade operacional, dos controles. É importante entender o que se quer dizer por controle: Um controle é um sistema que previne, detecta, ou corrige eventos ilegais.

Três aspectos chave apresentam-se nessa definição. Primeiro, um controle é um sistema, ou seja, compreende um conjunto de componentes inter-relacionados que colaboram para atingir algum propósito geral. Segundo, os controles focam eventos ilegais. Um evento ilegal pode surgir se uma entrada não autorizada, imprecisa, incompleta, redundante, ineficaz ou ineficiente adentra o sistema, ou se o sistema transforma a entrada de uma maneira não autorizada, imprecisa, incompleta,

redundante, ineficaz ou ineficiente através de código errôneo. Terceiro, os controles são usados para prevenir, detectar, ou corrigir eventos ilegais.

O propósito geral dos controles é reduzir perdas esperadas de eventos ilegais que possam ocorrer em um sistema. Isso se faz de duas maneiras. Primeira, controles preventivos reduzem a probabilidade de eventos ilegais ocorrerem em primeiro lugar. Segunda, controles de detecção e correção reduzem a quantidade de perdas que surgirão se um evento ilegal ocorrer.

A tarefa dos auditores é determinar se os controles estão no lugar e funcionando para prevenir que eventos ilegais possam ocorrer no sistema. Os auditores devem estar preocupados para ver se ao menos um controle existe para cobrir cada evento ilegal que possa ocorrer. Usualmente, alguns eventos ilegais em um sistema não serão cobertos porque um controle a um custo efetivo não pode ser encontrado. Mesmo que um evento ilegal esteja coberto por um controle, deve-se avaliar se o controle está operando efetivamente.

3.2 ADMINISTRAÇÃO DA COMPLEXIDADE

Conduzir uma auditoria de sistemas de informação é um exercício de tratar com a complexidade, dada a abundância de sistemas. Por ser a complexidade a causa principal de problemas encarada por muitos profissionais, foram desenvolvidas algumas orientações para reduzi-la, sendo as duas principais:

1. Fatorar o sistema para ser avaliado em subsistemas;
2. Determinar a confiança de cada subsistema e as implicações do nível de segurança de cada subsistema para o nível geral de confiança no sistema.

3.2.1 Fatoração em Subsistemas

O primeiro passo para entender um sistema complexo é parti-lo em subsistemas. Um subsistema é um componente de um sistema que desempenha uma função específica necessária para que o sistema atinja seus objetivos fundamentais. A fatoração é um processo repetitivo que termina quando se sente que o sistema foi partido em partes pequenas o suficiente para serem entendidas e avaliadas. O sistema a ser avaliado

pode então ser descrito como uma estrutura de níveis de subsistemas, cada um desempenhando uma função necessária por algum subsistema de nível superior (figura 3.1)

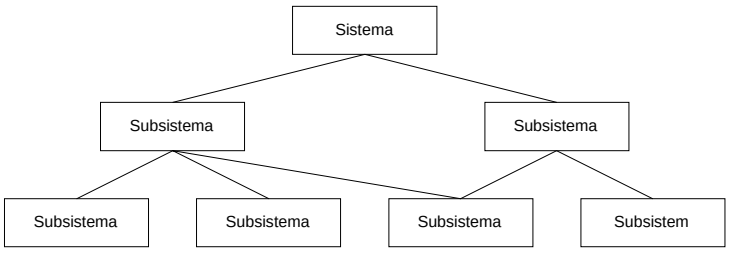


Figura 3.1: Estrutura de níveis de sistema e subsistemas

Uma forma sugerida para o processo de fatoração é considerar a função que ele desempenha: subsistema de entrada de pedidos, contas a receber, etc. Além da função, a teoria de sistemas indica duas outras orientações que deveriam fundamentar a forma de delinear subsistemas. Primeira, cada subsistema deveria ser relativamente independente de outros subsistemas, ou seja, fracamente acoplado a outros subsistemas. Dessa forma cada subsistema pode ser avaliado isoladamente, podendo-se desconsiderar os efeitos das forças e fraquezas dos controles em outros sistemas. Segunda, cada subsistema deveria ser internamente coeso. Todas as atividades desempenhadas deveriam ser direcionadas para executar uma única função, o que facilita o entendimento e avaliação das atividades cumpridas pelo subsistema.

Deve-se, portanto, avaliar a extensão do acoplamento e coesão no subsistema escolhido. Se ele não for fracamente acoplado e internamente coeso, deve-se realizar uma fatoração diferente. Se nenhuma fatoração parecer capaz de delinear subsistemas que possuam essas características, será difícil avaliar a confiança do sistema, porque suas atividades serão muito obscuras.

Conceitualmente, um sistema pode ser fatorado de diversas maneiras diferentes. Duas maneiras são consideradas especialmente úteis na condução de auditorias de sistemas de informações. A primeira é de acordo com as funções administrativas que devem ser desempenhadas para garantir que o desenvolvimento, implementação, operação, e manutenção de sistemas de informação prossigam em uma maneira planejada e controlada. Sistemas administrativos funcionam para prover uma

infraestrutura estável na qual sistemas de informação podem ser construídos, operados, e mantidos no cotidiano. Vários tipos de sistemas administrativos foram identificados que correspondem à organização hierárquica e algumas das principais tarefas desempenhadas pela função dos sistemas de informação:

- Administração superior: Deve garantir que a função de sistemas de informação seja bem administrada. É responsável primeiramente pelas decisões políticas de longo prazo de como os sistemas de informação serão usados na organização;
- Administração de sistemas de informação: Tem a responsabilidade geral pelo planejamento e controle de todas atividades de sistemas de informação. Também aconselha a administração superior em relação à tomada de decisão política de longo prazo e traduz as políticas de longo prazo em metas e objetivos de curto prazo;
- Administração de desenvolvimento de sistemas: É responsável pelo projeto, implementação, e manutenção dos sistemas de aplicação;
- Administração de programação: É responsável pela programação de novos sistemas, manutenção de sistemas antigos, e prover *software* para suporte de sistemas em geral;
- Administração de dados: É responsável por estabelecer questões de planejamento e controle com relação ao uso dos dados da organização;
- Administração de garantia de qualidade: É responsável por assegurar o desenvolvimento, implementação, operação, e manutenção de sistemas de informação, conforme padrões de qualidade estabelecidos;
- Administração de segurança: É responsável por controles de validação e segurança física sobre a função de sistemas de informação;
- Administração de operação: É responsável pelo planejamento e controle das operações diárias dos sistemas de informação.

A segunda fatoração é de acordo com as funções da aplicação que precisam ser assumidas para executar processamento de informação confiável. Corresponde à

abordagem cíclica usada tradicionalmente para conduzir uma auditoria. Os sistemas de informação suportando uma organização são agrupados primeiramente em ciclos, que correspondem às áreas de negócio das empresas. Cada ciclo é fatorado em um ou mais sistemas de aplicação. Esses, por sua vez, são fatorados em subsistemas. O conjunto de subsistemas de aplicação inclui o seguinte:

- Fronteira: Compreende os componentes que estabelecem a interface entre o usuário e o sistema;
- Entrada: Compreende os componentes que capturam, preparam, e entram comandos e dados no sistema;
- Comunicações: Compreende os componentes que transmitem dados entre subsistemas e sistemas.
- Processamento: Compreende os componentes que executam a tomada de decisões, cálculos, classificação, ordenação, e sumarização dos dados no sistema;
- Base de dados: Compreende os componentes que definem, adicionam, acessam, modificam, e excluem dados no sistema;
- Saída: Compreende os componentes que buscam e apresentam os dados aos usuários do sistema.

Nenhuma dessas abordagens é definitiva, mas fundamentam o que se defende em muitos corpos profissionais de auditores. Elas permitem reduzir a complexidade a um ponto em que se pode entender e avaliar a natureza e a confiança dos subsistemas.

3.3 RESUMO

Nesse capítulo foi examinada a natureza dos controles, definidos como um sistema que previne, detecta, ou corrige eventos ilegais. Foi também apresentada a necessidade de fatoração dos sistemas em subsistemas, de forma a reduzir a complexidade da tarefa de auditoria de sistemas de informação. Além disso, salientou-se a importância da independência dos subsistemas entre si, e a necessidade dos subsistemas serem internamente coesos, devendo ser todas as atividades desempenhadas

direcionadas para uma única função, o que facilita o entendimento e avaliação das atividades cumpridas pelo subsistema. Foram ainda discutidas duas abordagens de fatoração em subsistemas consideradas especialmente úteis: de acordo com as funções administrativas e de acordo com as funções de aplicação. Essa visão da estratégia de implementação de controles é muito importante para fundamentar a abordagem metodológica que será proposta nesse trabalho, a qual explorará unicamente os registros desses controles para gerar os resultados prometidos.

4 ABORDAGENS DE AUDITORIA DE SISTEMAS DE INFORMAÇÃO

Sistemas de auditoria registram evidências das modificações realizadas sobre as informações e ajudam a administração no controle de perdas financeiras ou vulnerabilidades legais.

A integridade das informações é uma preocupação vital para a maioria dos negócios. Dentro de uma escala plausível, a precisão ou validade do valor de um dado específico não é óbvia. Imprecisões podem interagir e multiplicar-se, resultando em erros maiores. Em um sistema financeiro, por exemplo, dados imprecisos podem resultar em perdas significativas. Registrar a correção e evolução de valores é a responsabilidade de um sistema de auditoria [7].

Quando sistemas de auditoria estão disponíveis, discrepâncias podem ser corrigidas, e padrões de discrepâncias podem ser identificados. É uma necessidade quando se deseja rastrear que ações levaram as informações ao estado suspeito.

Existem diversas abordagens utilizadas para a construção de uma auditoria de sistemas de informação. Especificamente será discutida, nesse momento, a auditoria focada na integridade da informação, a qual proporciona rastreabilidade das atividades dos usuários de sistemas de informação sobre as informações. Serão apresentadas, a seguir, algumas abordagens tradicionais.

4.1 AUDITORIA EMBUTIDA NO APLICATIVO

Talvez a abordagem mais comum, onde os dados de auditoria são gerados a partir do próprio aplicativo que executa as modificações na base de dados, ao aplicar regras de negócio sobre a base de dados de sistemas integrados de informação. A qualidade dos registros e o nível de detalhamento são dependentes da qualidade do projeto do sistema. Como a maioria dos sistemas é comprada com o código fonte fechado, não se pode ter certeza sobre isso. É, sobretudo, uma questão de confiança.

Muitas *software houses* limitam seus sistemas ao registro de alguns eventos mais sensíveis, representativos das regras de negócio consideradas mais críticas, em termos de abuso, e omitem um detalhamento mais amplo. Isso se deve ao enorme esforço de

desenvolvimento para implementar auditoria sobre um sistema composto de milhares de programas e milhares de estruturas de dados na base de dados. Elas esperam que seja gerada uma necessidade por parte dos clientes para então adicionar essa mudança ao sistema.

Sistemas com projetos mais apropriados para a necessidade de auditoria baseiam-se em um mecanismo que restringe todas as modificações a uma ferramenta de atualização. Assim todos os programas que operam modificações na base de dados são obrigados a referenciar essa ferramenta. Essa centralização permite que se possa manter registros de auditoria de tudo que acontece com as informações.

Para uma auditoria ser executada com o maior grau de efetividade, é pré-requisito que o auditor seja envolvido no estágio de projeto da aplicação, e não ser chamado somente depois que o sistema tenha sido instalado e implementado. Freqüentemente será identificado que os controles adequados não foram incluídos no sistema, e será oneroso ou virtualmente impossível integrá-los. Padrões de auditoria deveriam ser aplicados durante o desenvolvimento do sistema, antes que qualquer implementação tenha iniciado. Os auditores deveriam ter autoridade direta em testes de aceitação para garantir que os padrões de auditoria estão sendo aplicados [12]. Os auditores que participam no desenvolvimento de novos sistemas empregam técnicas de auditoria preventiva e são responsáveis por garantir que os atributos de auditabilidade/controle sejam construídos nos sistemas [25].

São relacionadas abaixo características particulares dessa abordagem de sistema de auditoria:

- O projeto do sistema de informação se torna mais complexo, pelo fato de incorporar todo o processo de geração de evidências em seu código fonte. Existe a necessidade de envolver pelo menos um auditor que oriente sobre os padrões de auditoria que devem ser aplicados, e realizar os testes do aplicativo, de forma a garantir a efetividade da geração da base de auditoria;
- O desenvolvimento do sistema de informação se torna mais oneroso, pois o código fonte dos aplicativos deverá incluir a programação necessária para implementar a auditoria definida durante o estágio de projeto;

- A manutenção e evolução do sistema também se tornarão mais onerosas, pois qualquer mudança na estrutura de objetos que compõem a base de dados do sistema de informação ou mudanças nas regras de negócio no modelo conceitual do sistema de informação, implicarão em mudanças na auditoria implementada no sistema. Isso representa uma interdependência entre o aplicativo e o suporte de auditoria, que pelo fato de não ser ativa, implica em controles manuais para garantir o sincronismo entre ação e registro da ação, e representa algum risco;
- Pelo fato da implementação da auditoria estar firmemente coesa às regras de negócio do aplicativo, os registros de auditoria serão representativos dessas regras, facilitando o entendimento pelos auditores, reduzindo a complexidade e o esforço para auditoria;
- A geração dos registros de auditoria ocorrerá no mesmo instante em que o aplicativo estiver aplicando a regra de negócio sobre a base de dados, permitindo uma auditoria em tempo real e a conseqüentemente a implantação de alarmes para determinados eventos;
- A geração de registros de auditoria será somente aplicada sobre as estruturas cobertas pela auditoria, durante o projeto de desenvolvimento. Se o projeto não contemplar uma abrangência ampla o suficiente para registras os fatos sobre sua base de dados completa, parte da atividade que aconteceu sobre ela ficará nebulosa. Também nada poderá ser comprovado sobre o que ocorrer sobre outras estruturas que não fazem parte desse sistema especificamente. Existem situações de integrações entre sistemas ou troca de informações entre empresas, onde a informação é exportada/importada através de programas que não fazem parte do sistema original, e que objetivam eliminar a reentrada de informações de forma manual. Esse tipo de atividade sobre a base de dados acaba não sendo registrado nos arquivos de auditoria, e isso dificulta e/ou confunde a auditoria;
- O fato da geração dos registros de auditoria estar vinculada à aplicação da regra de negócio pelo aplicativo, e o armazenamento se efetuar na mesma base de dados, sob o mesmo gerenciamento de banco de dados, o

sincronismo entre os dados do sistema de informação e os dados do sistema de auditoria fica garantido pelo ambiente transacional do banco de dados. Dessa forma, qualquer necessidade de recuperação do banco de dados trará não somente os dados do sistema de informação a um estado consistente, como também os dados do sistema de auditoria, não havendo necessidade de algum controle para garantir esse sincronismo. Porém não haverá registro de que esse fato ocorreu nos registros de auditoria;

- A implementação dessa abordagem representa maiores custos em armazenamento, já que se costuma armazenar os registros de auditoria na mesma base de dados do aplicativo. Sendo maior o custo de armazenamento, implica também em maior custo de processamento, pois o gerenciador de banco de dados também será responsável pelo gerenciamento dessa massa de dados adicional. Para o administrador de banco de dados também aumenta o esforço de administração, por haver um número maior de objetos para administrar. Também existe um custo maior para realização de cópias de segurança, pelo fato da base ser maior. Deve-se considerar que o volume de registros de auditoria pode ser maior do que o volume de dados do sistema de informação;
- Independente da forma com que o sistema de auditoria foi projetado, há a necessidade de um componente que previna a evasão dele através de uso de interfaces de usuário ou ferramentas genéricas. Como os registros de auditoria que identificam as ações aplicadas sobre a base de dados são gerados a partir do aplicativo, qualquer atividade realizada através de outro meio, que não o aplicativo, não gerarão as evidências nos arquivos de auditoria. Também, se houver possibilidade de ação sobre a base de dados, por meios de acesso direto a ela, poderão ser violados os próprios arquivos de auditoria, e eliminadas evidências de ações praticadas através do aplicativo.

4.2 AUDITORIA BASEADA EM RECURSOS DE BANCO DE DADOS

Essa abordagem é bastante utilizada para complementar um sistema de auditoria que não seja muito abrangente ou implementar uma auditoria, quando não houver uma disponível. Os recursos de banco de dados utilizados são *triggers* e *stored procedures* [27].

Trigger é um mecanismo definido na base de dados e que é ativado por um evento de modificação ocorrido sobre uma tabela. Os eventos monitorados são inclusão, exclusão e modificação. As ações que um *trigger* pode executar são modificações adicionais através de nova inclusão, exclusão ou modificação, ou referenciando um *stored procedure*, que é um procedimento cadastrado na base de dados, e que pode implementar uma lógica complexa [3]. Alguns sistemas de informação possuem interfaces para a implementação de auditoria sobre a base de dados utilizando tais recursos do banco de dados. Toda a definição de *triggers, stored procedures* e tabelas necessárias para suportar a auditoria é automatizada [28].

São essas algumas características particulares dessa abordagem de sistema de auditoria:

- A principal vantagem percebida nessa abordagem está na independência dos programas que executam as modificações na base de dados. Toda lógica do sistema de auditoria é implementada sobre a definição dos objetos da estrutura de dados, mais especificamente tabelas, sendo responsabilidade do gerenciador de banco de dados executar os procedimentos que irão alimentar os arquivos de auditoria. Logo, a utilização de interfaces de usuário ou ferramentas genéricas para realização de ações sobre a base de dados também ficará registrada nos arquivos de auditoria;
- Como a definição da lógica está disponível na definição dos objetos da estrutura de dados, de forma aberta, pode-se realizar uma série de personalizações nos registros de auditoria. Porém esse tipo de abordagem torna-se inviável se a abrangência precisar ser muito extensa, sobre muitos objetos. A base de dados do sistema de informação sendo auditado pode ser composta de milhares de tabelas. O esforço para implementar *triggers* sobre

todos os objetos de interesse pode se tornar um impedimento. Uma evolução do sistema de informação, que represente mudança na estrutura de dados sendo monitorada por *triggers* implica na reconstrução do ambiente para reconhecer a nova situação;

- Uma amplitude de cobertura específica é imposta, e geralmente requer programação para efetivar uma mudança nas políticas de auditoria. Essa implementação atua no nível de uma base de dados única, necessitando ser instalado individualmente, seguindo rotinas de procedimento para novas necessidades. Deve haver controles que garantam esse sincronismo entre as diversas bases de dados;

- Nessa abordagem, assim como na anterior, ocorre um aumento nos custos de: armazenamento, para conservar os registros de auditoria sendo gerados pelos *triggers*; processamento adicional para as ações associadas às ações impostas sobre os objetos sendo monitorados pelo gerenciador de banco de dados pelos mecanismos de *triggers*; administração, devido ao enorme esforço do DBA em manter essa estrutura trabalhosa e complexa; realização de *backups* do banco de dados, pelo incremento considerável no tamanho da base de dados para armazenar os registros de auditoria;

- Também nessa abordagem, o armazenamento dos registros de auditoria se efetua na mesma base de dados, sob o mesmo gerenciamento de banco de dados, e o sincronismo entre os dados do sistema de informação e os dados do sistema de auditoria também fica garantido pelo ambiente transacional do banco de dados, e qualquer necessidade de recuperação do banco de dados também trará os dados do sistema de informação e os dados do sistema de auditoria a uma situação consistente, sem necessidade de algum controle para garantir esse sincronismo. Também não haverá registro de que esse fato ocorreu nos registros de auditoria.

4.3 AUDITORIA ENTRE OBJETOS

Essa abordagem explora a características de algumas ferramentas de desenvolvimento. Baseia-se na captura de mensagens trocadas entre objetos, referente a

sistemas orientados a objetos, que compõem o sistema de informação e seu registro nos arquivos de auditoria. Algumas características dessa abordagem são:

- A captura ocorre sem que o desenvolvedor de sistemas precise se preocupar com esse detalhe na elaboração do projeto do sistema. Assim a preparação dos registros para auditoria ocorre de forma independente e *on-line*;
- Também não se mostra muito apropriada para auditoria de integridade, pois não se baseia sobre as ações executadas sobre a base de dados, mas sim as intenções das ações expressas nas mensagens trocadas entre os objetos;
- Não existe mecanismo automatizado de sincronismo entre a versão do banco de dados existente e os registros de auditoria. Mesmo mensagens trocadas entre objetos, mas que não resultaram nas mudanças esperadas na base dados, estarão presentes nos registros de auditoria;
- A utilização de interfaces de usuário ou ferramentas genéricas para realização de ações sobre a base de dados não ficará registrada nos arquivos de auditoria.

Apesar de ser uma abordagem interessante, está restrita a poucas ferramentas que permitem implementa-la e ainda apresenta limitações [4].

4.4 RESUMO

Nesse capítulo foram apresentadas algumas abordagens tradicionais de auditoria de sistemas de informação, bem como apontadas as forças e franquezas inerentes a cada uma delas. É importante o reconhecimento dessas características para realizar a seguir a apresentação da motivação que levou a proposta de uma abordagem diferente, e poder relacionar possíveis vantagens e desvantagens ao que já existe.

5 METODOLOGIA PARA UMA ABORDAGEM INDEPENDENTE DE AUDITORIA DE SISTEMAS DE INFORMAÇÃO

Um mecanismo de monitoração é independente, se as duas condições seguintes são satisfeitas [4]:

1. A seleção de eventos a serem monitorados e a escolha das informações a serem registradas sobre a ocorrência de um evento são feitas sem o conhecimento dos desenvolvedores do sistema sendo monitorado;
2. O processo de monitoração não pode ter qualquer efeito no estado ou comportamento do sistema sendo monitorado, exceto, talvez, na velocidade de execução das operações.

Nenhum tipo de sensor precisa ser embutido no sistema para originar os registros que irão compor a trilha de auditoria, permitindo que o sistema evolua livremente conforme a necessidade.

Raramente vamos encontrar sistemas aplicativos que não estejam vinculados a um banco de dados relacional comercial e um sistema operacional robusto. Os sistemas operacionais e os sistemas gerenciadores de banco de dados possuem recursos para registrar eventos que ocorrem em seus ambientes, independente do projeto do sistema de informação, os quais servem de evidência para uma auditoria independente de sistema.

O subsistema de auditoria do sistema operacional provê ao administrador de sistemas os meios para registrar informações relevantes de segurança, que podem ser analisadas para detectar violações potenciais e reais das políticas de segurança de sistemas. O subsistema de auditoria tem três funções [9]:

1. Detecção de eventos;
2. Coleta de informações;
3. Processamento de informações.

O sistema gerenciador de banco de dados mantém detalhes de todas operações de atualização — em particular, valores anteriores e posteriores do objeto atualizado —

registrados em arquivos ou áreas de *log*. Assim, se for necessário desfazer alguma atualização em particular, o sistema pode usar a entrada de *log* correspondente para recuperar o objeto atualizado ao seu valor anterior [3].

O *log* consiste de duas porções: uma porção ativa ou *on-line* e uma porção arquivada ou *off-line*. A porção *on-line* é utilizada durante a operação normal do sistema, e registra detalhes das atualizações conforme elas acontecem. Quando a porção *on-line* se enche, seu conteúdo é transferido para a porção *off-line*, que — por ser sempre processada seqüencialmente — é geralmente guardada em fita magnética ou outra mídia externa.

A metodologia que será apresentada fará uma abordagem explorando os subsistemas de fronteira e banco de dados, conforme definido na fatoração dos sistemas em subsistemas funcionais, como fornecedores de evidências para a auditoria independente de sistemas de informação. Serão feitas considerações aprofundadas sobre esses subsistemas de controle e a definição pelos subsistemas mais adequados para essa abordagem metodológica. Finalmente, será estabelecida a metodologia proposta em suas diversas fases, desde a coleta de evidências, até a fase de avaliação delas, para permitir identificar as ações ocorridas na base de dados provocadas pelo sistema de informações.

5.1 CONTROLE DE FRONTEIRA

O subsistema de fronteira estabelece a interface entre o suposto usuário de um sistema de computador e o sistema de computador em si. Quando um usuário senta em frente a um terminal, liga o terminal, e começa o procedimento inicial de identificação com o sistema operacional, as funções do subsistema de fronteira estão sendo desempenhadas. Controles no subsistema de fronteira têm três propósitos principais:

1. Estabelecer a identidade e autenticidade dos supostos usuários em um sistema de computador;
2. Estabelecer a identidade e autenticidade dos recursos que os usuários desejam empregar;

3. Restringir as ações tomadas pelos usuários que obtêm recursos de computador a um conjunto de ações autorizadas.

Dois fatores conduziram a um aumento no uso e reforço dos controles de fronteira. Primeiro, a ampla utilização de sistemas distribuídos resultou em muitos usuários dispersos fisicamente, via *WANs*, *LANs*, e processamento cliente-servidor. Maior segurança deve ser colocada nos controles baseados em computador para limitar e monitorar as ações tomadas pelos usuários. Segundo. O rápido crescimento dos sistemas de comércio eletrônico resultou em trabalho substancial, assumido em medidas para identificar e autenticar as partes que trocam valores através desses sistemas. Portanto, controles de fronteira são atualmente alguns dos mais complexos controles encontrados nos sistemas de computador.

5.1.1 Controles de Trilha de Auditoria

Dois tipos de trilha de auditoria devem existir: uma trilha de auditoria de contabilização, que mantém um registro de eventos dentro do subsistema, e uma trilha de auditoria de operações, para manter um registro do consumo de recurso associado com cada evento do subsistema.

5.1.1.1 Trilha de Auditoria de Contabilização

Todos eventos orientados à aplicação que ocorrem dentro de subsistema de fronteira deveriam ser registrados na trilha de auditoria de contabilização. Os seguintes tipos de dados associados com um evento poderiam ser guardados:

1. Identidade do suposto usuário do sistema;
2. Informação de autenticação fornecida;
3. Recursos requeridos;
4. Privilégios de ações requeridos;
5. Identificador de terminal;
6. Hora de início e fim;
7. Número de tentativas de *sign-on*;

8. Recursos providos/negados; e
9. Privilégios de ação permitidos/negados.

Esses dados permitem aos administradores ou auditores recriar as séries temporais de eventos que ocorrem quando um usuário tenta obter acesso e utilizar recursos do sistema.

Periodicamente a trilha de auditoria deveria ser analisada para detectar quaisquer fraquezas de controle no sistema. Tanto análises manuais como automatizadas poderiam ser assumidas. Por exemplo, o administrador poderia pesquisar a trilha de auditoria por eventos não usuais. Alternativamente, um programa, como um sistema de detecção de intrusão, poderia ser usado. Esses sistemas monitoram os usuários para determinar se o comportamento atual está de acordo com o comportamento passado. A trilha de auditoria é a fonte primária para a construção de um perfil do comportamento passado.

5.1.1.2 Trilha de Auditoria de Operações

Muito dos dados coletados na trilha de auditoria de contabilização também serve aos propósitos da trilha de auditoria de operações. Por exemplo, registros de hora de início e fim e de recursos pedidos também facilitam análises de utilização de recursos dentro do subsistema. Assim como com a trilha de auditoria de contabilização, certos tipos de consumo de recurso poderiam ser de interesse como base para detecção de atividades não autorizadas. Por exemplo, um sistema de detecção de intrusão poderia monitorar a quantidade de tempo de processador consumida por um usuário para detectar desvios da quantidade de tempo de processador pedida pelo usuário no passado.

5.2 CONTROLE DE BANCO DE DADOS

O subsistema de base de dados provê funções para definir, criar, modificar, excluir, e ler dados em um sistema de informações. Historicamente, o tipo primário de dados mantido em um subsistema de base de dados tem sido dados declarativos — ou seja, dados que descrevem os aspectos estáticos dos objetos do mundo real e associações entre esses objetos. O subsistema de base de dados poderia também ser usado para manter dados procedurais — ou seja, dados que descrevem os aspectos dinâmicos dos objetos do mundo real e as associações entre esses objetos. Quando

ambos, dados declarativos e procedurais, são armazenados, a base de dados é algumas vezes chamada de base de conhecimento para refletir o maior poder dos dados mantidos no subsistema de base de dados.

Inicialmente, os principais componentes no subsistema de base de dados eram os programas de aplicação que definiam, criavam, modificavam, e excluíam dados, o sistema operacional que desempenhava as operações básicas de entrada/saída para mover dados de e para as várias mídias de armazenamento, o processador central e a memória principal onde essas atividades eram desempenhadas, e a mídia de armazenamento que mantinha cópia permanente ou semipermanente dos dados.

5.2.1 Controles de Trilha de Auditoria

A trilha de auditoria em um subsistema de base de dados mantém a cronologia de eventos que ocorreram ou na definição da base de dados ou na própria base de dados. Em muitos casos, um conjunto completo de eventos deve ser registrado: criações, modificações, exclusões, e buscas. Senão, poderia ser impossível determinar como a definição da base de dados ou a base de dados chegou à situação corrente, ou quem, através de uma transação de busca, contou com alguma situação passada da definição da base de dados ou da base de dados.

5.2.1.1 Trilha de Auditoria de Contabilização

Para manter a trilha de auditoria de contabilização em um sistema de aplicação, o subsistema de base de dados deve assumir três funções. Primeira, deve ligar uma única marca de tempo para todas as transações aplicadas contra a definição da base de dados ou a base de dados. Essa marca de tempo tem dois propósitos:

1. Confirma que a transação atingiu a definição da base de dados ou a base de dados, e;
2. Identifica uma posição única da transação na série temporal de eventos que ocorreu a um item de dado na definição da base de dados ou na base de dados.

Segunda, o subsistema de base de dados deve ligar imagens anteriores e posteriores do item de dado, contra o qual a transação é aplicada, na entrada de trilha de

auditoria para a transação. Se a transação modifica um valor de item de dado existente, o valor do item de dado antes que ele seja atualizado e o valor do item após a atualização devem ser armazenados na entrada da trilha de auditoria da transação. Quando transações de inserção, exclusão, e busca ocorrem e não mudam os valores de item de dado existentes, um indicador deve ser ativado para indicar que a imagem anterior e posterior são a mesma. Essas imagens têm dois propósitos:

1. Facilitam pesquisas na trilha de auditoria porque os efeitos da transação na definição da base de dados ou na base de dados pode ser determinada imediatamente da entrada da trilha de auditoria, e;
2. Elas provem redundância para a marca de tempo porque a exclusão fraudulenta de uma entrada em uma trilha de auditoria ou a alteração fraudulenta de uma marca de tempo pode ser detectada através de desacordo entre a imagem posterior de uma transação e a imagem anterior da transação subseqüente.

Terceira, porque as entradas de trilha de auditoria requerem armazenamento permanente ou semipermanente, o subsistema de base de dados deve prover facilidades para definir, criar, modificar, excluir, e buscar dados em uma trilha de auditoria. Usualmente a trilha de auditoria não deveria ter de ser modificada porque ela é para contar uma história real do que aconteceu à base de dados.

Duas situações podem surgir, entretanto, na qual modificações na trilha de auditoria são necessárias. Primeira, o sistema de aplicação atualizando a base de dados processa os dados erroneamente. Como resultado, a trilha de auditoria é uma história de operações incorretas na base de dados. Desfazer resultados errôneos não é suficiente se alguém acessa a informação errônea na trilha de auditoria e toma decisões incorretas baseadas nessa informação. Segunda, os processos que criaram a trilha de auditoria poderiam ser falhos. Nesse caso, a trilha de auditoria não é uma história verdadeira do que aconteceu à base de dados. Novamente, decisões incorretas poderiam ser tomadas com base em trilha de auditoria errônea. Em ambos os casos, o melhor seria modificar a trilha de auditoria, de forma que decisões tomadas posteriormente, com base nos dados contidos na trilha de auditoria, não estejam infectadas por dados errôneos.

5.2.1.2 Trilha de Auditoria de Operações

A trilha de auditoria de operações no subsistema de base de dados mantém a cronologia de eventos de consumo de recursos que afetam a definição da base de dados ou a base de dados. Com base na trilha de auditoria de operações, administradores de dados ou administradores de base de dados podem tomar duas decisões. Primeira, considerando os tempos de resposta ou a quantidade de recursos consumidos quando as transações são aplicadas contra a base de dados, a necessidade de reorganização da base de dados poderia tornar-se clara. Segunda, dados de consumo de recurso poderiam indicar que os processos que aplicam transações na definição da base de dados ou na base de dados necessitam ser reestruturados ou reescritos.

5.2.2 Controles de Existência

Toda ou uma parte da base de dados pode ser perdida (destruída ou corrompida) através de cinco tipos de falha:

1. Erro de programa de aplicação: Um programa de aplicação pode atualizar dados incorretamente porque ele contém um erro. Usualmente só ocorre dano localizado à base de dados porque o programa atualiza apenas um subconjunto de dados na base de dados;
2. Erro de *software* de sistema: *Software* de sistema, tal como um sistema operacional, sistema gerenciador de base de dados, monitor de telecomunicações, ou programa utilitário, poderia conter um erro. O erro poderia conduzir a atualizações errôneas da base de dados, corrupção de dados, ou travamento de sistema. Se dano local ou global ocorre à base de dados, depende da natureza do erro.
3. Falha de *hardware*: Apesar da grande confiança da maioria dos componentes de *hardware*, falhas ainda podem ocorrer. A falha poderia ser menor ou transitória, e como resultado apenas dano localizado ocorre na base de dados. A falha poderia ser séria e permanente, entretanto, nesse caso um dano abrangente poderia ocorrer na base de dados, como no caso de travamento de disco com destruição do conteúdo do disco;

4. Erro de procedimento: Um operador ou usuário poderia cometer um erro de procedimento que danificasse a base de dados. Por exemplo, operadores poderiam cometer uma ação incorreta ao recuperar um travamento de sistema, ou um usuário poderia fornecer parâmetros incorretos para a execução de uma atualização. Se o dano é local ou global, depende da natureza do erro cometido.
5. Falha de ambiente: Tais como inundação, incêndio, ou sabotagem podem ocorrer. Freqüentemente danos abrangentes à base de dados ocorrem. O armazenamento de arquivos distante do local é essencial para restaurar a base de dados após muitos tipos de falha de ambiente.

Controles de existência em um subsistema de base de dados devem restaurar a base de dados no evento de perda. Eles abrangem uma estratégia de *backup* e uma estratégia de recuperação. Todas estratégias de *backup* envolvem manter uma versão anterior da base de dados e um *log* de transações ou mudanças na base de dados.

Logging envolve registrar uma transação que modifica a base de dados ou uma imagem do registro modificado por uma ação de atualização. O *log* pode então ser usado para recuperar a base de dados ao ponto de falha. A estratégia alternativa, que é requerer aos usuários para submeter novamente as transações para recuperar a base de dados ao ponto de falha, poderia não ser viável por uma série de razões. Primeira, o tempo parado requerido para os usuários submeterem novamente todas as transações poderia ser inaceitável. Segunda, a recuperação da base de dados poderia exigir que as transações fossem submetidas em uma seqüência específica. Se os usuários não registraram as transações em uma seqüência de tempo ou múltiplas fontes de entrada existem, a obtenção da seqüência cronológica para repetir a submissão das transações poderia ser impossível. Terceira, os dados da transação poderiam não ser recebidos diretamente de um usuário. Em vez disso, poderiam ser gerados automaticamente por um programa ou recebidos de outro computador. Nesses casos, alguma forma de *logging* deve ocorrer.

Há vários tipos de registro de *log*. Um registro de atualização de *log* descreve uma única escrita do banco de dados e tem os seguintes campos [23]:

- Identificador de transação: identifica unicamente a transação que realiza a operação de escrita;
- Identificador de item de dado: identifica unicamente o item de dado escrito, sendo normalmente a localização do item de dado no disco;
- Valor antigo: valor do item de dado antes da escrita;
- Valor novo: valor do item de dado após a escrita.

Há outros registros de *log* para eventos significativos durante o processamento de transação, como início da transação, sua efetivação ou aborto.

Se um programa de atualização cria uma nova versão física de um arquivo, a versão prévia e o arquivo de transações usado durante uma atualização podem ser usados para propósitos de *backup*. Se a atualização ocorre por sobreposição, entretanto, periodicamente uma descarga da base de dados deve ser feita, e um *log* das mudanças na base de dados desde a descarga também precisa ser mantido.

Estratégias de recuperação tomam duas formas. Primeira, a situação atual da base de dados poderia ter que ser restaurada se a base de dados inteira ou uma porção da base de dados for perdida. Essa tarefa envolve uma operação de reparação para frente usando uma versão anterior ou descarga da base de dados e um *log* de transações e mudanças que ocorreram na base de dados desde que a descarga foi feita. Segunda, uma situação anterior da base de dados poderia ter que ser restaurada porque a situação corrente da base de dados está inválida. Essa tarefa envolve uma operação de reparação para trás para desfazer as atualizações que causaram a corrupção da base de dados. Um *log* das mudanças na base de dados é usado para restaurar a base de dados para uma situação anterior válida.

5.3 UMA PROPOSTA DE ABORDAGEM METODOLÓGICA

A abordagem ora apresentada foca-se, dentro da definição de segurança de informação, na integridade da informação, não como controle ou monitoração em tempo real do que ocorre sobre a informação, mas na disponibilização de recursos de auditoria, para rastreio dos fatos ocorridos sobre a informação, caso haja suspeita sobre a sua situação ou para identificar utilização inadequada ou abusiva de privilégios de acesso.

As abordagens citadas anteriormente apresentam características adequadas para a monitoração e controle em tempo real, além de gerar registros para auditoria do sistema de informação. Porém percebe-se claramente que elas estão sujeitas às mudanças evolutivas dos aplicativos, e apresentam um custo elevado para se adequar a essas mudanças.

Dentro do enfoque de abordagem metodológica voltada para a auditoria de sistemas de informação, especificamente orientada para a rastreabilidade das atividades sobre a informação, também apresenta a desejável característica de ser independente do sistema de informação, por buscar as evidências explorando os registros de controles internos do ambiente computacional utilizado para suportar o sistema de informação. Assim, sem a necessidade de embutir sensores nos sistemas de informação para originar os registros que irão compor a trilha de auditoria, os sistemas podem evoluir livremente conforme a necessidade, sem comprometer a geração de evidências para auditoria.

Os controles internos, geradores dos registros necessários para a aplicação da abordagem metodológica proposta, são os controles de fronteira, disponibilizados pelo sistema operacional, e o controle de existência do banco de dados (*logging*), disponibilizado pelo sistema gerenciador de banco de dados.

A coleta e o tratamento desses registros, e o relacionamento entre informações do processo e de suas requisições ao gerenciador de banco de dados possibilitarão responder algumas perguntas:

- Quem e quando executou determinado aplicativo?
- Que objetos do banco de dados foram modificados por determinado aplicativo ou determinado usuário?
- Que usuários ou aplicativos realizaram modificações sobre determinado objeto em determinado período de tempo?

Caracterizando por tipologia, essa abordagem se classifica como indutiva, partindo de um caso específico e propondo sua generalização. Para sua definição foram usados métodos dedutivos, através da escolha da possibilidade considerada mais adequada, dentro de princípios lógicos e resultados empíricos.

5.3.1 Descrição da Abordagem Metodológica

A metodologia proposta tem por base a integração dos registros do subsistema de auditoria do sistema operacional, onde são executados os aplicativos que modificam as informações da base de dados, e do *log* do banco de dados, onde ficam registradas as modificações ocorridas na base de dados, originando uma nova base de dados, conforme esquema da figura 5.1, a qual sustenta a auditoria independente dos sistemas de informação sendo utilizados nesse ambiente.

A estrutura é modular e particiona o processo de construção da base de auditoria de forma a simplificar o entendimento do fluxo das informações, bem como reaproveitar alguns módulos que são genéricos, enquanto outros serão específicos ao sistema operacional e gerenciador de banco de dados.

Nesse primeiro momento, a abordagem metodológica proposta ficará restrita a um ambiente cliente-servidor, onde a parte cliente será um servidor de aplicativos, ou seja, um ambiente centralizado de processamento. Porém não haverá restrição à plataforma sendo utilizada.

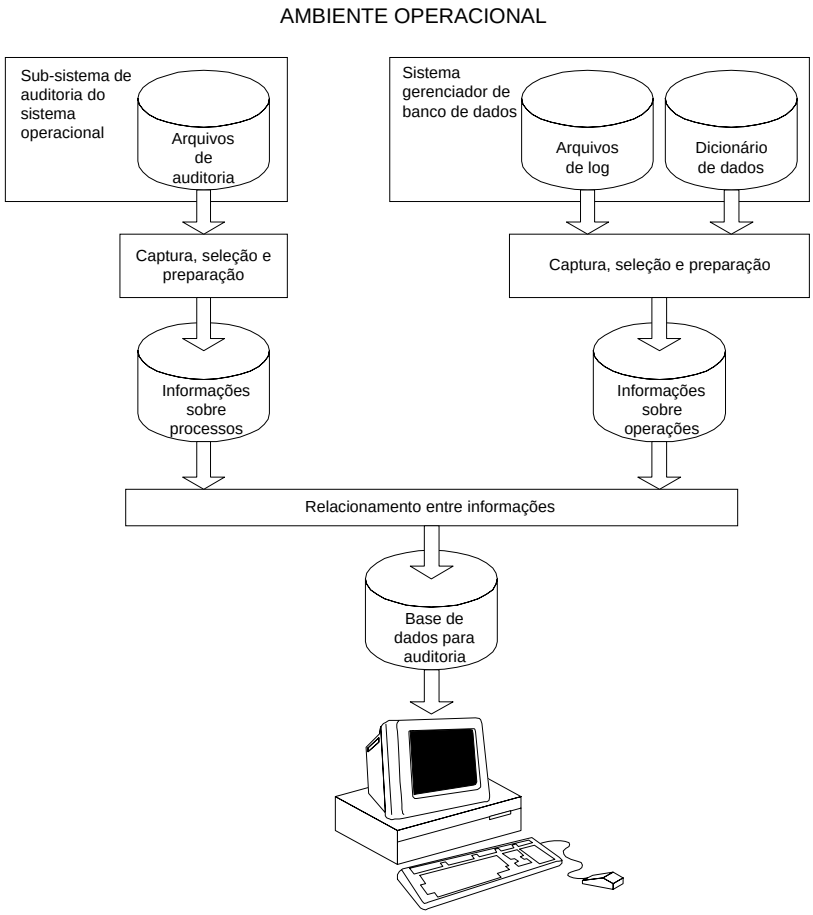


Figura 5.1: Estrutura da metodologia proposta

A metodologia define etapas de captura de registros das fontes citadas acima e da construção de um dicionário para identificação dos diversos objetos sendo monitorados, de forma a facilitar o reconhecimento dos fatos registrados, quando da realização de consultas pelo auditor. Por essas características, fica evidente que a auditoria não terá informações em tempo real, mas sim após a preparação da base para consultas, ocasionando uma demora entre o fato e a identificação do mesmo.

5.3.2 Dicionário de Dados Temporal

Antes de coletar qualquer dado sobre as atividades do sistema operacional e do gerenciador de banco de dados, é necessário preparar o dicionário de dados dos objetos presentes em todo o processo, lembrando que tais objetos mudam devido a evoluções do

sistema de informação durante o tempo. Assim, cada entrada do dicionário deve estar representa num período de validade.

As seguintes estruturas do dicionário de dados são sugeridas, objetivando facilitar o entendimento para quem realizar as consultas de auditoria:

- `usuario`, que contêm o *login* do usuário, o nome do usuário e outros dados considerados de interesse para facilitar a apresentação do usuário para o auditor. Deve sofrer manutenção a cada modificação ocorrida na relação de usuários do sistema de informação;
- `programa`, que contêm o comando de chamada, uma breve descrição do programa e outros dados que possam facilitar e complementar a identificação do programa. Deve sofrer manutenção a cada modificação ocorrida na relação de programas que compõem o sistema de informação;
- `tabela`, que relaciona a identificação interna da tabela para o gerenciador de banco de dados, o nome da tabela, como é conhecida externamente, e outros dados que possam melhorar o entendimento do auditor. Deve sofrer manutenção a cada mudança sobre os atributos dos objetos que compõem a base dados do sistema de informação. Os dados são obtidos do dicionário de dados da base de dados;
- `coluna`, que relaciona as colunas das tabelas da base de dados do sistema de informação, e seus atributos. Alem dessa tradução, pode conter outros dados que permitam facilitar o reconhecimento do dado contido na coluna. Deve sofrer manutenção a cada mudança nas estruturas do banco de dados que armazena os dados do sistema de informação. Os dados são obtidos do dicionário de dados da base de dados.

5.3.3 Obtenção dos Dados dos Processos do Sistema Operacional

Cada sistema operacional possui particularidades em sua forma de configurar e disponibilizar dados de auditoria. O subsistema de auditoria do sistema operacional deve ser configurado para produzir dados referentes a:

- O usuário que se autenticou no sistema operacional;

- Identificação do processo no sistema operacional;
- Horário de execução do processo;
- Comando que causou a criação do processo no sistema operacional.

Se o subsistema de auditoria permitir, poderá ser aplicado algum filtro que restrinja a geração de dados para usuários específicos, aplicativos específicos ou período de tempo específico. Assim é possível se restringir a captura de dados a um universo a ser auditado para situações consideradas críticas.

Além desses dados é necessário possuir dados referentes ao usuário, para facilitar a identificação. Pelo menos o nome é desejável. Também a aplicação deve possuir uma descrição da função que desempenha no sistema de informação. São necessárias também referências de validade desses dados, pois pode haver mudanças nesses atributos em função do tempo. Assim, ao relacionar as informações na base de auditoria, deverá ser relacionada a data do evento com o período de validade da informação complementar. Essas informações devem ser mantidas na estrutura de dicionário de dados temporal, como descrito anteriormente.

Deve haver um aplicativo que realize o tratamento dos dados gerados pelo subsistema de auditoria do sistema operacional, de forma a reduzir os dados ao que é de interesse para a auditoria, e associar outros dados complementares, como visto acima. Se o subsistema não permitir filtragens, ou não foi configurado para isso, esse aplicativo pode ser preparado para essas funções de uma forma personalizada. Mesmo não sendo o objetivo dessa abordagem, esse aplicativo poderia também ser preparado para emitir alertas ou relatórios sobre fatos considerados relevantes.

Como resultado desse módulo, obtém-se um armazenamento intermediário de dados referentes aos processos relacionados aos aplicativos que serão alvo de auditoria futura. Com esses dados já é possível responder as seguintes perguntas:

- Quem executou determinado programa?
- Que programas foram executados por determinado usuário?
- Quando determinado programa foi executado?

5.3.4 Obtenção dos Dados das Atividades no Banco de Dados

Um pré-requisito para essa abordagem metodológica é que o ambiente transacional do banco de dados esteja ativo, de forma a gerar os registros no arquivo de *log*. É quase inconcebível um sistema de informação operar sem a preocupação de garantir a integridade da informação, caso ocorra alguma falha localizada, como o cancelamento de algum programa, ou ocorra uma falha global, por queda de energia, pane no sistema gerenciador de banco de dados, sistema operacional, *hardware* ou mesmo o aplicativo.

Como visto anteriormente, em um arquivo de *log* há vários tipos de registros. Um registro de atualização de *log* descreve uma única escrita do banco de dados e contém um identificador da transação, um identificador do item de dado, o valor antigo e o valor novo. Há outros registros de *log* para arquivar eventos significativos durante o processamento de uma transação, como o início da transação, sua efetivação ou aborto.

Para obter os resultados esperados por essa metodologia, do arquivo de *log* será necessário coletar as seguintes informações:

- Tipo da operação;
- Identificação da transação do banco de dados;
- Identificação do usuário do banco de dados;
- Horário de execução da operação no banco de dados;
- Identificação do objeto sendo alvo da operação no banco de dados;
- Detalhes da operação sendo executada no banco de dados.

Geralmente os gerenciadores de banco de dados já possuem algum programa utilitário para extrair e decodificar os registros do arquivo de *log*. Aqui também, se houver recursos de filtragem, pode-se restringir o universo de objetos sobre os quais se guardarão evidências de ações praticadas para auditoria futura. Caso não exista essa ferramenta pronta, deverá ser analisada a documentação disponível para construir um programa que realize essa operação. Caso não haja documentação disponibilizada, no caso de gerenciadores de banco de dados de código aberto, será necessário analisar o código relativo ao tratamento das funções de *logging*. No caso de produtos com código

proprietário, deverá ser realizada engenharia reversa sobre os arquivos de *log* e deduzida sua estrutura.

As informações provenientes do arquivo de *log* foram projetadas para atender à necessidade interna do gerenciador de banco de dados para garantir a integridade dos dados e permitir a recuperação do banco de dados a situações consistentes. Estão, portanto, vinculadas à visão interna do banco de dados, e, portanto não se preocupam com o entendimento do usuário ou DBA. Uma característica comum é a identificação dos objetos através de valores definidos internamente e associados ao nome do objeto, como ele é conhecido externamente. Para realizar essa tradução entre as visões externa e interna existe um dicionário de dados, que além da associação entre identificações interna e externa, guarda ainda uma série de atributos sobre os objetos. De interesse para a presente abordagem metodológica são os atributos referentes às tabelas e às colunas associadas às tabelas.

É necessário preservar esse dicionário. Assim será possível apresentar para o auditor as informações, coletadas no arquivo de *log*, de uma forma compreensível. Uma característica comum aos sistemas de informação é a evolução do aplicativo, decorrente de novas necessidades. Isso geralmente resulta em mudanças nas estruturas de dados do banco de dados. Também aqui é necessário anexar uma validade aos itens que compõem o dicionário. Quando forem relacionadas as informações provenientes do arquivo de *log* com as informações do dicionário de dados salvo, deverá ser relacionada a data da operação no banco de dados com o período de validade da informação do dicionário de dados, para garantir uma identificação correta dos objetos e seus atributos. Esses dados também comporão o dicionário de dados temporal descrito anteriormente, e devem ser mantidos atualizados, conforme as mudanças ocorram sobre as estruturas de dados que compõem a base de dados dos sistemas de informação que serão auditados.

Torna-se necessário construir aplicativos para preparação dos dados extraídos do arquivo de *log* e preparação do dicionário de dados com características temporais. Esses aplicativos podem agregar funções adicionais para emissão de alarmes e relatórios sobre fatos considerados relevantes, embora não seja esse o objetivo dessa abordagem. Também se podem implementar funções de filtragens personalizadas.

Nesse módulo, obtém-se um armazenamento intermediário de dados referentes às operações realizadas sobre os dados dos objetos e sobre os objetos no banco de dados pelos aplicativos que atuam sobre o banco de dados.

Esses dados, nessa fase, já permitem conseguir algumas respostas:

- Quem executou operações na base de dados?
- Sobre quais estruturas esse usuário realizou operações?
- Que tipos de operações foram realizados?
- Quais os valores anteriores e posteriores a essas operações?

5.3.5 Criação da Base de Dados para Auditoria

Esse módulo tem como entrada os dados intermediários armazenados resultantes dos dois módulos anteriores. É sua função associar as informações referentes aos processos relacionados aos aplicativos executados e as informações referentes às operações realizadas sobre os dados dos objetos e sobre os objetos na base de dados pelos aplicativos.

Esse módulo representa, estrategicamente, o mecanismo que permitirá aos auditores obter as respostas às suas perguntas.

A dificuldade está em encontrar um atributo de relacionamento entre as duas entidades. Para cada processo cliente é criado um processo servidor no ambiente do servidor de banco de dados para atender às solicitações do processo cliente. Todas as solicitações de operações ficam associadas a esse processo no servidor.

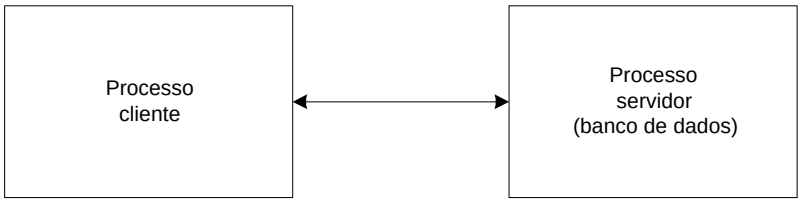


Figura 5.2: Comunicação entre processos cliente/servidor

O arquivo de *log* está internamente associado ao gerenciador de banco de dados. O seu objetivo é disponibilizar registros para preservar a consistência dos dados e

permitir a recuperação do banco de dados. Nessa atividade não existe nenhum aplicativo externo operando sobre a base de dados e, portanto não existe a necessidade de preservar dados do aplicativo que originou as operações sobre a base de dados. Dados que são preservados e devem orientar na forma de relacionar as entidades são a identificação do usuário associado à transação do banco de dados e os horários em que as operações foram submetidas ao gerenciador de banco de dados. Para esse último dado, caso o aplicativo cliente e o gerenciador de banco de dados não estejam no mesmo computador, deve estar ativo algum mecanismo de sincronismo dos relógios dos computadores. Sob essa perspectiva, os dados referentes às operações realizadas no banco de dados pela transação associada ao mesmo usuário que ativou o aplicativo imediatamente antes do início da transação, podem ser relacionadas.

Para os sistemas que se conectam ao gerenciador de banco de dados sempre como o mesmo usuário, e utilizam um mecanismo interno de autenticação, será necessário identificar nos registros de *log* o usuário através das operações realizadas nos objetos do banco de dados responsáveis pelo controle de acesso ao sistema de informação.

Para situações em que um mesmo usuário pode realizar múltiplas conexões no gerenciador de banco de dados, torna-se necessário buscar alguma evidência adicional, como através da ativação de alguma opção do subsistema de auditoria do gerenciador de banco de dados que permita distinguir a procedência da solicitação ao gerenciador do banco de dados. Porém isso não será abordado nesse momento. Será considerada apenas uma conexão por usuário.

Uma vez definida a forma como as partes serão relacionadas, sugere-se a utilização de um identificador seqüencial ou *time-stamp* como atributo de relacionamento, facilitando a construção de pesquisas futuras e reduzindo o nível de processamento para resolver as solicitações à base de dados de auditoria. Para objetivos de rastreabilidade é sugerido ignorar os processos que abortaram e que não estão refletidos em ações permanentes na base de dados.

5.3.6 Interface de Consulta da Base de Dados para Auditoria

Nesse módulo já se tem a base de dados de auditoria construída e disponível para consultas. O que se tem, na verdade, é uma base com informações adequadas para o pessoal de processamento de dados, com registros dos aplicativos executados pelos usuários e, associados a esses registros, os dados das operações realizadas por eles sobre a base de dados. Não fica claro para um auditor saber que um determinado programa executou determinada atividade sobre uma determinada coluna de uma determinada tabela de uma certa base de dados. Tanto o auditor, quanto o administrador a quem ele deverá reportar qualquer suspeita, entendem de regras de negócio e gostariam de ter respostas relacionadas a essas regras.

Uma forma simples de resolver essa complexidade é a utilização de visões (*views*) do banco de dados. Cada visão terá embutida uma consulta que agregará os dados necessários para representar uma determinada regra de negócio. Por exemplo, para visualizar uma alteração de salário será criada uma visão que relacionará a operação de atualização sobre o objeto tabela de funcionários, mais especificamente sobre o atributo salário dessa tabela. O resultado será a relação de usuários, aplicativos e horários que estão associados com a execução dessa regra de negócio.

Ainda sobre as visões poderão ser sobrepostos outros critérios de seleção que restringirão a busca a somente o que interessa, dependendo da ferramenta de consulta que for utilizada para consultas.

É claro que os sistemas de informação atuais, principalmente os de gestão de negócio são enormes em volume de programas e estruturas de dados. Porém o processo de auditoria, quando focado em atividades de rastreamento, não tem a pretensão de verificar tudo em uma única ação, pelo contrário, tem necessidades bem específicas. Isso possibilita que as visões representativas das regras de negócio possam ser construídas conforme a necessidade.

5.3.7 Aplicação da Metodologia em Ambiente de Produção

A característica modular dessa abordagem metodológica permite sua utilização em ambiente de produção de acordo com certos critérios de necessidade. Por não ser

uma solução em tempo-real, certas etapas podem ser processadas em períodos de maior ociosidade de processamento.

Também certos módulos podem ter sua execução dispensada, até que haja necessidade de auditar alguma atividade suspeita. Serão somente realizadas as atividades relacionadas aos módulos anteriores, ou talvez nenhum módulo seja executado, sendo apenas feito o armazenamento dos dados capturados dos controles internos do subsistema de auditoria de sistema operacional ou dos arquivos de *log* gerados pelo gerenciador de banco de dados, para um processamento futuro que se faça necessário. Porém, o dicionário de dados temporal deverá ser mantido sempre atualizado.

Nesse ponto se percebe a importância da disponibilização de recursos de filtragem nos aplicativos que processam as informações capturadas. Somente seriam considerados os dados relacionados às atividades suspeitas. Porém isso não é imprescindível. A restrição dos critérios de pesquisa poderá ser informada na ferramenta de consulta utilizada.

5.4 RESUMO

Nesse capítulo foram detalhados os subsistemas de fronteira e do banco de dados, sendo definidos como adequados para a tarefa de coleta de evidências. No subsistema de fronteira optou-se pelo controle de trilha de auditoria, enquanto que no subsistema de banco de dados, foi considerado o controle de existência (*log*) como suficiente para os objetivos propostos. Foi apresentada a abordagem metodológica proposta, esquematicamente de forma modular, sendo cada módulo responsável por uma fase do tratamento da informação. Isso posto, será apresentado no próximo capítulo um experimento breve, que deverá ser suficiente para demonstrar a viabilidade e aplicabilidade da metodologia ora em questão.

6 APLICAÇÃO DA METODOLOGIA

Para demonstrar a aplicabilidade da abordagem metodológica proposta, será realizado um experimento, utilizando um sistema de aplicação comercial desprovido de módulo específico de auditoria, e comprovado, em conformidade com as especificações da metodologia, que é possível realizar uma auditoria futura, rastreando as atividades que foram executadas, de forma independente do sistema de informação.

O sistema de informação utilizado é o Logix versão 4.00, *software* de ERPII da empresa Logocenter Tecnologia de Informática S.A. A plataforma de execução para o sistema de informação é composta por equipamento IBM RS/6000 (processador risc), sistema operacional AIX versão 4.3.2 (Unix IBM) e banco de dados relacional Informix Dynamic Server versão 7.31.UC2. O aplicativo foi desenvolvido utilizando a linguagem de programação Informix 4GL versão 4.13.UD2. A seqüência de operações realizadas no sistema de informação é apresentada no anexo 1.

6.1 DICIONÁRIO DE DADOS TEMPORAL

Antes de coletar qualquer dado sobre as atividades do sistema operacional e do gerenciador de banco de dados, é necessário preparar o dicionário de dados dos objetos presentes em todo o processo.

Para esse experimento foram criadas as seguintes estruturas para o dicionário de dados, conforme previsto na metodologia, objetivando facilitar o entendimento para a realização das consultas de auditoria:

- usuário, que contêm apenas o *login* e o nome do usuário;
- programa, que contêm o comando de chamada e uma breve descrição do programa;
- tabela, que relaciona a identificação interna da tabela para o gerenciador de banco de dados e o nome da tabela, como é conhecida externamente. Os dados são obtidos do dicionário de dados da base de dados [29];

- coluna, que relaciona as colunas das tabelas da base de dados do sistema de informação, e seus atributos. Os dados são obtidos do dicionário de dados da base de dados [29].

No anexo 2 são apresentados os esquemas e conteúdos para as estruturas que compõem o dicionário de dados necessário para esse experimento.

6.2 COLETA DE DADOS DO SUBSISTEMA DE AUDITORIA DO SISTEMA OPERACIONAL

Essa primeira etapa consiste na captura de dados gerados a partir do subsistema de auditoria do sistema operacional. O subsistema deve ser configurado para monitorar e registrar os eventos de interesse para a auditoria futura. Os objetos de interesse para esse experimento são os programas do sistema de informação que serão executados durante o processo de operação do sistema.

O anexo 3 documenta a configuração necessária para esse experimento. Também faz parte do anexo a saída do programa de processamento dos registros obtidos a partir dessa monitoração. Observa-se no relatório uma série de ocorrências idênticas referentes aos acessos feitos aos arquivos que contêm os programas, que sumarizadas apresentam os dados necessários para identificar:

- usuário (`real`);
- processo do sistema operacional (`process`);
- data e hora de execução do processo (`time`);
- comando que deu origem ao processo (`command`).

O resultado intermediário dessa etapa é apresentado a seguir (figuras 6.1 e 6.2):

Nome do campo	Tipo de dados	Descrição
id_processo	Número	Identificação do processo para o sistema operacional
dat_processo	Data/Hora	Data de execução do processo
cod_usuario	Texto	Código de identificação do usuário
comando	Texto	Comando que iniciou o processo

Figura 6.1: Esquema dos dados intermediários dos processos

id_processo	dat_processo	cod_usuario	comando
11130	30/04/02 15:36:29	informix	fglgo /home/sidney/men0000.4gi
13316	30/04/02 15:39:29	informix	fglgo /home/sidney/men0010.4gi

Figura 6.2: Conteúdo dos dados intermediários dos processos

6.3 COLETA DE DADOS DO SUBSISTEMA DE CONTROLE DE EXISTÊNCIA DO BANCO DE DADOS

Essa etapa deve ser conduzida praticamente em paralelo com a primeira. Nela serão capturados os dados dos registros derivados das operações submetidas ao gerenciador de banco de dados pelos programas do sistema de informação. Esses dados serão extraídos dos arquivos de *log* do banco de dados. Como esses arquivos são produzidos automaticamente num banco de dados com ambiente transacional ativado, não há necessidade de nenhuma configuração adicional.

A observação sobre o paralelismo aparente entre essa etapa e a anterior deve-se ao fato da geração dos dados de interesse para a abordagem metodológica ocorrerem de forma inter-relacionada, ou seja, a execução de um programa resulta na submissão de uma série de operações ao gerenciador de banco de dados.

O anexo 4 apresenta a saída produzida pelo utilitário do gerenciador de banco de dados responsável pela extração e processamento dos dados registrados nos arquivos de *log* do banco de dados. Cada instrução do tipo `BEGIN` contém os dados que identificam [29]:

- processo servidor (9ª coluna);
- transação (4ª coluna);
- usuário (10ª coluna);
- data e hora correspondentes a uma conexão de um aplicativo cliente do servidor de banco de dados (7ª e 8ª colunas).

Todas as demais operações estão apresentadas seqüencialmente dentro de cada transação pertencente ao processo interno do gerenciador de banco de dados relativo à conexão do aplicativo cliente. Nesse experimento foram consideradas somente as operações de inserção [registro tipo `HINSERT`], exclusão [registro tipo `HDELETE`] e

modificação [registro tipo `HUPDAT`] de registros [29]. Relacionando e organizando os dados das operações sobre os dados da base de dados e os dados das transações presentes no relatório, obtêm-se os seguintes resultados intermediários (figuras 6.3 a 6.6):

Nome do campo	Tipo de dados	Descrição
cod_processo	Número	Código seqüencial de identificação do processo no banco de dados
cod_log	Número	Código seqüencial de identificação da operação no banco de dados
cod_usuario	Texto	Código de identificação do usuário
cod_tabela	Texto	Código de identificação da tabela
dat_processo	Data/Hora	Data de execução do processo no banco de dados
cod_operacao	Texto	Código da operação sobre o banco de dados
rowid	Texto	Localização da linha de dados no banco de dados

Figura 6.3: Esquema dos dados intermediários referentes ao arquivo de *log*

cod_processo	cod_log	cod_usuario	cod_tabela	dat_processo	cod_operacao	rowid
14	1	informix	10004d	30/04/02 15:39:20	HUPDAT	309
14	2	informix	10004d	30/04/02 15:44:36	HUPDAT	309
15	1	informix	100052	30/04/02 15:41:00	HINSERT	30a
15	2	informix	100056	30/04/02 15:41:00	HINSERT	20c
15	3	informix	100056	30/04/02 15:42:39	HUPDAT	20c
15	4	informix	100052	30/04/02 15:44:09	HDELETE	30a
15	5	informix	100056	30/04/02 15:44:09	HDELETE	20c

Figura 6.4: Conteúdo dos dados intermediários referentes ao arquivo de *log*

Nome do campo	Tipo de dados	Descrição
cod_processo	Número	Código seqüencial de identificação do processo
cod_log	Número	Código seqüencial de identificação do processo no banco de dados
cod_coluna	Número	Código seqüencial de identificação da coluna
pos_coluna	Número	Deslocamento dentro da coluna
val_antes	Texto	Valor antes da operação de UPDATE ou DELETE
val_depois	Texto	Valor depois da operação de UPDATE ou INSERT

Figura 6.5: Esquema dos dados intermediários dos detalhes das operações de banco de dados

cod_processo	cod_log	cod_coluna	pos_coluna	val_antes	val_depois
14	1	4	1	2	3
14	2	4	1	3	2
15	1	1	0		sidney
15	1	2	0		01
15	1	3	0		1
15	1	4	0		T
15	2	1	0		sidney
15	2	2	0		01
15	2	3	0		
15	2	4	0		SIDNEY
15	2	5	0		
15	2	6	0		
15	2	7	0		
15	2	8	0		
15	3	4	7		VICENTE MENDES
15	4	1	0	sidney	
15	4	2	0	01	
15	4	3	0	1	
15	4	4	0	T	
15	5	1	0	sidney	
15	5	2	0	01	
15	5	3	0		
15	5	4	0	SIDNEY VICENTE MENDES	
15	5	5	0		
15	5	6	0		
15	5	7	0		
15	5	8	0		

Figura 6.6: Conteúdo dos dados intermediários dos detalhes das operações de banco de dados

6.4 RELACIONAMENTO ENTRE DADOS INTERMEDIÁRIOS

Nesse momento estão disponíveis os resultados intermediários do processo de captura e tratamento dos dados obtidos dos registros dos subsistemas de controle interno do sistema operacional e sistema gerenciador de banco de dados. Para profissionais de informática eles já seriam suficientes para procurar evidências sobre fatos ocorridos sobre a informação, porém não seriam adequados para um auditor ou administrador. Mesmo para profissionais de informática ainda haveria um processo trabalhoso pela frente.

Nessa etapa será feito o processamento que resultará no relacionamento entre as informações de cada subsistema de controle interno, ou seja, os dados referentes aos programas serão associados aos dados referentes às operações submetidas por eles ao gerenciador de banco de dados.

Nesse experimento a associação será obtida pelo processamento seqüencial dos registros dos resultados intermediários das duas etapas. Ao primeiro processo registrado pelo subsistema de auditoria do sistema operacional serão associadas às operações referentes ao primeiro processo interno do gerenciador do banco de dados, para o mesmo usuário. Nesse experimento houve a preocupação de sincronizar o processo de captura dos dados. Num ambiente de produção é imprescindível o sincronismo dos relógios dos equipamentos cliente e servidor, e a associação poderá ser feita respeitando-se a correspondência de usuário e instante de execução do programa cliente e das operações no servidor de banco de dados.

Após a realização dessa associação entre os resultados intermediários das etapas anteriores, os dados poderão ser carregados na base de consulta da auditoria. Para esse experimento foi utilizado o Microsoft Access como ferramenta de análise dos dados de auditoria, mas qualquer gerenciador de banco de dados relacional poderá ser empregado. Essa base de dados será composta pelos objetos que compõem a estrutura de dicionário de dados, presentes no anexo 2, e mais as seguintes estruturas (figuras 6.7 a 6.12):

Estrutura `processo`

	Nome do campo	Tipo de dados	Descrição
🔑	cod_processo	Número	Código seqüencial de identificação do processo
	id_processo	Número	Identificação do processo para o sistema operacional
	dat_processo	Data/Hora	Data de execução do processo
	cod_usuario	Texto	Código de identificação do usuário
	comando	Texto	Comando que iniciou o processo

Figura 6.7: Esquema dos dados referentes aos processos

cod_processo	id_processo	dat_processo	cod_usuario	comando
1	11130	30/04/02 15:36:29	informix	fglgo /home/sidney/men0000.4gi
2	13316	30/04/02 15:39:29	informix	fglgo /home/sidney/men0010.4gi

Figura 6.8: Conteúdo dos dados referentes aos processos

Estrutura `log`

	Nome do campo	Tipo de dados	Descrição
🔑	cod_processo	Número	Código seqüencial de identificação do processo
🔑	cod_log	Número	Código seqüencial de identificação da operação no banco de dados
	cod_usuario	Texto	Código de identificação do usuário
	cod_tabela	Texto	Código de identificação da tabela
	dat_processo	Data/Hora	Data de execução do processo no banco de dados
	cod_operacao	Texto	Código da operação sobre o banco de dados
	rowid	Texto	Localização da linha de dados no banco de dados

Figura 6.9: Esquema dos dados referentes ao arquivo de *log*

cod_processo	cod_log	cod_usuario	cod_tabela	dat_processo	cod_operacao	rowid
1	1	informix	10004d	30/04/02 15:39:20	HUPDAT	309
1	2	informix	10004d	30/04/02 15:44:36	HUPDAT	309
2	1	informix	100052	30/04/02 15:41:00	HINSERT	30a
2	2	informix	100056	30/04/02 15:41:00	HINSERT	20c
2	3	informix	100056	30/04/02 15:42:39	HUPDAT	20c
2	4	informix	100052	30/04/02 15:44:09	HDELETE	30a
2	5	informix	100056	30/04/02 15:44:09	HDELETE	20c

Figura 6.10: Conteúdo dos dados referentes ao arquivo de *log*

Estrutura `operacao`

	Nome do campo	Tipo de dados	Descrição
🔑	cod_processo	Número	Código seqüencial de identificação do processo
🔑	cod_log	Número	Código seqüencial de identificação do processo no banco de dados
🔑	cod_coluna	Número	Código seqüencial de identificação da coluna
	pos_coluna	Número	Deslocamento dentro da coluna
	val_antes	Texto	Valor antes da operação de UPDATE ou DELETE
	val_depois	Texto	Valor depois da operação de UPDATE ou INSERT

Figura 6.11: Esquema dos dados dos detalhes das operações de banco de dados

cod_processo	cod_log	cod_coluna	pos_coluna	val_antes	val_depois
1	1	4	1	2	3
1	2	4	1	3	2
2	1	1	0		sidney
2	1	2	0		01
2	1	3	0		1
2	1	4	0		T
2	2	1	0		sidney
2	2	2	0		01
2	2	3	0		
2	2	4	0		SIDNEY
2	2	5	0		
2	2	6	0		
2	2	7	0		
2	2	8	0		
2	3	4	7		VICENTE MENDES
2	4	1	0	sidney	
2	4	2	0	01	
2	4	3	0	1	
2	4	4	0	T	
2	5	1	0	sidney	
2	5	2	0	01	
2	5	3	0		
2	5	4	0	SIDNEY VICENTE MENDES	
2	5	5	0		
2	5	6	0		
2	5	7	0		
2	5	8	0		

Figura 6.12: Conteúdo dos dados dos detalhes das operações de banco de dados

6.5 ANÁLISE DA BASE DE DADOS DE AUDITORIA

A partir desse ponto a base de dados para suportar as consultas dos auditores está pronta. Mas ainda há a necessidade de preparar a interface para que os auditores possam visualizar a informação de maneira simples. A base de dados está normalizada, e são necessários comandos de banco de dados para relacionar os dados distribuídos nos diversos objetos e transformar esses dados em informação esclarecedora para o auditor.

Foram criadas algumas visões primeiramente para relacionar os dados que compõem os dois principais objetos da base de dados: o processo e as operações submetidas ao gerenciador de banco de dados.

A primeira visão construída relaciona os dados da estrutura `processo` com as estruturas do dicionário de dados `usuario` e `programa`. A visão está baseada no seguinte comando em linguagem SQL:

```
SELECT      processo.*,
            usuario.nom_usuario,
            programa.des_programa
FROM        processo,
            usuario,
            programa
WHERE       processo.cod_usuario = usuario.cod_usuario
AND         (processo.dat_processo = usuario.validade_ini
            Or processo.dat_processo > usuario.validade_ini)
AND         (processo.dat_processo = usuario.validade_fim
            Or processo.dat_processo<usuario.validade_fim)
AND         processo.comando = programa.comando
AND         (processo.dat_processo = programa.validade_ini
            Or processo.dat_processo > programa.validade_ini)
AND         (processo.dat_processo = programa.validade_fim
            Or processo.dat_processo < programa.validade_fim);
```

Os dados referentes ao processo agora poderiam ser visualizados complementados pelos dados do dicionário de dados conforme a figura 6.13. Agora é possível visualizar a identificação do usuário e seu nome completo, e o nome do programa e a descrição de sua função.

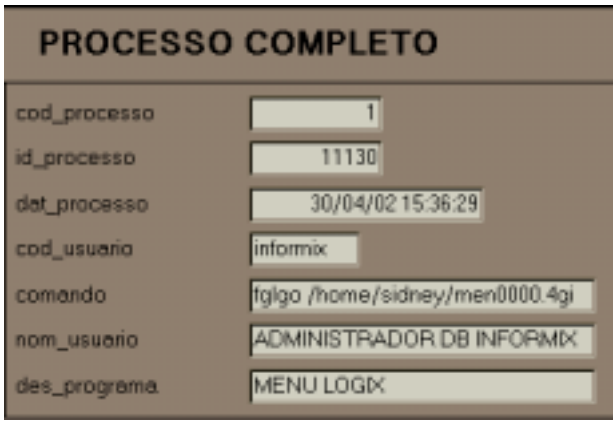


Figura 6.13: Visão dos detalhes dos registros do processo

A próxima visão relaciona todos os detalhes de cada operação submetida ao gerenciador de banco de dados pelo aplicativo cliente, tais como o nome da tabela, cada coluna que foi alvo da operação e suas propriedades. O comando em linguagem SQL que fundamenta essa visão é:

```
SELECT      log.*,
            tabela.nom_tabela,
            coluna.nom_coluna,
            coluna.tip_coluna,
            coluna.tam_coluna,
            operacao.pos_coluna,
            operacao.val_antes,
            operacao.val_depois
FROM        log,
            tabela,
            coluna,
            operacao,
            usuario
WHERE       log.cod_tabela = tabela.cod_tabela
AND         (log.dat_processo = tabela.validade_ini
            Or log.dat_processo > tabela.validade_ini)
AND         (log.dat_processo = tabela.validade_fim
            Or log.dat_processo < tabela.validade_fim)
AND         log.cod_processo = operação.cod_processo
AND         log.cod_log = operação.cod_log
AND         operação.cod_coluna = coluna.cod_coluna
AND         log.cod_tabela = coluna.cod_tabela
AND         (log.dat_processo = coluna.validade_ini
            Or log.dat_processo > coluna.validade_ini)
AND         (log.dat_processo = coluna.validade_fim
            Or log.dat_processo < coluna.validade_fim);
```

Os dados relacionados referentes às operações realizadas no banco de dados poderiam agora ser visualizados conforme a figura 6.14:

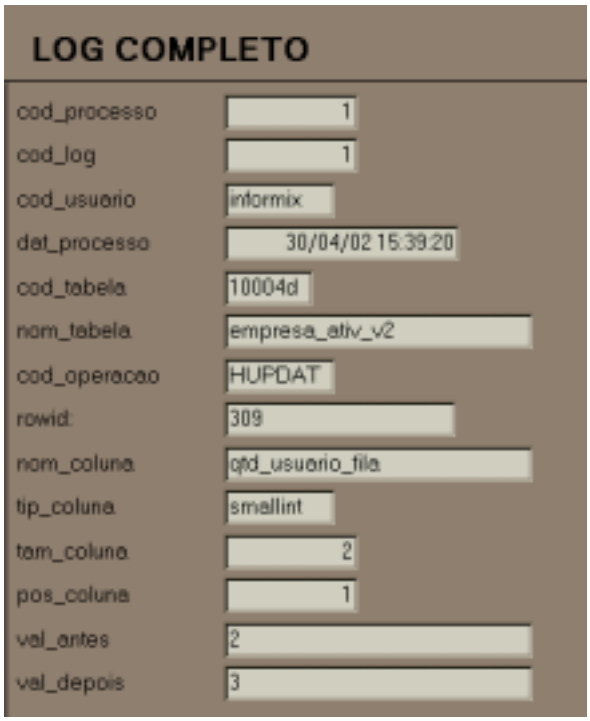


Figura 6.14: Visão dos detalhes dos registros de *log*

A próxima visão relaciona os dados obtidos a partir das duas visões anteriores e apresenta todos os dados registrados de cada operação executada sobre a base dados, bem como a origem do comando submetido ao gerenciador de banco de dados e o momento em que tal operação foi realizada. O comando SQL para essa visão é:

```
SELECT      processo_completo.*,
            log_completo.*
FROM        processo_completo,
            log_completo
WHERE       processo_completo.cod_processo =
                  log_completo.cod_processo;
```

A visualização dos dados possível agora seria a seguinte (figura 6.15):

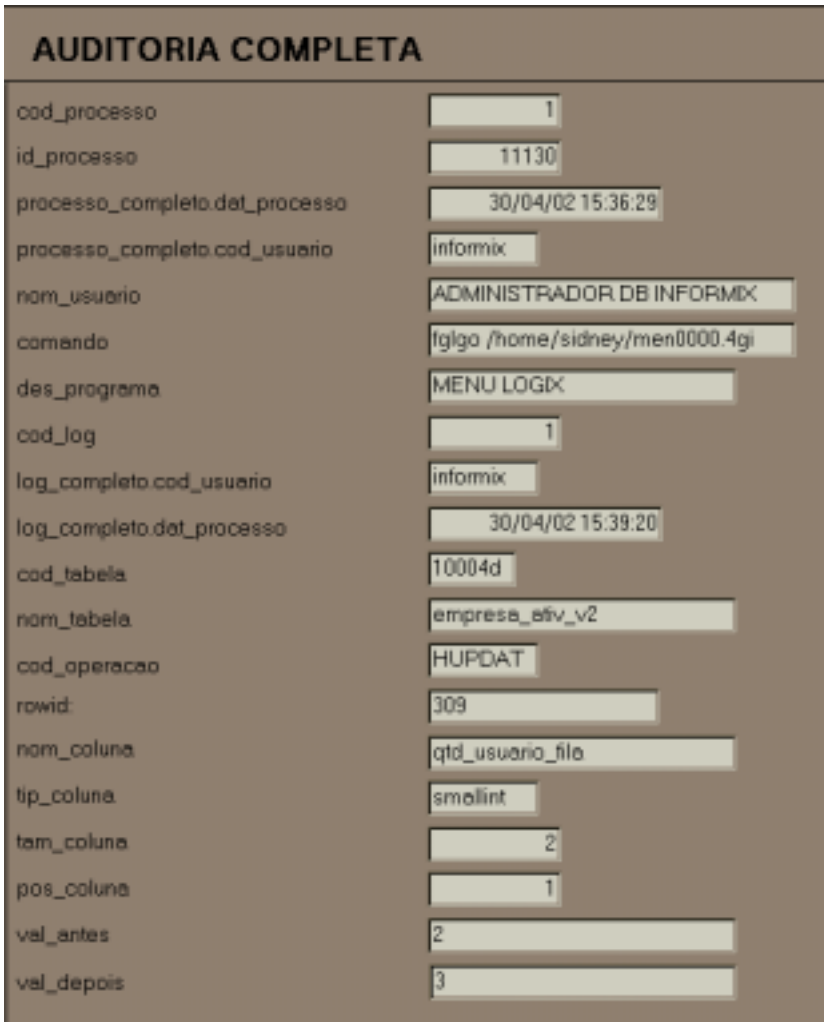


Figura 6.15: Visão completa dos dados relacionados

A construção dessa visão é importante, pois permite a visualização de todos os dados registrados e inter-relacionados, e a partir dela poderá ser construída a maioria

das consultas necessárias para a auditoria. Como exemplo, foi elaborada uma consulta mais apropriada para o auditor, omitindo uma série de dados que são mais do interesse do pessoal de informática. O resultado é apresentado na figura 6.16:

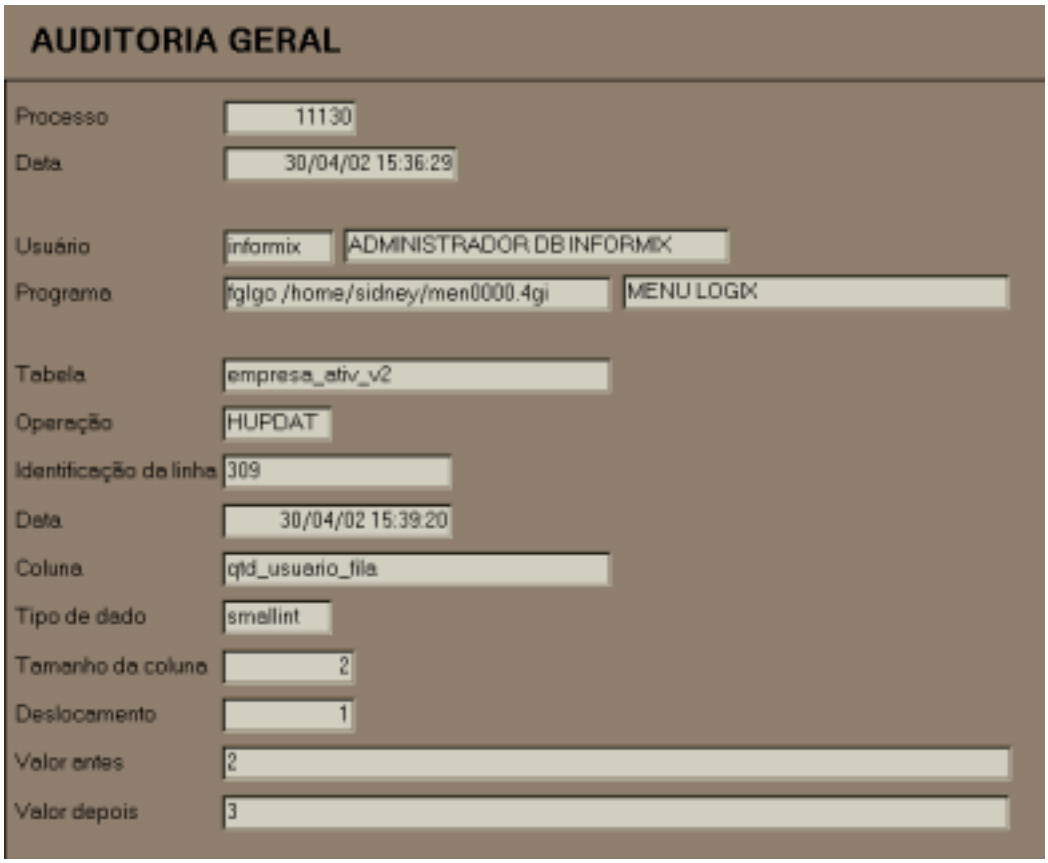


Figura 6.16: Visão geral com omissão de detalhes mais técnicos

Essa visão é ainda bastante genérica. Facilitaria muito apresentar os dados de auditoria relativos a uma regra de negócio. Como exemplo foi construída uma visão que apresenta a regra de negócio “Cadastro de Usuário”, que apresenta somente os dados relativos ao programa `men0010`, responsável pela manutenção do cadastro de usuário, e relativos à tabela `usuarios`. Abaixo está o comando SQL responsável por essa visão:

```
SELECT      auditoria_completa.processo_completo.dat_processo,
            auditoria_completa.processo_completo.cod_usuario,
            auditoria_completa.nom_usuario,
            auditoria_completa.cod_operacao,
            auditoria_completa.rowid,
            auditoria_completa.nom_coluna,
            auditoria_completa.tam_coluna,
            auditoria_completa.pos_coluna,
            auditoria_completa.val_antes,
            auditoria_completa.val_depois,
            auditoria_completa.log_completo.dat_processo
FROM        auditoria_completa
WHERE       auditoria_completa.comando Like "*men0010*”
AND         auditoria_completa.nom_tabela = “usuarios”;
```

A visão dos dados agora fica restrita aos dados relativos ao processamento para realização dessa regra de negócio. A figura 6.17 mostra o resulta dessa consulta:

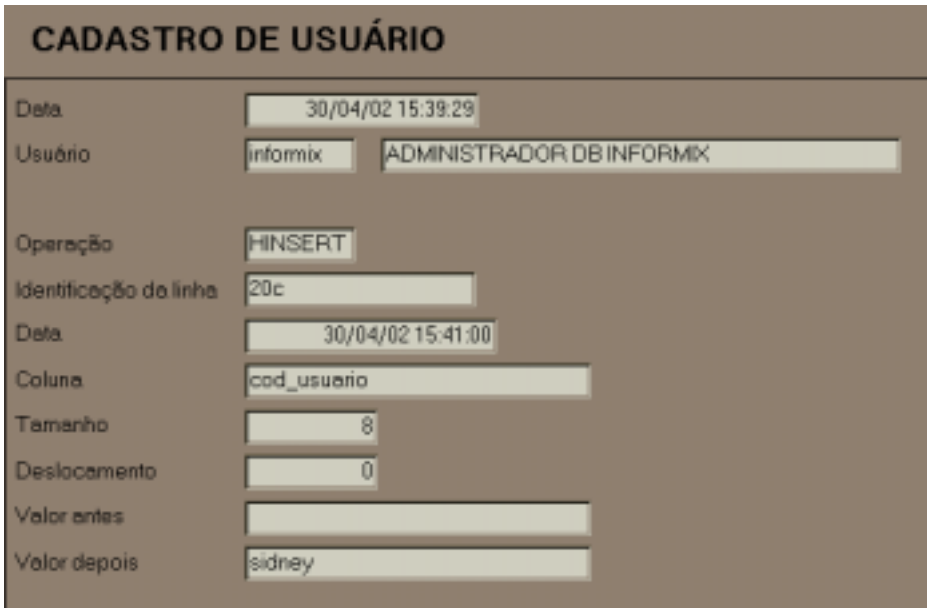


Figura 6.17: Visão relativa a uma regra de negócio

Observa-se agora que é possível determinar quando e quem realizou alguma manutenção no cadastro de usuários do sistema de informação, e com detalhes relativos a cada coluna da tabela `usuarios` envolvida nas operações no banco de dados. A tarefa do auditor fica reduzida a informar os argumentos de pesquisa para a ferramenta de consulta genérica.

6.6 RESUMO

Nesse capítulo foram aplicadas as orientações formalizadas na abordagem metodológica proposta em um ambiente tradicional de sistemas de informação, especificamente um sistema de informação desprovido de módulo de auditoria implementado. Todas as fases se mostraram exeqüíveis e com baixo fator de complexidade, apresentando um resultado totalmente adequado às expectativas da proposta. Apesar de ser reconhecidamente um recurso para profissionais de informática, os registros gerados pelos controles internos, se devidamente tratados, podem ser de fácil utilização e compreensão para auditores e administradores. O recurso de visões e as ferramentas genéricas de consulta permitem essa facilidade.

Fica assim comprovada a aplicabilidade da abordagem metodológica proposta, mediante a realização desse experimento, conforme as especificações da metodologia. Foi possível praticar uma auditoria futura, rastreando as atividades que foram realizadas, de forma independente do sistema de informação.

7 CONCLUSÕES

O objetivo desse trabalho de apresentar uma metodologia para o desenvolvimento de auditoria independente de sistemas de informação, focada na integridade da informação, baseada em controles internos do sistema operacional e do gerenciador de banco de dados, permitindo rastrear as atividades dos usuários de sistemas de informação sobre as informações, e provendo uma interface aos auditores, que oculte a complexidade de análise dos registros de controle, diante do experimento realizado, foi plenamente atingido.

Foi apresentada a abordagem metodológica proposta, com características modulares. Foram detalhados os subsistemas de fronteira (especificamente controle de trilha de auditoria) e do banco de dados (especificamente controle de existência (*log*)), definidos como adequados para a tarefa de coleta de evidências para tal abordagem. As orientações formalizadas na abordagem metodológica proposta foram aplicadas em um ambiente tradicional de sistema de informação desprovido de módulo de auditoria específico.

Porém existe a necessidade de um alto nível de conhecimento técnico sobre o sistema operacional e o sistema gerenciador de banco de dados, para a realização da coleta e preparação dos dados gerados pelos controles internos desses produtos, e que serão a base para a auditoria futura.

Além disso, não existe compatibilidade entre os diversos sistemas operacionais e sistemas gerenciadores de banco de dados. Portanto, os módulos de preparação dos dados coletados são dependentes do produto com que estão tratando, e não são portáveis entre plataformas distintas.

Também estão sujeitos às mudanças entre versões dos produtos, uma vez que os controles são internos e não existem garantias de continuidade, nem padrões formais. As evoluções desses produtos impactam fortemente na engenharia interna do produto, e pode ser que haja necessidade de nova análise sobre a forma que os eventos são registrados.

Porém uma vez desenvolvidos esses módulos, de acordo com a abordagem metodológica, eles podem ser proliferados por toda a base de mesma plataforma. Eles são genéricos, independendo dos sistemas de informação sendo utilizados. Ou seja, novas necessidades de aplicativos sobre essa mesma plataforma, já estarão sujeitas a uma auditoria futura, pois são suportadas pelo mesmo ambiente de produção. E mesmo os aplicativos existentes, se sofrerem mudanças evolutivas em seu projeto, estarão automaticamente sujeitos da mesma forma à auditoria, pois a geração dos dados independe do aplicativo.

Se houver o esforço em desenvolver os módulo de coleta e tratamento para diversas opções de sistemas operacionais e gerenciadores de banco de dados, será possível compor diversas combinações e até mesmo operacionalizar essa solução em ambientes heterogêneos, pois desse ponto em diante, os módulos de associação e apresentação independem da origem dos dados, proporcionando uma solução genérica.

Fica a sugestão para trabalhos futuros sobre essa abordagem orientados a ambientes distribuídos. Há também certas dificuldades na identificação de usuários, quando o aplicativo realiza a conexão ao banco de dados utilizando sempre o mesmo usuário e realizando a segurança internamente. Outra dificuldade ocorre quando somente um grande aplicativo contém todo o sistema de informação, e disponibiliza as funções internamente através de menus e realiza internamente chamadas a módulos compartilhados. A combinação dessas duas situações associaria todas operações submetidas ao gerenciador de banco de dados a somente um usuário e um aplicativo.

A abordagem proposta é caracterizada por oferecer informações não em tempo-real. Por um lado isso é interessante, pois o processamento dos dados e a preparação da base de consulta podem ser agendados para horários mais ociosos de processamento, de forma a não prejudicar o nível de serviço nos horários mais atuantes dos usuários. Ou talvez nem haja necessidade de processamento, sendo feito apenas o armazenamento dos dados coletados para uma possível necessidade futura de auditoria. Os recursos de filtragem propostos também permitem focar-se em determinadas regras de negócio e reduzir o esforço computacional para geração da base de dados de consulta. Por outro lado existe uma demora entre a ocorrência do fato e sua constatação.

Demonstrou-se ser possível preparar visões orientadas às regras de negócio, para facilitar o trabalho do auditor. Porém é perceptível que algumas regras de negócio são complexas e fica difícil representa-las em termos de programas e operações sobre objetos do banco de dados. Também as mudanças evolutivas do sistema de informação podem ocasionar mudanças internas que modifiquem essa relação. Isso pode ser corrigido alterando a visão que representa a regra de negócio. A grande vantagem dessa abordagem proposta é que as evidências para os fatos ocorridos vão estar sempre disponíveis, devido à independência do aplicativo. Todos as operações submetidas pelo aplicativo ao gerenciador de banco de dados serão registradas e estarão associadas ao aplicativo. Deve-se, no entanto, estar ciente que não haverá a relação direta entre os comandos SQL presentes na aplicação com as operações registradas na base de consulta, derivadas do arquivo de *log*. Uma coisa é a solicitação passada para o gerenciador de banco de dados através do comando SQL, outra é a seqüência de operações definidas pelo gerenciador de banco de dados para ser aplicada no banco de dados para atender essa solicitação.

Além de servir à equipe de auditoria, essa abordagem pode ser de utilidade para a equipe técnica, pois outras respostas podem ser obtidas da base de consulta, e que são de interesse dela, tais como volume de transações, indicadores temporais de sobrecargas do ambiente computacional relacionadas a determinadas aplicações, etc.

É importante garantir que os dados originais gerados pelo subsistema de auditoria do sistema operacional e os arquivos de *log* gerados pelo gerenciador de banco de dados sejam copiados, imediatamente após sua disponibilização, para alguma mídia não regravável e seja evitado o armazenamento local, para evitar alterações e destruição, caso ocorra alguma invasão. Também os servidores e o armazenamento das mídias devem estar em ambiente fisicamente protegido. Ainda é recomendável que mais de uma pessoa fique responsabilizada por essa solução, pois a restrição a uma única pessoa abre a possibilidade de ação não submetida a controle, adulteração ou destruição das evidências.

Já a segurança de acesso à base de dados de auditoria pode ser implementada pelo próprio gerenciador de banco de dados, restringindo o acesso somente aos

auditores e administradores, e também restringindo à realização de operações somente de consulta.

Uma situação que precisa ser considerada nessa abordagem é o fato da separação não somente física do banco de dados e dos dados representantes das evidências que ocorrem sobre o banco de dados, mas a separação lógica entre a situação do banco de dados e das evidências, ocasionada pela demora entre a coleta e processamento dos dados e sua carga na base de dados de consulta. Podem surgir situações em que acertos na base de auditoria precisem ser feitos, devido à necessidade de recuperação do banco de dados ocasionada por erros de processamento ou corrupção do banco de dados. Nesse caso, a base de auditoria não é uma história verdadeira do que aconteceu à base de dados, e decisões incorretas poderiam ser tomadas com base nos dados de auditoria errônea.

Também devem ser consideradas as situações em que uma transação não chega ao seu final, devido a alguma falha de processamento, e suas operações devem ser desfeitas, para que a base de dados retorne ao ponto de consistência do início da transação. Nesses casos, mesmo havendo registros do processo e das operações realizadas sobre a base de dados, eles contam uma história confusa, e devem ser desconsiderados em termos de auditoria focada na rastreabilidade de aplicação de regras de negócio.

A abordagem proposta apresenta limitações e não tem a pretensão de tornar obsoletas ou inviáveis as demais abordagens, mas antes de complementa-las. Evidentemente que a abordagem de auditoria embutida no aplicativo é mais coesa com as regras de negócio do aplicativo, de utilização mais simples, e mesmo o custo adicional para implementação do sistema de informação pode ser rateado entre os clientes. Há apenas a questão de confiança na competência e abrangência do projeto. Porém essa abordagem só garante a auditoria sobre as ações realizadas pelos aplicativos desse sistema de informação. Qualquer outra forma de interação com a base de dados desse sistema de informação ou com outras bases de outros sistemas de informação não terão dados registrados para auditoria.

Com a comprovação da aplicabilidade e potencial da metodologia, pensa-se na sua evolução. Algumas funcionalidades poderiam ser adicionadas, tais como o suporte a

múltiplas bases de dados, a definição de uma interface que permita a construção de visões sem a necessidade de conhecer as estruturas da base de dados de auditoria, e a modelagem de estruturas adicionais que representem a hierarquia dos sistemas de informação por área de negócio, as quais permitiriam uma navegação e apresentação superior e mais intuitiva.

8 BIBLIOGRAFIA

[1] 7ª Pesquisa Nacional sobre Segurança da Informação. Módulo Security Solutions S.A. http://www.modulo.com.br. 2001.

[2] CARISSIMI, Leonardo. Segurança da Informação Agrega Valor? Modulo e-security Magazine. Agosto 2001.

[3] DATE, C. J. An Introduction to Database Systems. Addison Wesley. 1995.

[4] MINSKY, Naftaly H. Independent *On-Line* Monitoring of Evolving Systems. International Conference on *Software* Engineering. Berlim. 1996.

[5] PENTEADO, Sônia. De Olho na Segurança. Informationweek Brasil Ano 1 nº 6. p. 26-34

[6] PEREIRA, Cristiane. Visão Geral de Gestão de Segurança. Módulo e-security Magazine. Agosto 2001.

[7] SALLACH, David L. A Deductive Database Audit Trail. Symposium on Applied Computing. Kansas City. 1992. p. 314.

[8] WEBER, Ron. Information Systems Control and Audit. Prentice-Hall. 1998.

[9] CD AIX Documentation. AIX Version 4.3 Books. System Management Concepts: Operating System and Devices

[10] FANTINATTI, João Marcos. Auditoria em Informática. São Paulo. McGraw-Hill. 1988.

[11] HANSEN, James V. e MESSIER, William F. A Relational Approach to Decision Support for EDP Auditing. Communications of the ACM. November 1984, p.1129-1133.

[12] KOCH, Harvey S. Computer Auditing and Control. Proceedings of the 1979 Annual Conference. Janeiro 1979, p. 191.

[13] COLTRO, Renata. Dever de Casa Corporativo. Módulo Security Solutions S.A. http://www.modulo.com.br. 2001.

[14] COLTRO, Renata. Questão de Sobrevivência. Módulo Security Solutions S.A. http://www.modulo.com.br. 2001.

[15] PERSIN, David. Treating Information as an Asset. Proceedings of the 1984 Annual Conference of the ACM on The Fifth Generation Challenge. January 1984, p. 244.

[16] PIMENTA, Cristiano. Tomando Decisão no Contexto da Segurança da Informação. Módulo e-security Magazine. Agosto 2001.

[17] SÊMOLA, Marcos. Sete Dicas para Proteger o seu Negócio. IDG Now. Novembro 2000.

[18] CASANAS, Alex D. G. e MACHADO, Cesar de S. O Impacto da Implementação da Norma NBR ISO/IEC 17799 — Código de Prática para a Gestão da Segurança da Informação — nas Empresa. UFSC. 2000.

[19] NBR ISO/IEC 17799 — Tecnologia da Informação — Código de Prática para a Gestão da Segurança da Informação. ABNT. 2001.

[20] ORMAN, Levent V. Database Auditing. Proceedings of the Eighteenth Conference on Information Systems. 1997, p. 297-314.

[21] HEUSER, Carlos A. Projeto de Banco de Dados. Ed. Sagra Luzzatto. 2001.

[22] GIL, Antônio de L. Auditoria de Computadores. São Paulo. Ed. Atlas. 1993.

[23] SILBERSCHATZ, Abraham; KORTH, Henry F. e SUDARSHAN, S. Sistema de Banco de Dados. São Paulo. Ed. Makron Books. 1999.

[24] ALAGIÉ, Suad. Relational Database Technology. Nova York. Ed. Springer-Verlag. 1986.

[25] VINCENT, James G. Internal Audit: A User of Computer-Based Information Systems. Proceedings of the 17th annual computer personnel research conference Julho 1980. p. 41.

[26] Oracle Application System Administrator´s Guide

[27] Understanding Data Auditing in Oracle Application Tables. Agosto 2001.

[28] Overview of Oracle Applications Audit Trails. Agosto 2001.

[29] Administrator´s Guide for Informix Dynamic Server. INFORMIX Press. Menlo Park. Fevereiro 1998.

[30] Trusted Facility Manual for Informix Dynamic Server. INFORMIX Press. Menlo Park. Fevereiro 1998.

[31] DAVIS, Gordon B. Auditing & EDP. American Institute of Certified Public Accountants. New York. 1974.

[32] PEREIRA, Rafael. Como os Registros de *Log* Podem Ajudar em Processos de Investigação. Módulo e-security Magazine. Setembro 2001.

[33] Elements of Security: AIX 4.1. IBM International Technical Support Organization. New York. Outubro 1984.

[34] CHIN, Francis e OZSOYOGLU, Gultekin. Auditing for Secure Statistical Databases. Proceedings of the ACM '81 Conference. Janeiro 1981. p. 53.

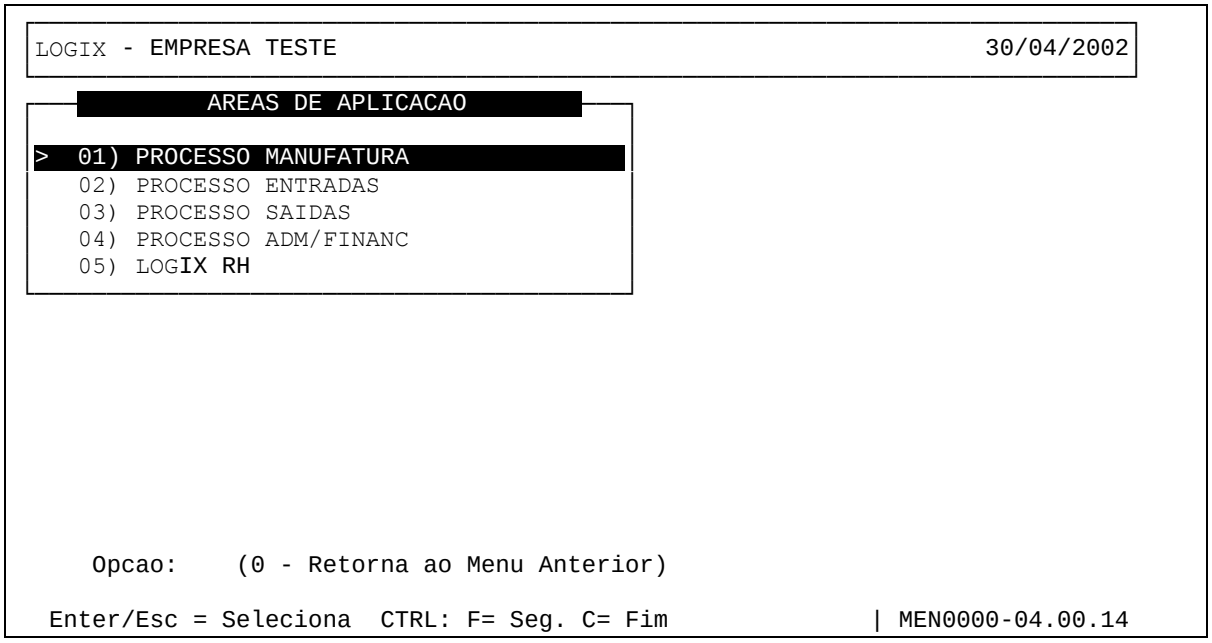
[35] PERRY, Raymond S. A System Manager´s View of Computing Auditing. Proceedings of the 1979 Annual Conference. Janeiro 1979. p. 189.

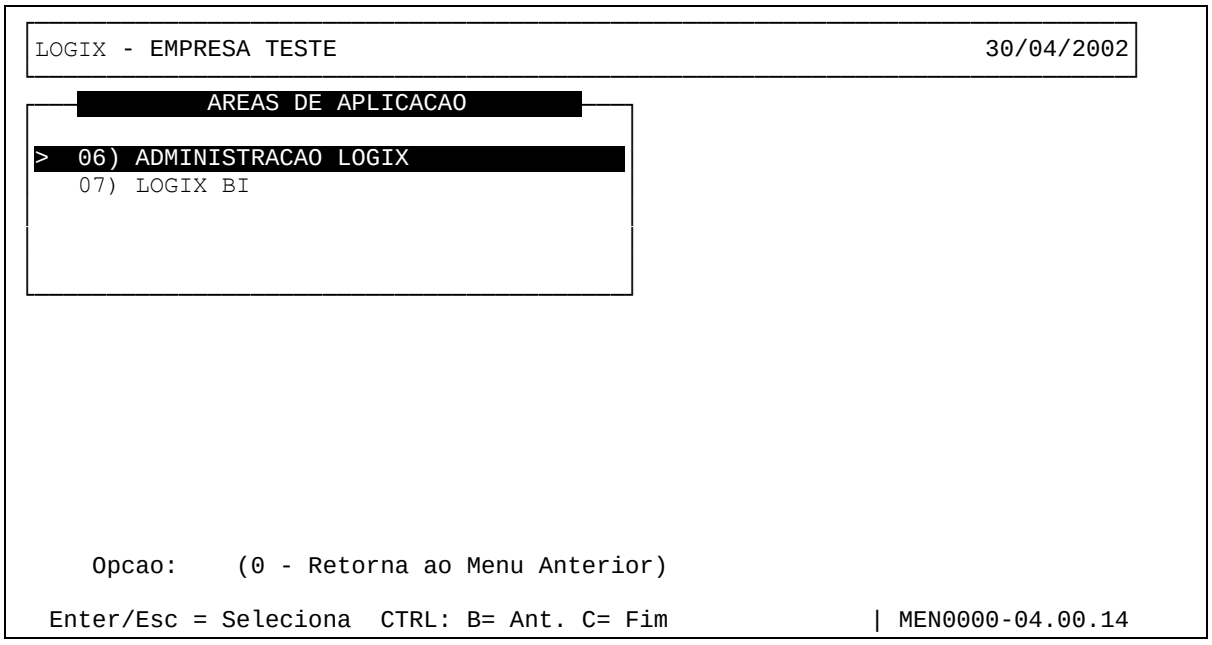
[36] KUMAR, Kuldeep. Post Implementation Evaluation of Computer-Based Information Systems: Current Practices. Communications of the ACM. Volume 35 n^{o} 2. Fevereiro 1990. p. 203.

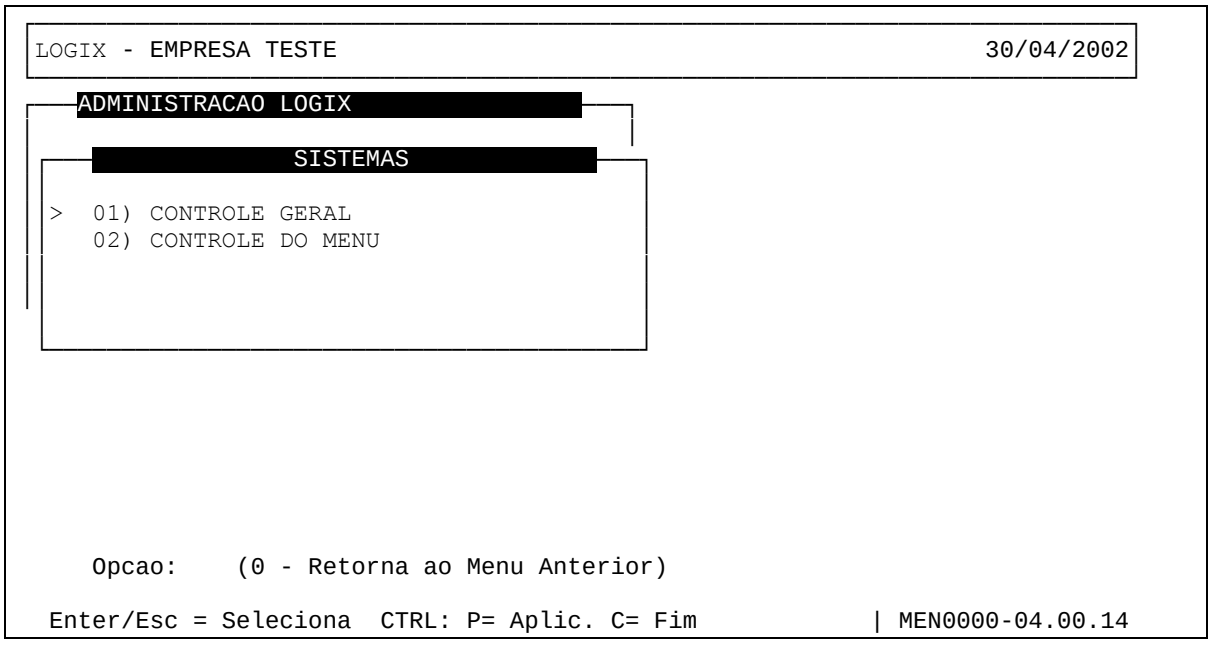
[37] CROSSMAN, Trevor D. Some Experiences in the Use of Inspection Teams. Proceedings of the Fifteenth Annual SIGCPR conference. Agosto 1977. p.143.

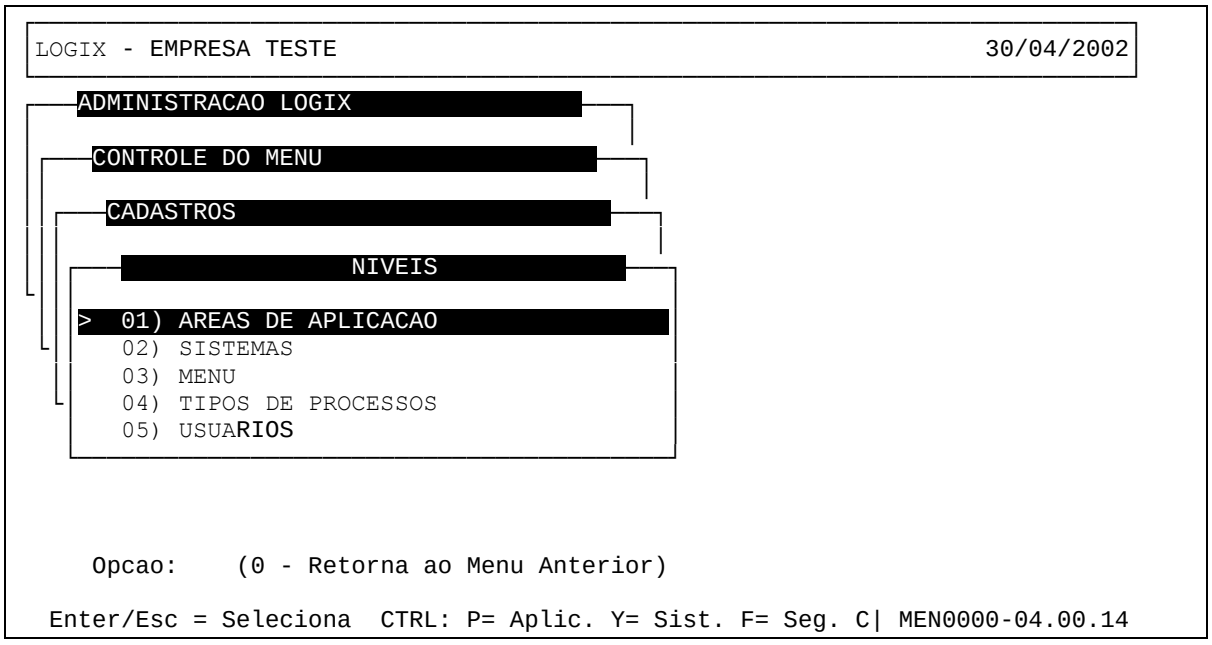
ANEXO 1 – SEQÜÊNCIA DE OPERAÇÕES NO SISTEMA DE INFORMAÇÃO

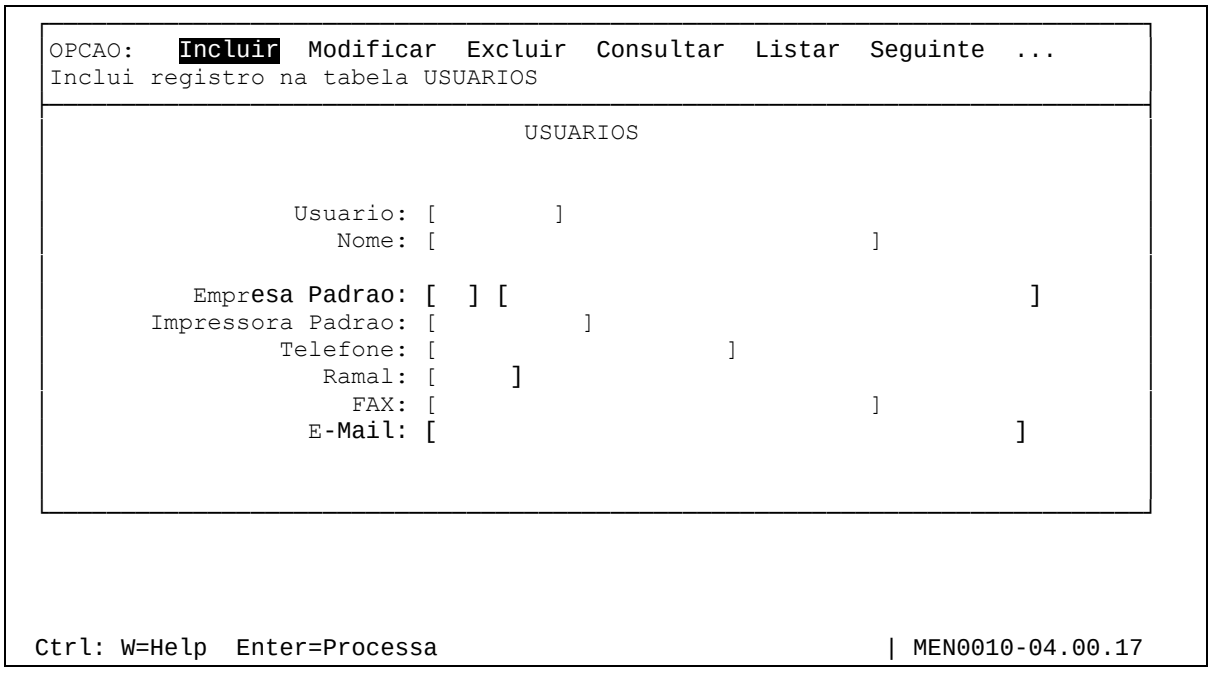
As figuras seguintes representam a interface do sistema Logix e a seqüência de operações realizadas sobre ele para esse experimento. A seqüência representa a chamada do menu principal do sistema de informação e do aplicativo de manutenção do cadastro de usuários do sistema, onde será feito o cadastro de um usuário, depois a modificação de um atributo desse usuário, e por fim a exclusão desse mesmo usuário:

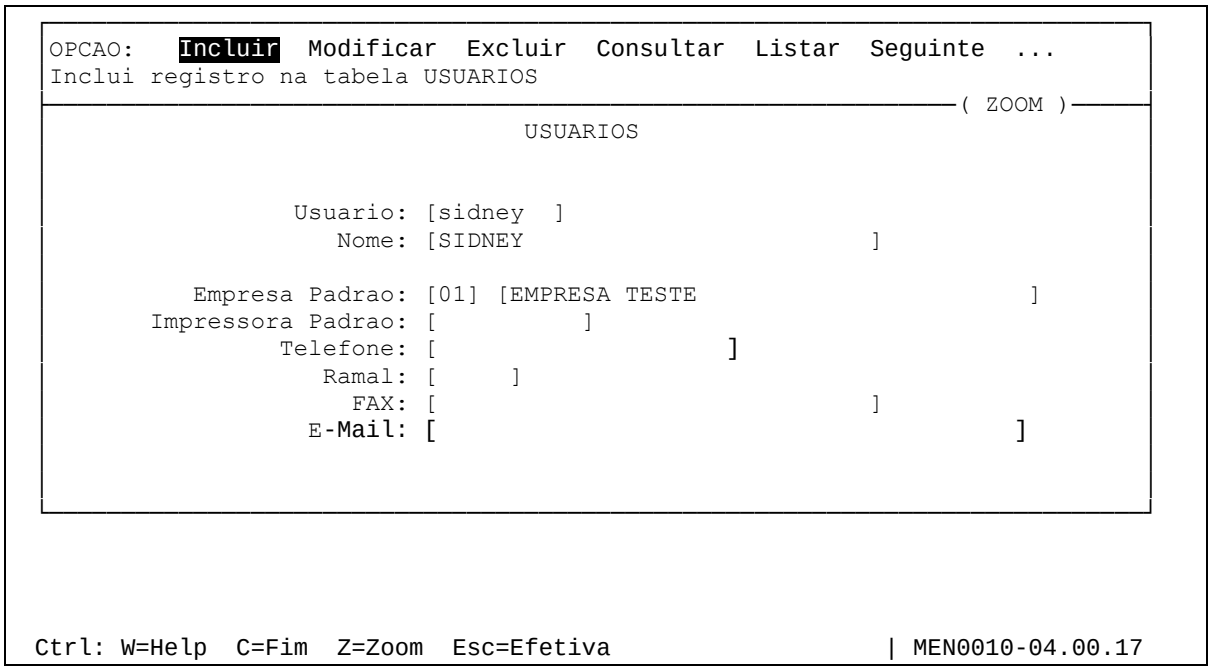


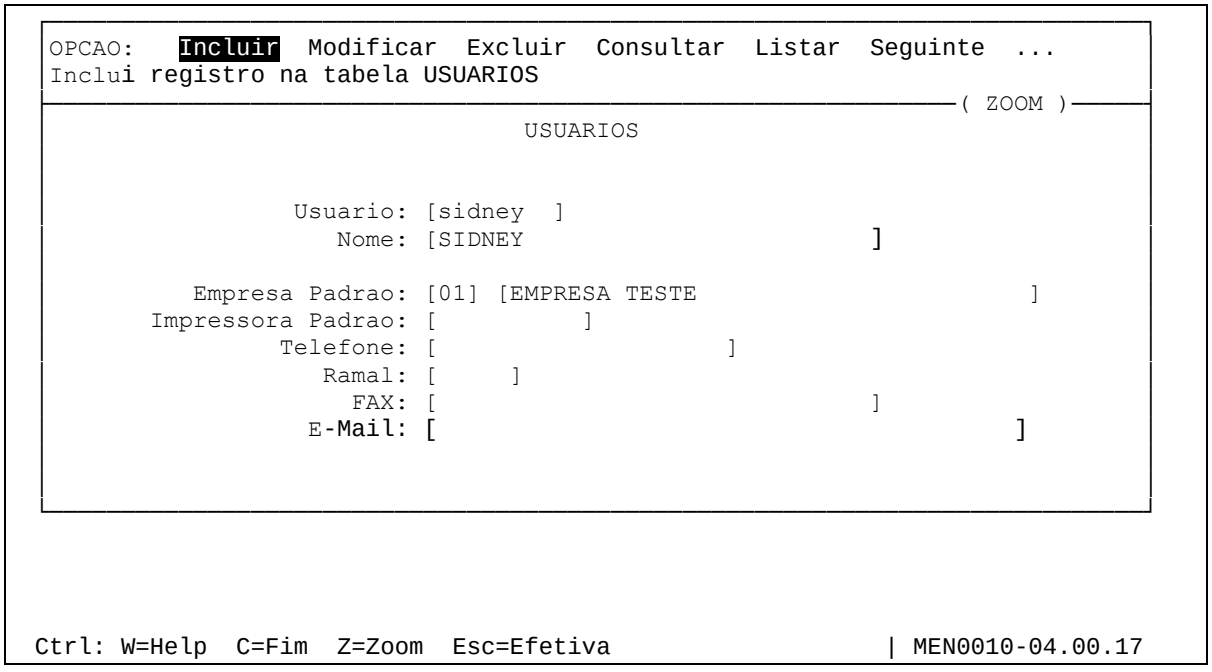

LOGIX - EMPRESA TESTE
30/04/2002
AREAS DE APLICACAO
> 06) ADMINISTRACAO LOGIX
07) LOGIX BI
Opcao: (0 - Retorna ao Menu Anterior)
Enter/Esc = Seleciona CTRL: B= Ant. C= Fim
| MEN0000-04.00.14

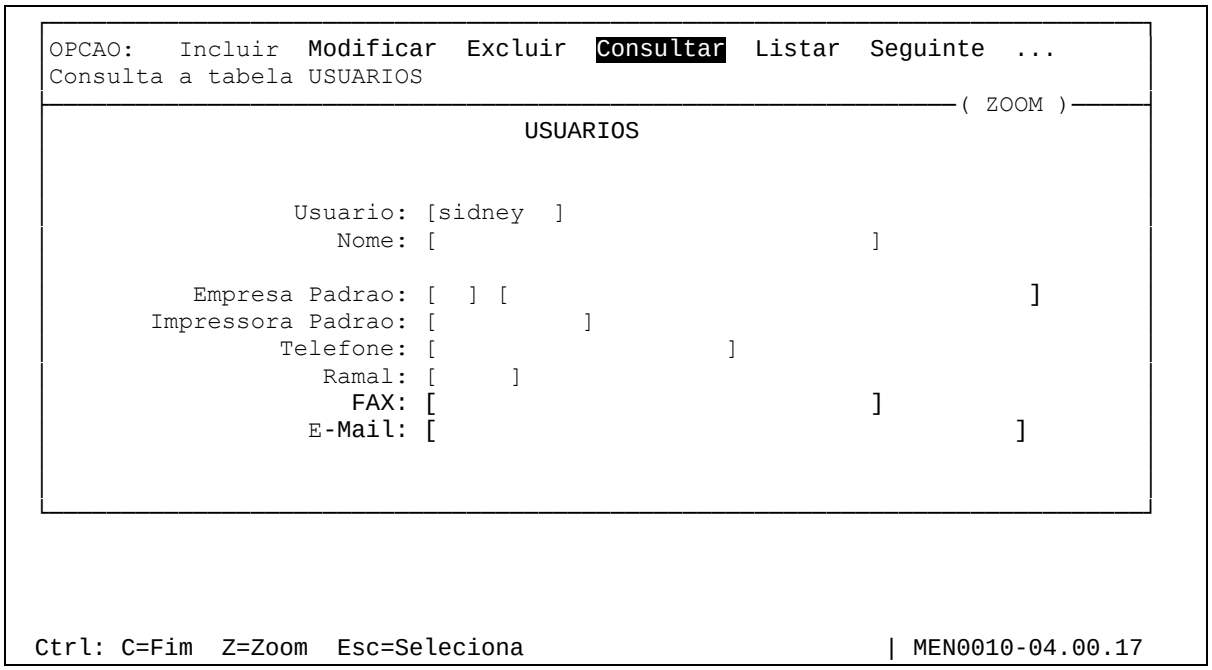

LOGIX - EMPRESA TESTE
30/04/2002
ADMINISTRACAO LOGIX
SISTEMAS
> 01) CONTROLE GERAL
02) CONTROLE DO MENU
Opcao: (0 - Retorna ao Menu Anterior)
Enter/Esc = Seleciona CTRL: P= Aplic. C= Fim
| MEN0000-04.00.14

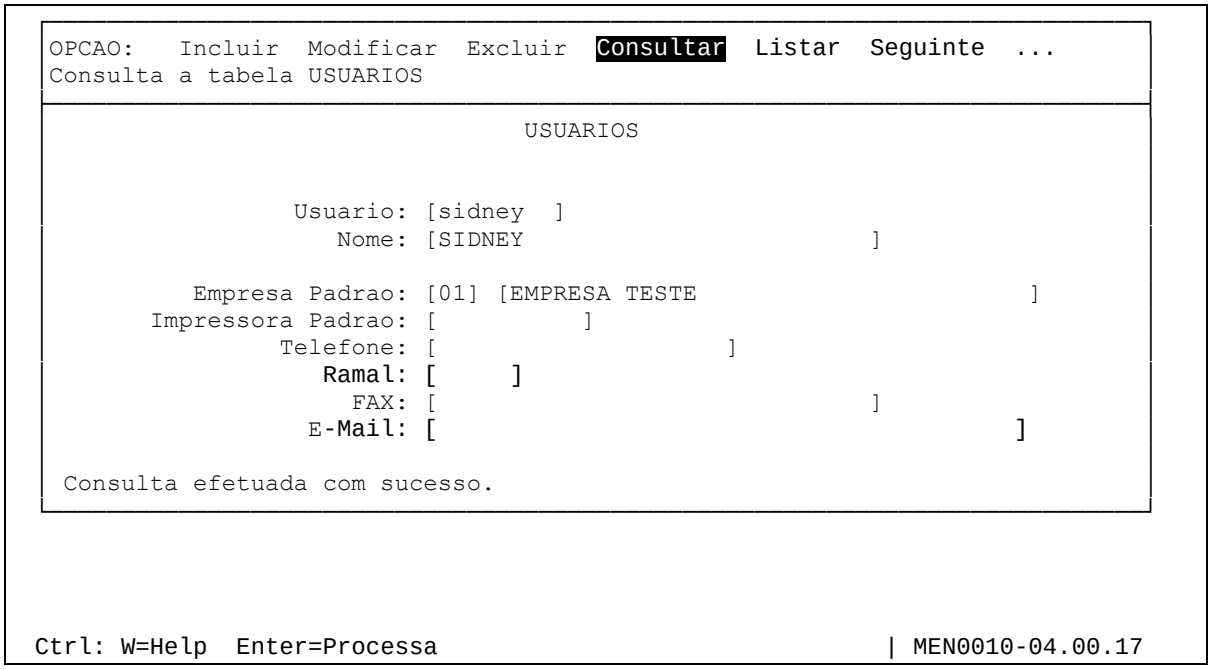

LOGIX - EMPRESA TESTE
30/04/2002
ADMINISTRACAO LOGIX
CONTROLE DO MENU
CADASTROS
NIVEIS
> 01) AREAS DE APLICACAO
02) SISTEMAS
03) MENU
04) TIPOS DE PROCESSOS
05) USUARIOS
Opcao: (0 - Retorna ao Menu Anterior)
Enter/Esc = Seleciona CTRL: P= Aplic. Y= Sist. F= Seg. C| MEN0000-04.00.14

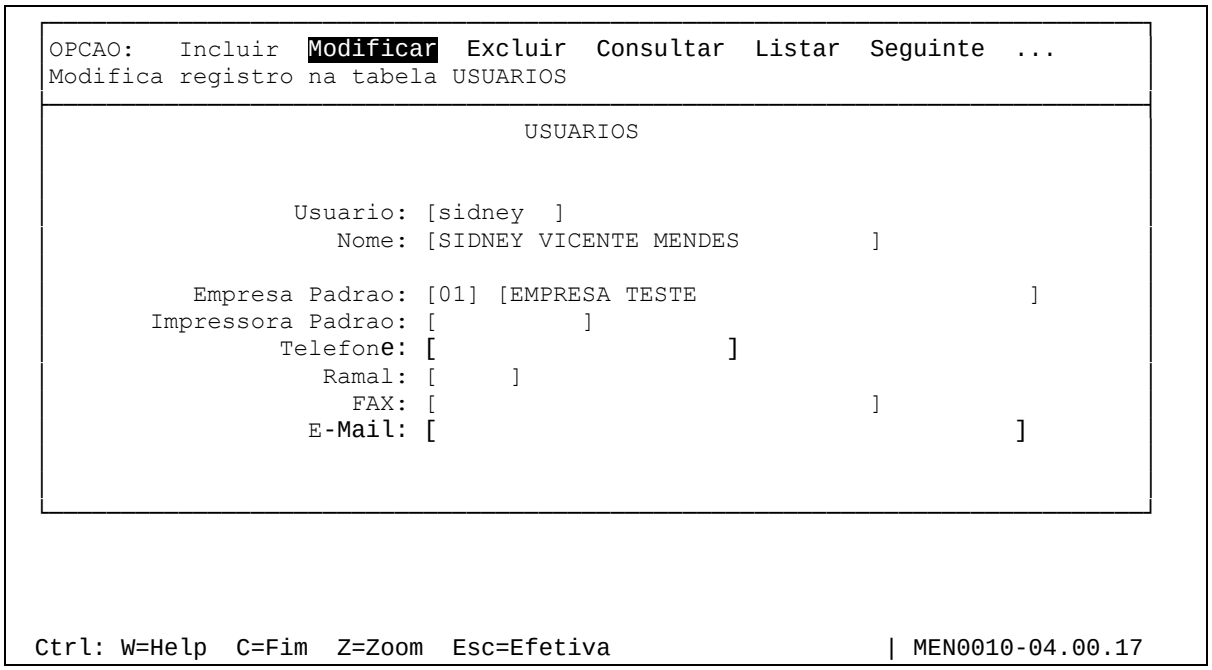

OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Inclui registro na tabela USUARIOS
USUARIOS
Usuario: [ ]
Nome: [ ]
Empresa Padrao: [ ] [ ]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Ctrl: W=Help Enter=Processa
| MEN0010-04.00.17

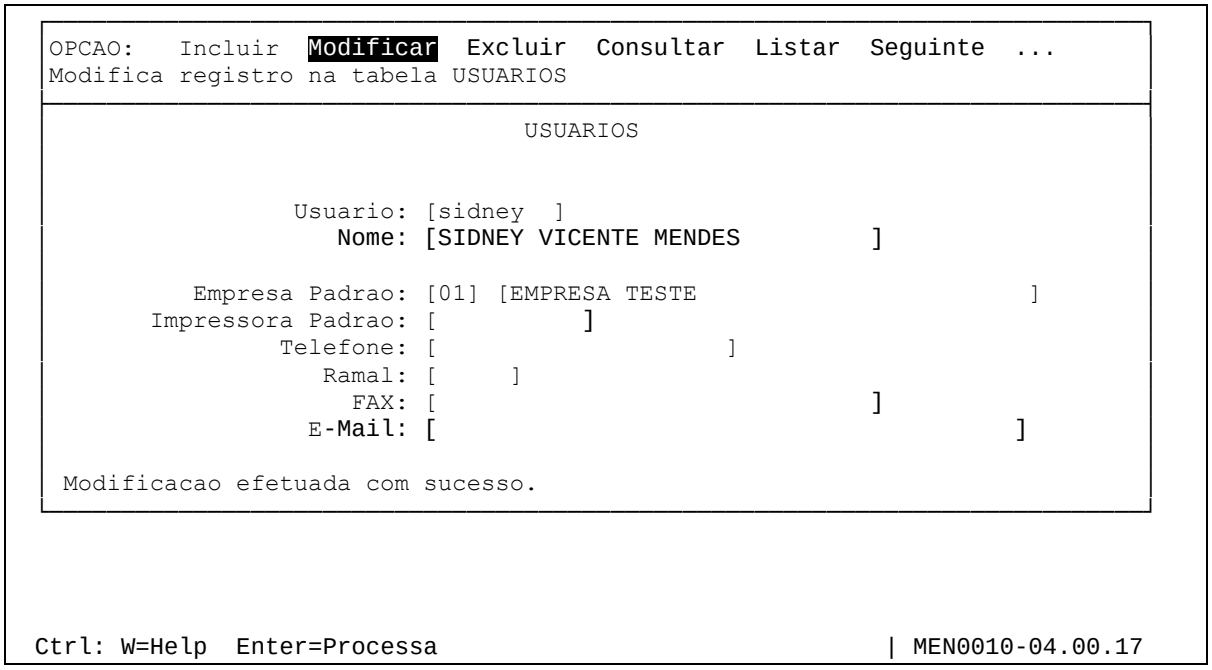

OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Inclui registro na tabela USUARIOS
(ZOOM)
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Ctrl: W=Help C=Fim Z=Zoom Esc=Efetiva | MEN0010-04.00.17

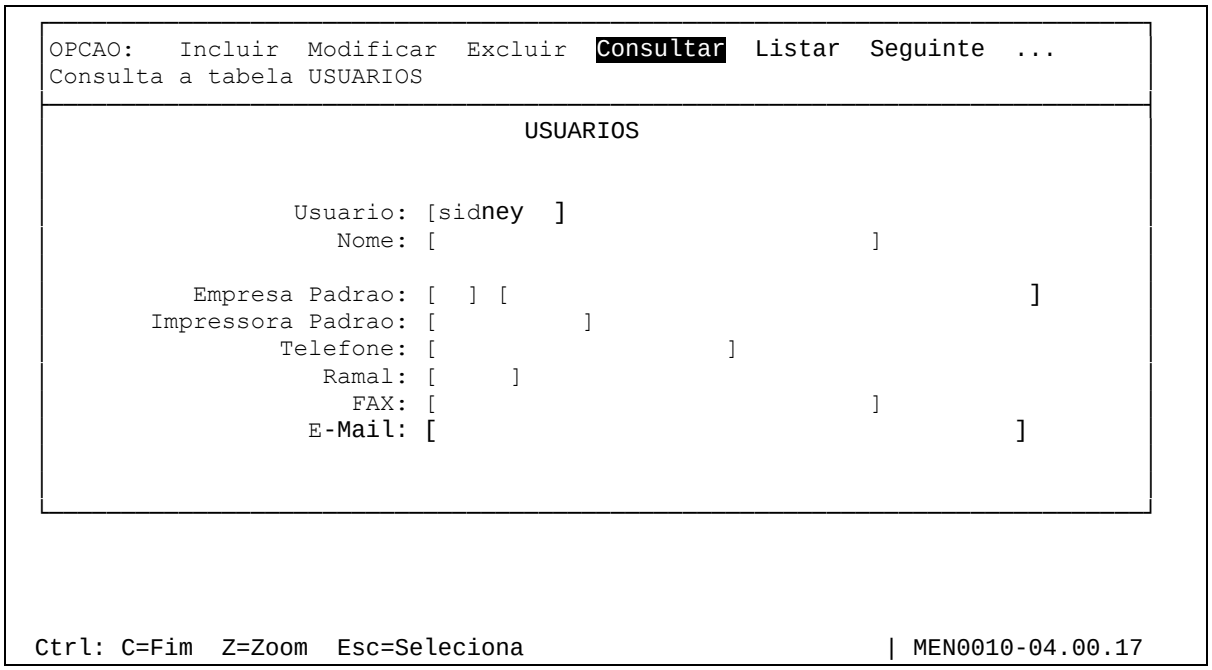

OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Incluir registro na tabela USUARIOS
(ZOOM)
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Ctrl: W=Help C=Fim Z=Zoom Esc=Efetiva | MEN0010-04.00.17

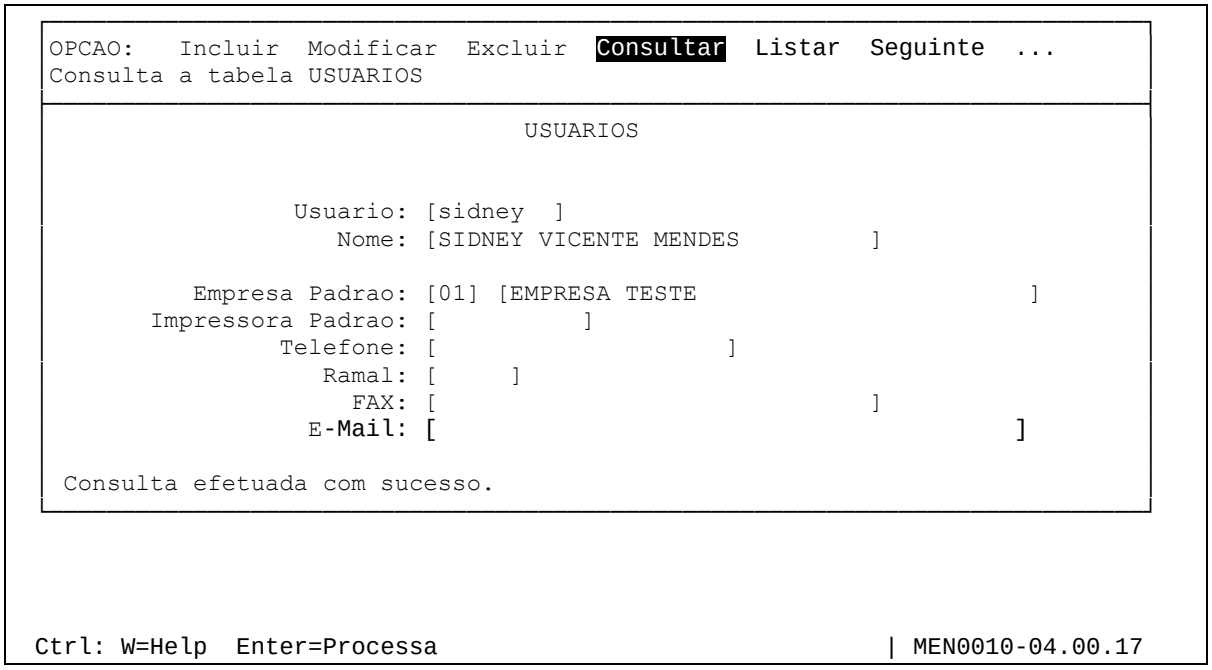

OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Consulta a tabela USUARIOS
(ZOOM)
USUARIOS
Usuario: [sidney]
Nome: [ ]
Empresa Padrao: [ ] [ ]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Ctrl: C=Fim Z=Zoom Esc=Seleciona | MEN0010-04.00.17

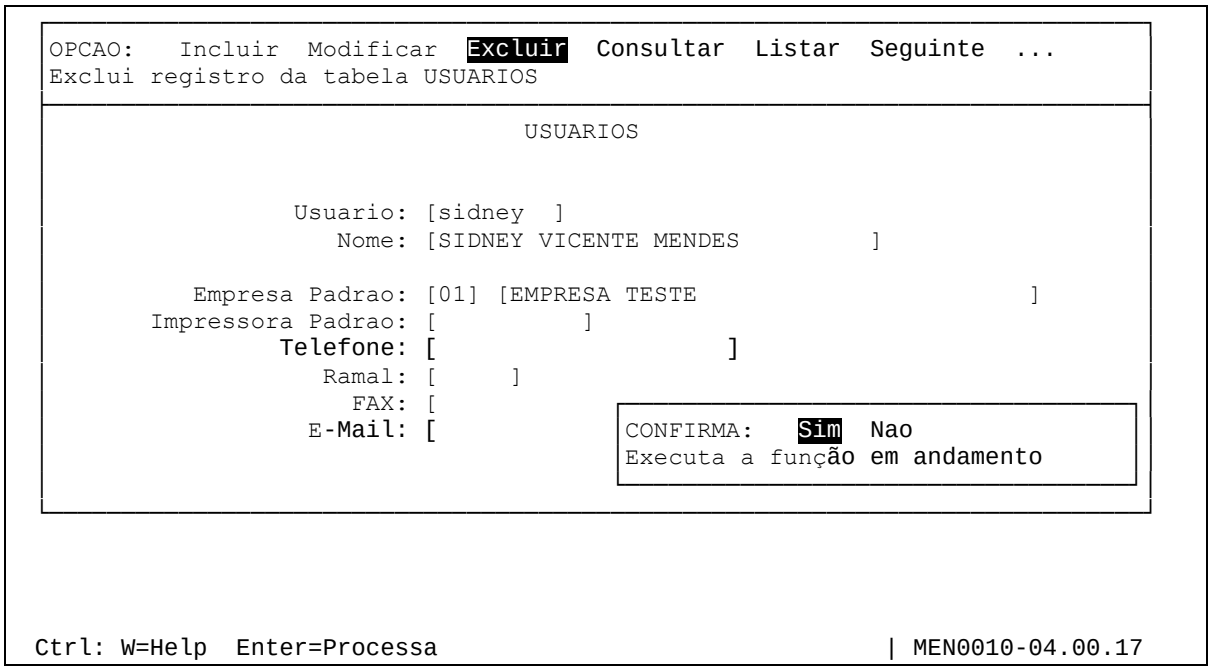

OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Consulta a tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Consulta efetuada com sucesso.
Ctrl: W=Help Enter=Processa | MEN0010-04.00.17

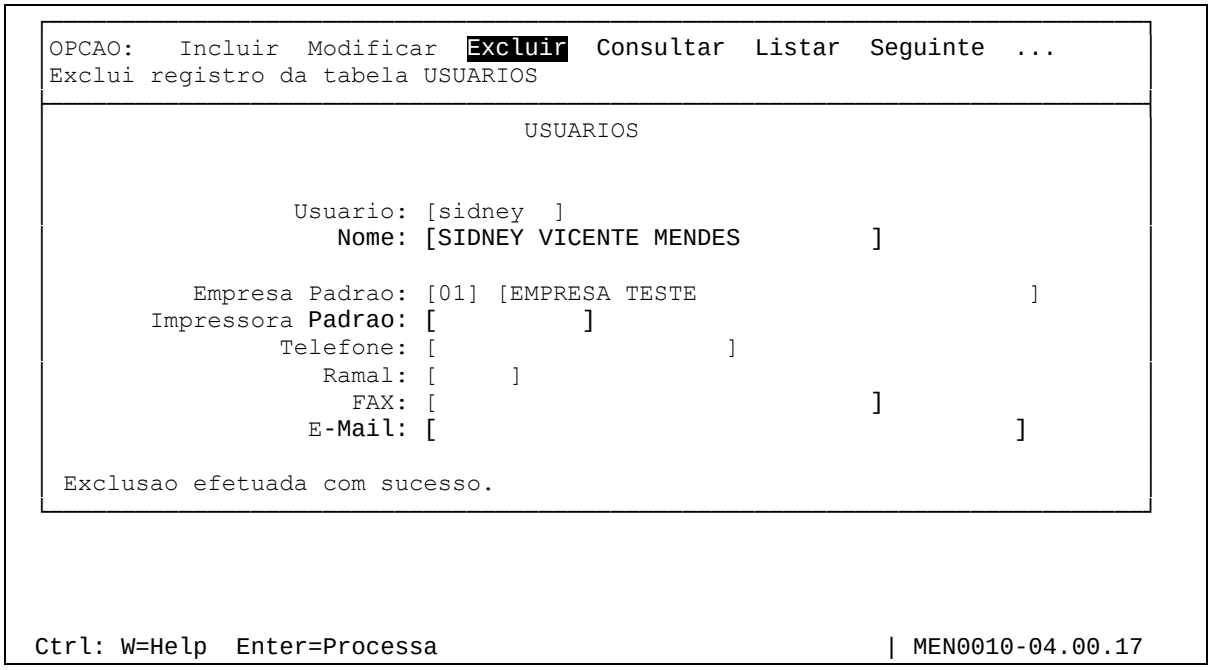

OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Modifica registro na tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY VICENTE MENDES]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Ctrl: W=Help C=Fim Z=Zoom Esc=Efetiva | MEN0010-04.00.17


OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Modifica registro na tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY VICENTE MENDES]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Modificacao efetuada com sucesso.
Ctrl: W=Help Enter=Processa | MEN0010-04.00.17


OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Consulta a tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [ ]
Empresa Padrao: [ ] [ ]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Ctrl: C=Fim Z=Zoom Esc=Seleciona | MEN0010-04.00.17


OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Consulta a tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY VICENTE MENDES]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Consulta efetuada com sucesso.
Ctrl: W=Help Enter=Processa | MEN0010-04.00.17


OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Exclui registro da tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY VICENTE MENDES]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [
E-Mail: [
CONFIRMA: Sim Nao
Executa a função em andamento
Ctrl: W=Help Enter=Processa
MEN0010-04.00.17


OPCAO: Incluir Modificar Excluir Consultar Listar Seguinte ...
Exclui registro da tabela USUARIOS
USUARIOS
Usuario: [sidney]
Nome: [SIDNEY VICENTE MENDES]
Empresa Padrao: [01] [EMPRESA TESTE]
Impressora Padrao: [ ]
Telefone: [ ]
Ramal: [ ]
FAX: [ ]
E-Mail: [ ]
Exclusao efetuada com sucesso.
Ctrl: W=Help Enter=Processa
MEN0010-04.00.17

ANEXO 2 – DICIONÁRIO DE DADOS TEMPORAL

A seguir são apresentados os esquemas e conteúdos das tabelas que compõem o dicionário de dados temporal (figuras A2.1 a A2.8) necessário para a realização do experimento de demonstração da aplicabilidade da metodologia proposta:

Estrutura `usuario`

	Nome do campo	Tipo de dados	Descrição
🔑	cod_usuario	Texto	Código de identificação do usuário
🔑	validade_ini	Data/Hora	Início da validade
🔑	validade_fim	Data/Hora	Fim da validade
	nom_usuario	Texto	Nome do usuário

Figura A2.1: Esquema dos dados referentes aos usuários

cod_usuario	validade_ini	validade_fim	nom_usuario
informix	30/04/02 15:36:29	31/12/2999 23:59:59	ADMINISTRADOR DB INFORMIX

Figura A2.2: Conteúdo dos dados referentes aos usuários

Estrutura `programa`

	Nome do campo	Tipo de dados	Descrição
🔑	comando	Texto	Comando que iniciou o processo
🔑	validade_ini	Data/Hora	Início da validade
🔑	validade_fim	Data/Hora	Fim da validade
	des_programa	Texto	Descrição do programa

Figura A2.3: Esquema dos dados referentes aos programas

comando	validade_ini	validade_fim	des_programa
fglgo /home/sidney/men0000.4gi	30/04/02 15:36:29	31/12/2999 23:59:59	MENU LOGIX
fglgo /home/sidney/men0010.4gi	30/04/02 15:39:29	31/12/2999 23:59:59	CADASTRO DE USUARIO LOGIX

Figura A2.4: Contúdo dos dados referentes aos programas

Estrutura `tabela`

	Nome do campo	Tipo de dados	Descrição
🔑	cod_tabela	Texto	Código de identificação da tabela
🔑	validade_ini	Data/Hora	Início da validade
🔑	validade_fim	Data/Hora	Fim da validade
	nom_tabela	Texto	Nome da tabela

Figura A2.5: Esquema dos dados referentes às tabelas

cod_tabela	validade_ini	validade_fim	nom_tabela
10004D	30/04/02 15:39:20	31/12/2999 23:59:59	empresa_ativ_v2
100052	30/04/02 15:39:20	31/12/2999 23:59:59	usuario_empresa
100056	30/04/02 15:39:20	31/12/2999 23:59:59	usuarios

Figura A2.6: Conteúdo dos dados referentes às tabelas

Estrutura `coluna`

	Nome do campo	Tipo de dados	Descrição
🔑	cod_tabela	Texto	Código de identificação da tabela
🔑	validade_ini	Data/Hora	Início da validade
🔑	validade_fim	Data/Hora	Fim da validade
🔑	cod_coluna	Número	Código seqüencial de identificação da coluna
	nom_coluna	Texto	Nome da coluna
	tip_coluna	Texto	Tipo de dado da coluna
	tam_coluna	Número	tamanho da coluna
	pos_registro	Número	Deslocamento dentro do registro

Figura A2.7: Esquema dos dados referentes às colunas das tabelas

cod_tabela	validade_ini	validade_fim	cod_coluna	nom_coluna	tip_coluna	tam_coluna	pos_registro
10004D	30/04/02 15:39:20	31/12/2999 23:59:59	1	cod_empresa	char	2	0
10004D	30/04/02 15:39:20	31/12/2999 23:59:59	2	nom_usuario	char	8	2
10004D	30/04/02 15:39:20	31/12/2999 23:59:59	3	ies_ambiente	char	1	10
10004D	30/04/02 15:39:20	31/12/2999 23:59:59	4	qtd_usuario_fila	smallint	2	11
100052	30/04/02 15:39:20	31/12/2999 23:59:59	1	cod_usuario	char	8	0
100052	30/04/02 15:39:20	31/12/2999 23:59:59	2	cod_empresa	char	2	8
100052	30/04/02 15:39:20	31/12/2999 23:59:59	3	qtd_max_usuarios	smallint	2	10
100052	30/04/02 15:39:20	31/12/2999 23:59:59	4	ies_acesso_niveis	char	1	12
100056	30/04/02 15:39:20	31/12/2999 23:59:59	1	cod_usuario	char	8	0
100056	30/04/02 15:39:20	31/12/2999 23:59:59	2	cod_empresa_padrao	char	2	8
100056	30/04/02 15:39:20	31/12/2999 23:59:59	3	cod_impress_padrao	char	10	10
100056	30/04/02 15:39:20	31/12/2999 23:59:59	4	nom_funcionario	char	30	20
100056	30/04/02 15:39:20	31/12/2999 23:59:59	5	num_telefone	char	20	50
100056	30/04/02 15:39:20	31/12/2999 23:59:59	6	num_ramal	char	5	70
100056	30/04/02 15:39:20	31/12/2999 23:59:59	7	num_fax	char	30	75
100056	30/04/02 15:39:20	31/12/2999 23:59:59	8	e_mail	char	30	105

Figura A2.8: Conteúdo dos dados referentes às colunas das tabelas

ANEXO 3 – AUDITORIA DE SISTEMA OPERACIONAL

O subsistema de auditoria no AIX foi configurado para monitorar os eventos de interesse para a abordagem metodológica proposta, e alguns arquivos foram modificados para esse experimento.

O arquivo `/etc/security/audit/objects`, que identifica os objetos que serão monitorados de acordo com a operação sendo executada sobre eles, teve incluídas as seguintes entradas:

```
/usr/informix/bin/fglgo:
      r = "FILE_Accessx"
      x = "PROC_Execute"

/home/sidney/men0000.4gi:
      r = "FILE_Accessx"

/home/sidney/men0010.4gi:
      r = "FILE_Accessx"
```

O arquivo `/usr/informix/bin/fglgo` é o programa (*run time*) responsável pela execução dos programas que compõem o sistema Logix, e será monitorado para operações de leitura e execução. Os arquivos `/home/sidney/men0000.4gi` e `/home/sidney/men0010.4gi` são os programas que serão alvo desse experimento, e serão monitorados para operações de leitura.

O arquivo `/etc/security/audit/oconfig` especifica como será realizada a auditoria e onde serão gravados os arquivos com os dados capturados. Somente o texto em negrito foi modificado:

```
start:
      binmode = on
      streammode = off

bin:
      trail = /home/sidney/audit/trail
      bin1 = /home/sidney/audit/bin1
      bin2 = /home/sidney/audit/bin2
      binsize = 10240
      cmds = /etc/security/audit/bincmds

stream:
      cmds = /etc/security/audit/streamcmds

classes:
```

```
      general=USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link
,FILE_Rename,FS_Chdir,FS_Chroot,PORT_Locked,PORT_Change,FS
_Mkdir,FS_Rmdir
      objects=S_ENVIRON_WRITE,S_GROUP_WRITE,S_LIMITS_WRITE,
S_LOGIN_WRITE,S_PASSWD_READ,S_PASSWD_WRITE,S_USER_WRITE,AU
D_CONFIG_WR
      SRC=SRC_Start,SRC_Stop,SRC_Addssys,SRC_Chssys,SRC_Del
ssys,SRC_Addserver,SRC_Chserver,SRC_Delserver
      kernel=PROC_Create,PROC_Delete,PROC_Execute,PROC_Real
UID,PROC_AuditID,PROC_RealGID,PROC_Environ,PROC_SetSignal,
PROC_Limits,PROC_SetPri,PROC_Setpri,PROC_Privilege,PROC_Se
ttimer
      files=FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_
Link,FILE_Unlink,FILE_Rename,FILE_Owner,FILE_Mode,FILE_Acl
,FILE_Privilege,DEV_Create
      svipc=MSG_Create,MSG_Read,MSG_Write,MSG_Delete,MSG_Ow
ner,MSG_Mode,SEM_Create,SEM_Op,SEM_Delete,SEM_Owner,SEM_Mo
de,SHM_Create,SHM_Open,SHM_Close,SHM_Owner,SHM_Mode
      mail = SENDMAIL_Config,SENDMAIL_ToFile
      cron=AT_JobAdd,AT_JobRemove,CRON_JobAdd,CRON_JobRemov
e,CRON_Start,CRON_Finish
      tcpip=TCPIP_config,TCPIP_host_id,TCPIP_route,TCPIP_co
nnect,TCPIP_data_out,TCPIP_data_in,TCPIP_access,TCPIP_set_
time,TCPIP_kconfig,TCPIP_kroute,TCPIP_kconnect,TCPIP_kdata
_out,TCPIP_kdata_in,TCPIP_kcreate
      lvm=LVM_AddLV,LVM_KDeleteLV,LVM_ExtendLV,LVM_ReduceLV
,LVM_KChangeLV,LVM_AvoidLV,LVM_MissingPV,LVM_AddPV,LVM_Add
MissPV,LVM_DeletePV,LVM_RemovePV,LVM_AddVGSA,LVM_DeleteVGS
A,LVM_SetupVG,LVM_DefineVG,LVM_KDeleteVG,LVM_ChgQuorum,LVM
_Chg1016,LVM_UnlockDisk,LVM_LockDisk,LVM_ChangeLV,LVM_Chan
geVG,LVM_CreateLV,LVM_CreateVG,LVM_DeleteVG,LVM_DeleteLV,L
VM_VaryoffVG,LVM_VaryonVG
      sidney = FILE_Read,PROC_Execute,FILE_Accessx

users:
      root = sidney
```

Os demais arquivos de configuração do subsistema de auditoria foram mantidos com os valores padrões. Para a geração de relatório de auditoria de sistema operacional foi utilizado o seguinte comando:

```
auditpr –d –helRtcrph –v
```

O relatório resultante do processamento dos dados capturados pelo subsistema de auditoria durante a execução do sistema Logix, para esse experimento, é apresentado nas próximas páginas:

```
event           login    status      time                     command                          real     process
--------------- -------- ----------- ------------------------ -------------------------------- -------- --------
FILE_Read       root     OK          Tue Apr 30 15:36:01 2002 ksh                              informix 13272
        file descriptor = 0
FILE_Read       root     OK          Tue Apr 30 15:36:28 2002 ksh                              informix 13272
        file descriptor = 63
FILE_Accessx    root     OK          Tue Apr 30 15:36:28 2002 ksh                              informix 11130
        mode: 0, who: 1, path: /usr/lib/nls/msg/en_US/ksh.cat
FILE_Read       root     OK          Tue Apr 30 15:36:28 2002 ksh                              informix 11130
        file descriptor = 3
PROC_Execute    root     OK          Tue Apr 30 15:36:28 2002 fglgo                            informix 11130
        euid: 200 egid: 11 epriv: 0 name
FILE_Read       root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        file descriptor = 3
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Read       root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        file descriptor = 4
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                            informix 11130
```

```
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0000.4gi
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 4, who: 0, path: /etc/security/passwd
```

```
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        file descriptor = 5
FILE_Accessx    root     OK          Tue Apr 30 15:36:29 2002 fglgo                           informix 11130
        mode: 1, who: 1, path: /u/informix/lib/sqlexec
PROC_Execute    root     OK          Tue Apr 30 15:36:29 2002 sqlexec                         informix 13950
        euid: 200 egid: 11 epriv: 0 name
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        file descriptor = 4
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        file descriptor = 3
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        file descriptor = 0
FILE_Accessx    root     FAIL        Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        mode: 0, who: 1, path: logix.dbs
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                         informix 13950
```

```
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                    informix 13950
        file descriptor = 3
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                    informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:36:30 2002 sqlexec                    informix 13950
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 sqlexec                    informix 13950
        file descriptor = 4
FILE_Read       root     OK          Tue Apr 30 15:36:30 2002 fglgo                      informix 11130
        file descriptor = 4
FILE_Read       root     OK          Tue Apr 30 15:36:33 2002 sqlexec                    informix 13950
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:40 2002 fglgo                      informix 11130
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:36:40 2002 fglgo                      informix 11130
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:36:41 2002 fglgo                      informix 11130
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:36:44 2002 fglgo                      informix 11130
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:36:44 2002 fglgo                      informix 11130
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:36:44 2002 fglgo                      informix 11130
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:36:45 2002 fglgo                      informix 11130
        file descriptor = 5
FS_Chdir        root     OK          Tue Apr 30 15:36:55 2002 rwhod                      root     5418
        change current directory to: /dev
FS_Chdir        root     OK          Tue Apr 30 15:36:55 2002 rwhod                      root     5418
        change current directory to: /usr/spool/rwho
FILE_Read       root     OK          Tue Apr 30 15:38:02 2002 fglgo                      informix 11130
        file descriptor = 0
FILE_Read       root     OK          Tue Apr 30 15:38:32 2002 fglgo                      informix 11130
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:38:59 2002 fglgo                      informix 11130
        file descriptor = 5
```

```
FILE_Read       root     OK          Tue Apr 30 15:39:00 2002 fglgo                            informix 11130
        file descriptor = 5
PROC_Execute    root     OK          Tue Apr 30 15:39:29 2002 ksh                              informix 13316
        euid: 200 egid: 11 epriv: 0 name
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 ksh                              informix 13316
        mode: 0, who: 1, path: /usr/lib/nls/msg/en_US/ksh.cat
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 ksh                              informix 13316
        file descriptor = 3
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 ksh                              informix 13316
        mode: 0, who: 1, path: /usr/lib/nls/msg/en_US/ksh.cat
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 ksh                              informix 13316
        file descriptor = 3
PROC_Execute    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        euid: 200 egid: 11 epriv: 0 name
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        file descriptor = 3
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        file descriptor = 4
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                            informix 13316
```

```
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK         Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
```

```
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 14135262151, who: 1948282722, path: ject read event detected /home/sidney/men0010.4gi
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        file descriptor = 5
FILE_Accessx    root     OK          Tue Apr 30 15:39:29 2002 fglgo                           informix 13316
        mode: 1, who: 1, path: /u/informix/lib/sqlexec
PROC_Execute    root     OK          Tue Apr 30 15:39:29 2002 sqlexec                         informix 14448
        euid: 200 egid: 11 epriv: 0 name
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                         informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                         informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                         informix 14448
```

```
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 4
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 3
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 0
FILE_Accessx    root     FAIL        Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        mode: 0, who: 1, path: logix.dbs
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 3
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 3
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        mode: 4, who: 0, path: /etc/security/passwd
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 sqlexec                    informix 14448
        file descriptor = 4
FILE_Read       root     OK          Tue Apr 30 15:39:29 2002 fglgo                      informix 13316
        file descriptor = 4
FILE_Unlink     root     OK          Tue Apr 30 15:39:29 2002 compress                   root     14706
        filename /audit/tempfile.00014190
FILE_Read       root     OK          Tue Apr 30 15:39:31 2002 sqlexec                    informix 14448
        file descriptor = 3
```

```
FILE_Read       root     OK          Tue Apr 30 15:39:32 2002 fglgo                    informix 13316
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:39:32 2002 fglgo                    informix 13316
        file descriptor = 5
FILE_Read       root     OK          Tue Apr 30 15:39:33 2002 fglgo                    informix 13316
        file descriptor = 5
FS_Chdir        root     OK          Tue Apr 30 15:39:55 2002 rwhod                    root     5418
        change current directory to: /dev
FS_Chdir        root     OK          Tue Apr 30 15:39:55 2002 rwhod                    root     5418
        change current directory to: /usr/spool/rwho
FILE_Read       root     OK          Tue Apr 30 15:40:13 2002 fglgo                    informix 13316
        file descriptor = 0
FS_Chdir        root     OK          Tue Apr 30 15:42:55 2002 rwhod                    root     5418
        change current directory to: /dev
FS_Chdir        root     OK          Tue Apr 30 15:42:55 2002 rwhod                    root     5418
        change current directory to: /usr/spool/rwho
FS_Chdir        root     OK          Tue Apr 30 15:45:55 2002 rwhod                    root     5418
        change current directory to: /dev
FS_Chdir        root     OK          Tue Apr 30 15:45:55 2002 rwhod                    root     5418
        change current directory to: /usr/spool/rwho
FILE_Rename     root     OK          Tue Apr 30 15:46:58 2002 sendmail                 root     13952
        frompath: tfQAA11394 topath: qfQAA11394
FILE_Unlink     root     OK          Tue Apr 30 15:46:58 2002 sendmail                 root     13952
        filename xfQAA11394
FILE_Accessx    root     FAIL_ACCESS Tue Apr 30 15:47:51 2002 ksh                      informix 13272
        mode: 1, who: 0, path: /usr/sbin/audit
FILE_Accessx    root     OK          Tue Apr 30 15:47:51 2002 ksh                      informix 13954
        mode: 0, who: 1, path: /usr/lib/nls/msg/en_US/ksh.cat
FILE_Read       root     OK          Tue Apr 30 15:47:51 2002 ksh                      informix 13954
        file descriptor = 3
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                      informix 13954
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                      informix 13954
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                      informix 13954
PROC_Execute    root     FAIL_ACCESS Tue Apr 30 15:47:51 2002 ksh                      informix 13954
FILE_Accessx    root     OK          Tue Apr 30 15:47:51 2002 ksh                      informix 13954
        mode: 0, who: 0, path: /usr/sbin/audit
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                      informix 13954
```

```
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                    informix 13954
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                    informix 13954
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                    informix 13954
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                    informix 13954
PROC_Execute    root     FAIL        Tue Apr 30 15:47:51 2002 ksh                    informix 13954
FILE_Read       root     OK          Tue Apr 30 15:47:51 2002 ksh                    informix 13954
        file descriptor = 10
```

ANEXO 4 – *LOG* DO SISTEMA GERENCIADOR DE BANCO DE DADOS

Antes da execução do sistema Logix para esse estudo de caso, o *log* do banco de dados foi forçado para uma nova área de armazenamento de *log* através do comando `onmode -l`, para garantir que somente os dados pertencentes a essa execução seriam gravados nessa área. A identificação da área de *log* para esse experimento foi a de número 20. Após a execução dos programas do sistema Logix referentes a esse experimento, foi novamente executado o comando `onmode -l` para que essa área de *log* não fosse mais utilizada por outras solicitações ao gerenciador de banco de dados. Para a geração de relatório de *log* foi utilizado o comando:

```
onlog –n 20 -l
```

onde o parâmetro `-n 20` identifica a área de *log* que contém os registros gerados durante o experimento. Nas próximas páginas é apresentada a saída do programa `onlog`:

```
Informix Dynamic Server  Logical Log display
Software Serial Number AAC#A622840
Copyright (C) 1987-1998 Informix Software, Inc.


log number: 20.

addr     len  type     xid      id link
18       32   CKPOINT  1        1  e018     0
         00000020 00010042 00100000 00000001 ... ...B ........
         0000e018 00017cc5 00000000 00000000 ......|. ........
1018     40   BEGIN    5        20 0        04/30/2002 15:39:20 14       informix
         00000028 00140001 00000000 00000005 ...(.... ........
         00000000 00017d2e 00000000 3cceac98 ......}. ....<...
         000000c8 0000000e                   ........
1040     52   HUPDAT   5        0  1018     10004d   309      0        13  13  1
         00000034 00000049 00100000 00000005 ...4...I ........
         00001018 00017d2e 00000000 0010004d ......}. .......M
         00000309 00000000 000d000d 0001000c ........ ........
         00010203                            ....
1074     36   COMMIT   5        0  1040     04/30/2002 15:39:20
         00000024 00000002 00100000 00000005 ...$.... ........
         00001040 00017d2f 00000000 3cceac98 ...@..}/ ....<...
         3cceac98                            <...
1098     40   BEGIN    7        20 0        04/30/2002 15:41:00 15       informix
         00000028 00140001 00000000 00000007 ...(.... ........
         00000000 00017d53 00000000 3cceacfc ......}S ....<...
         000000c8 0000000f                   ........

addr     len  type     xid      id link
10c0     60   HINSERT  7        0  1098     100052   30a      13
         0000003c 00000028 01120000 00000007 ...<...( ........
         00001098 00017d53 00100052 00100052 ......}S ...R...R
         0000030a 000d0000 00000000 7369646e ........ ....sidn
         65792020 30310001 54000000          ey  01.. T...
10fc     60   ADDITEM  7        0  10c0     100052   30a      1     1     10
         0000003c 0000001c 00100000 00000007 ...<.... ........
```

```
        000010c0 00017d57 00000000 00100052 ......}W .......R
        00100052 0000030a 00000001 0001000a ...R.... ........
        7369646e 65792020 30310000          sidney   01..
1138    180  HINSERT  7        0  10fc     100056   20c      135
        000000b4 00000028 01120000 00000007 .......( ........
        000010fc 00017d59 00100056 00100056 ......}Y ...V...V
        0000020c 00870000 00000000 7369646e ........ ....sidn
        65792020 30310020 20202020 20202020 ey  01.
        5349444e 45592020 20202020 20202020 SIDNEY
        20202020 20202020 20202020 20200020                .
        20202020 20202020 20202020 20202020
        20200020 20202000 20202020 20202020   .    .
        20202020 20202020 20202020 20202020
        20202020 20002020 20202020 20202020      .
        20202020 20202020 20202020 20202020
        20202000                                .

addr    len  type     xid      id link
11ec    56   ADDITEM  7        0  1138     100056   20c      1     1     8
        00000038 0000001c 00100000 00000007 ...8.... ........
        00001138 00017d5d 00000000 00100056 ...8..}] .......V
        00100056 0000020c 00000001 00010008 ...V.... ........
        7369646e 65792020                   sidney
1224    36   COMMIT   7        0  11ec     04/30/2002 15:41:02
        00000024 00000002 00100000 00000007 ...$.... ........
        000011ec 00017d5e 00000000 3cceacfe ......}^ ....<...
        3cceacfc                            <...
1248    40   BEGIN    8        20 0        04/30/2002 15:41:41 15       informix
        00000028 00140001 00000000 00000008 ...(.... ........
        00000000 00017d6d 00000000 3ccead25 ......}m ....<..%
        000000c8 0000000f                   ........
1270    128  BLDCL    8        0  1248     1000ba 8 8 147 0 _temptable
        00000080 00000020 00100000 00000008 .......  ........
        00001248 00017d6d 00000000 001000ba ...H..}m ........
        00000008 00000008 00930021 6c6f6769 ........ ...!logi
        78000000 00100056 00000001 00000069 x......V .......i
        6e666f72 6d697800 5f74656d 70746162 nformix. _temptab
```

```
        6c65009c 200bd2a0 200bd965 6e5f5553 le.. ...  ..en_US
        2e383139 00390b28 22023480 10074494 .819.9.( ".4...D.
        00000000 00000000 00000000 00000000 ........ ........

addr    len  type     xid      id link
12f0    36   CHALLOC  8        0  1270     100ad9   8
        00000024 00000033 00100000 00000008 ...$...3 ........
        00001270 00017d6f 00000000 00100ad9 ...p..}o ........
        00000008                            ....
1314    40   PTEXTEND 8        0  12f0     1000ba   7        100ad9
        00000028 00000032 00100000 00000008 ...(...2 ........
        000012f0 00017d70 00000000 001000ba ......}p ........
        00000007 00100ad9                   ........
133c    36   COMMIT   8        0  1314     04/30/2002 15:41:41
        00000024 00000002 00100000 00000008 ...$.... ........
        00001314 00017d75 00000000 3ccead25 ......}u ....<..%
        3ccead25                            <..%
1360    40   BEGIN    7        20 0        04/30/2002 15:42:39 15       informix
        00000028 00140001 00000000 00000007 ...(.... ........
        00000000 00017d8b 00000000 3ccead5f ......}. ....<.._
        000000c8 0000000f                   ........
1388    80   HUPDAT   7        0  1360     100056   20c      0        135 135 1
        00000050 00000049 00100000 00000007 ...P...I ........
        00001360 00017d8b 00000000 00100056 ...`..}. .......V
        0000020c 00000000 00870087 0001001b ........ ........
        000f2020 20202020 20202020 20202020 ..
        20564943 454e5445 204d454e 44455320  VICENTE  MENDES

addr    len  type     xid      id link
13d8    36   COMMIT   7        0  1388     04/30/2002 15:42:39
        00000024 00000002 00100000 00000007 ...$.... ........
        00001388 00017d8c 00000000 3ccead5f ......}. ....<.._
        3ccead5f                            <.._
2018    32   CKPOINT  1        0  18       0
        00000020 00000042 00100000 00000001 ... ...B ........
        00000018 00017d95 00000000 00000000 ......}. ........
3018    40   BEGIN    8        20 0        04/30/2002 15:43:21 15       informix
```

```
         00000028 00140001 00000000 00000008 ...(.... ........
         00000000 00017d9d 00000000 3ccead89 ......}. ....<...
         000000c8 0000000f                   ........
3040     32   ERASE    8        0  3018     1000ba
         00000020 0000000d 00100000 00000008 ... .... ........
         00003018 00017d9d 00000000 001000ba ..0...}. ........
3060     36   CHFREE   8        0  3040     100ad9   8
         00000024 00000034 00100000 00000008 ...$...4 ........
         00003040 00017d9e 00000000 00100ad9 ..0@..}. ........
         00000008                            ....

addr     len  type     xid      id link
3084     36   COMMIT   8        0  3060     04/30/2002 15:43:21
         00000024 00000002 00100000 00000008 ...$.... ........
         00003060 00017da2 00000000 3ccead89 ..0`..}. ....<...
         3ccead89                            <...
30a8     40   BEGIN    8        20 0        04/30/2002 15:43:21 15       informix
         00000028 00140001 00000000 00000008 ...(.... ........
         00000000 00017da7 00000000 3ccead89 ......}. ....<...
         000000c8 0000000f                   ........
30d0     128  BLDCL    8        0  30a8     1000ba 8 8 147 0 _temptable
         00000080 00000020 00100000 00000008 .......  ........
         000030a8 00017da7 00000000 001000ba ..0...}. ........
         00000008 00000008 00930021 6c6f6769 ........ ...!logi
         78000000 00100056 00000001 00000069 x......V .......i
         6e666f72 6d697800 5f74656d 70746162 nformix. _temptab
         6c65009c 200bd2a0 200bd965 6e5f5553 le.. ...  ..en_US
         2e383139 00390b28 22023480 10074494 .819.9.( ".4...D.
         00000000 00000000 00000000 00000000 ........ ........
3150     36   CHALLOC  8        0  30d0     100ad9   8
         00000024 00000033 00100000 00000008 ...$...3 ........
         000030d0 00017da8 00000000 00100ad9 ..0...}. ........
         00000008                            ....

addr     len  type     xid      id link
3174     40   PTEXTEND 8        0  3150     1000ba   7        100ad9
         00000028 00000032 00100000 00000008 ...(...2 ........
```

```
         00003150 00017da9 00000000 001000ba ..1P..}. ........
         00000007 00100ad9                   ........
319c     36   COMMIT   8        0  3174     04/30/2002 15:43:21
         00000024 00000002 00100000 00000008 ...$.... ........
         00003174 00017dae 00000000 3ccead89 ..1t..}. ....<...
         3ccead89                            <...
31c0     40   BEGIN    7        20 0        04/30/2002 15:44:09 15       informix
         00000028 00140001 00000000 00000007 ...(.... ........
         00000000 00017dc4 00000000 3cceadb9 ......}. ....<...
         000000c8 0000000f                   ........
31e8     60   HDELETE  7        0  31c0     100052   30a      13
         0000003c 00000029 01120000 00000007 ...<...) ........
         000031c0 00017dc4 00100052 00100052 ..1...}. ...R...R
         0000030a 000d0200 00000000 7369646e ........ ....sidn
         65792020 30310001 54000000          ey  01.. T...
3224     60   DELITEM  7        0  31e8     100052   30a      1     1     10
         0000003c 0000001d 00100000 00000007 ...<.... ........
         000031e8 00017dc9 00000000 00100052 ..1...}. .......R
         00100052 0000030a 00000001 0001000a ...R.... ........
         7369646e 65792020 30310001          sidney   01..

addr     len  type     xid      id link
3260     180  HDELETE  7        0  3224     100056   20c      135
         000000b4 00000029 01120000 00000007 .......) ........
         00003224 00017dcb 00100056 00100056 ..2$..}. ...V...V
         0000020c 00870200 00000000 7369646e ........ ....sidn
         65792020 30310020 20202020 20202020 ey  01.
         5349444e 45592056 4943454e 5445204d SIDNEY V ICENTE M
         454e4445 53202020 20202020 20200020 ENDES          .
         20202020 20202020 20202020 20202020
         20200020 20202000 20202020 20202020   .    .
         20202020 20202020 20202020 20202020
         20202020 20002020 20202020 20202020      .
         20202020 20202020 20202020 20202020
         20202000                              .
3314     56   DELITEM  7        0  3260     100056   20c      1     1     8
         00000038 0000001d 00100000 00000007 ...8.... ........
```

```
        00003260 00017dcf 00000000 00100056 ..2`..}. .......V
        00100056 0000020c 00000001 00010008 ...V.... ........
        7369646e 65792020                    sidney

addr    len  type     xid      id link
334c    36   COMMIT   7        0  3314     04/30/2002 15:44:13
        00000024 00000002 00100000 00000007 ...$.... ........
        00003314 00017dd0 00000000 3cceadbd ..3...}. ....<...
        3cceadb9                            <...
3370    40   BEGIN    8        20 0        04/30/2002 15:44:36 15       informix
        00000028 00140001 00000000 00000008 ...(.... ........
        00000000 00017dd3 00000000 3cceadd4 ......}. ....<...
        000000c8 0000000f                   ........
3398    32   ERASE    8        0  3370     1000ba
        00000020 0000000d 00100000 00000008 ... .... ........
        00003370 00017dd3 00000000 001000ba ..3p..}. ........
33b8    36   CHFREE   8        0  3398     100ad9   8
        00000024 00000034 00100000 00000008 ...$...4 ........
        00003398 00017dd3 00000000 00100ad9 ..3...}. ........
        00000008                            ....
33dc    36   COMMIT   8        0  33b8     04/30/2002 15:44:36
        00000024 00000002 00100000 00000008 ...$.... ........
        000033b8 00017dd6 00000000 3cceadd4 ..3...}. ....<...
        3cceadd4                            <...

addr    len  type     xid      id link
3400    40   BEGIN    5        20 0        04/30/2002 15:44:36 14       informix
        00000028 00140001 00000000 00000005 ...(.... ........
        00000000 00017de3 00000000 3cceadd4 ......}. ....<...
        000000c8 0000000e                   ........
3428    52   HUPDAT   5        0  3400     10004d   309      0        13  13  1
        00000034 00000049 00100000 00000005 ...4...I ........
        00003400 00017de3 00000000 0010004d ..4...}. .......M
        00000309 00000000 000d000d 0001000c ........ ........
        00010302                            ....
345c    36   COMMIT   5        0  3428     04/30/2002 15:44:36
        00000024 00000002 00100000 00000005 ...$.... ........
```

```
         00003428 00017de4 00000000 3cceadd4 ..4(..}. ....<...
         3cceadd4                           <...
4018     32   CKPOINT  1        0  2018     0
         00000020 00000042 00100000 00000001 ... ...B ........
         00002018 00017df7 00000000 00000000 .. ...}. ........
```