Auditoria de Comandos DML em Bancos de Dados Oracle e Firebird

Article

2 authors, including:

Cristiano R. Cervi
Universidade de Passo Fundo
42 PUBLICATIONS **19** CITATIONS

SEE PROFILE

Auditoria de Comandos DML em Bancos de Dados Oracle e Firebird

Rogério Marros Teixeira[1], Cristiano Roberto Cervi[1]

[1]Instituto de Ciências Exatas e Geociências – Universidade de Passo Fundo (UPF)
Caixa Postal 611 – 99.052-900 – Passo Fundo – RS – Brasil

{marros,cervi}@upf.br

Abstract. *This paper discusses issues and implications of data auditing in enterprise systems, and presents a tool to generate, manage and query historical data in Oracle and Firebird databases. Furthermore, it is presented the experiments with the proposed tool and the analysis of obtained results.*

Resumo. *Este artigo discute problemas e implicações sobre auditoria de dados em sistemas corporativos, bem como apresenta uma ferramenta para gerar, gerenciar e consultar dados históricos em bancos de dados Oracle e Firebird. Ainda, são apresentados experimentos realizados com a ferramenta proposta e análises dos resultados obtidos.*

1. Introdução

Em sistemas corporativos é comum, e praticamente inevitável, que vários usuários de um ou mais departamentos utilizem os mesmos módulos de aplicativos, incluindo, alterando e excluindo dados. Dados estes que podem pertencer a uma simples tabela de manutenção básica ou a uma tabela mais complexa, como por exemplo, uma tabela de indicadores para cálculos financeiros, ou uma tabela com variáveis para precificação de produtos. Um equívoco no cadastramento, em qualquer dos casos, pode comprometer o resultado final da aplicação.

Alguns erros nas informações geradas por aplicações computacionais estão relacionados diretamente com o cadastramento equivocado ou uma alteração indevida de dados. Parte desses problemas pode vir a ser percebidos em um espaço de tempo significativo entre o fato ocorrido e o resultado apresentado, tornando sua solução mais difícil. Em alguns casos, pode ser necessário saber o momento exato em que ocorreu o equívoco no cadastramento do dado, para que o resultado gerado pelo aplicativo possa ser corrigido. Pode haver, ainda, a necessidade de saber qual usuário gerou o dado equivocado e em que ponto de trabalho a operação equivocada foi realizada.

Segundo Cantu (2009), não são raras as situações onde precisa-se descobrir quem manipulou determinada informação na base de dados, seja inserindo, alterando ou removendo registros. Nos casos em que registros são excluídos, pode-se ter a necessidade de saber exatamente o que existia nestes determinados registros, para que o problema seja solucionado a contento.

Para que os problemas possam ser resolvidos, precisa-se armazenar os dados históricos de todas as operações que ocasionem alterações de dados realizadas nas tabelas vitais de um sistema computacional. Uma das formas de realizar esta tarefa é

armazenando todas estas alterações de dados em uma tabela auxiliar, organizando-as de forma cronológica.

O objetivo deste trabalho é apresentar uma solução para o problema descrito anteriormente, através do desenvolvimento da ferramenta Audi-DML, projetada para sistemas que operam com bancos de dados Oracle e Firebird. A solução é baseada em *triggers*[1] que são disparados a cada operação DML (*Data Manipulation Language* – Linguagem de Manipulação de Dados) que ocasionarem alterações de dados, armazenando em uma tabela auxiliar o valor dos dados anteriores a ação efetuada pelo comando executado.

Este artigo está dividido da seguinte forma. A seção 2 descreve uma breve explanação sobre auditoria de dados. Na seção 3 é descrita a Audi-DML, suas funcionalidades e a comparação com outras ferramentas. Os experimentos com a Audi-DML, bem como a análise de resultados são descritos na seção 4. Finalmente, na seção 5, são apresentadas as considerações finais e perspectivas de trabalhos futuros.

2. Auditoria de Dados

Auditoria é o controle e registro das ações de usuários de bancos de dados, que pode ser baseada em ações individuais, como o tipo de instrução SQL (*Structured Query Language* – Linguagem de Consulta Estruturada) executada, ou em combinações de fatores que podem incluir nome de usuário, aplicação, tempo, entre outros. Oracle (2008).

Conforme Oliveira (2008), em muitas organizações existe a necessidade que boa parte das operações executadas em ambientes transacionais possam ser auditadas *a posteriori*. Práticas de auditoria são comuns e muito úteis no processo de identificação de discrepâncias na base ou no processo de correção de problemas gerados por *bugs* na aplicação.

Para que se possa identificar a origem de um dado cadastrado errado em uma base de dados, precisa-se manter um histórico de todas as alterações feitas nos dados das tabelas, ou pelo menos, das tabelas que contenham os dados críticos das aplicações. Estes históricos, também chamados de *logs*[2], são muito importantes para aplicativos que manipulam dados confidenciais ou críticos, porque nesses *logs* podem ser armazenadas a data e a hora das alterações, o usuário que as fez, o endereço IP da estação de trabalho, e outros dados que se julgar necessários para uma auditoria.

Segundo Cantu (2009), um mecanismo de auditoria pode ser implementado em bancos de dados relacionais de forma relativamente simples, através do uso de *triggers* e tabelas de *log*. No entanto, o processo é trabalhoso, pois as tabelas a serem monitoradas devem ser "preparadas" com a criação dos *triggers*. Além disso, nem sempre deseja-se monitorar todos os campos de uma tabela. Dados que não são importantes podem ficar de fora, contribuindo para a redução das informações geradas, e até mesmo para uma maior facilidade na análise das informações.

---

[1] Gatilhos. São procedimentos executados automaticamente em resposta a certos eventos a eles vinculados.

[2] Registro de atividades gerado por programas de computador.

Os *logs* de auditoria também podem ser um importante ponto de apoio para a depuração de novas aplicações durante a fase de implantação, já que fornecem uma visão fidedigna do estado dos dados em um determinado espaço no tempo.

3. Apresentação da Ferramenta

Esta seção apresenta a Audi-DML, desenvolvida para o gerenciamento e consulta de auditoria de comandos DML. Ela pode ser utilizada para gerar *logs* de auditoria em sistemas construídos para bases de dados Oracle e Firebird, padronizando a forma de auditoria nestes diferentes bancos de dados.

A ferramenta, ao ser instalada em uma base de dados, gera inicialmente uma estrutura que consiste apenas em duas tabelas de controle e um *role*[3] para permissões de acesso, como mostra a Figura 1. A cada tabela selecionada para ser auditada, a Audi-DML gera uma tabela auxiliar para armazenamento dos dados a serem auditados, e um *trigger* para inserção dos dados históricos na tabela auxiliar.

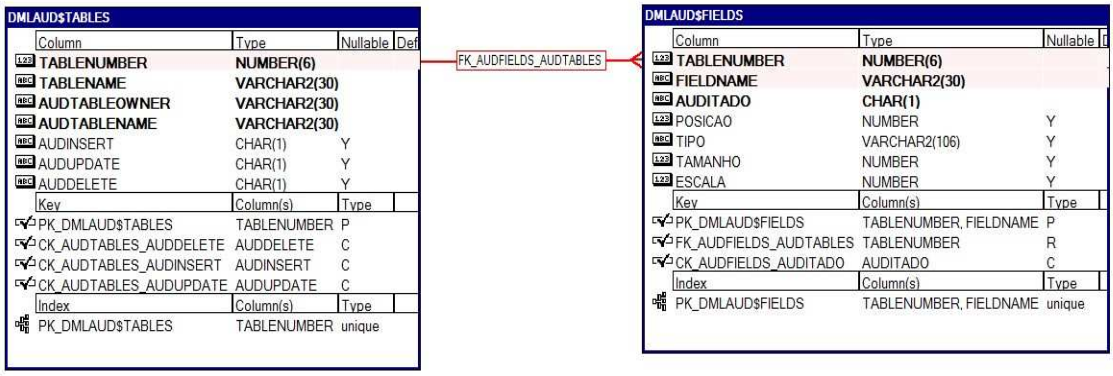

Figura 1. Estrutura de controle da ferramenta Audi-DML

A cada alteração nos registros da tabela auditada, geradas pelos comandos *insert*, *update* ou *delete*, a ferramenta insere na tabela auxiliar os dados anteriores a alteração feita, além de dados adicionais que consistem em usuário, momento (*timestamp*), ação e endereço IP ou nome da estação de trabalho onde a alteração foi executada.

As tabelas auxiliares são criadas sem índices nem chaves primárias ou únicas, para diminuir o impacto no desempenho das aplicações que utilizam as tabelas auditadas.

3.1. Descrição de Funcionalidades

A ferramenta trabalha com bancos de dados Oracle, versões 9i ou superior, e Firebird versão 2.x, e é dividida em duas seções principais: (i) seção de gerenciamento de auditoria; e (ii) seção de consulta aos dados auditados. Na seção de gerenciamento a conexão com o banco de dados é feita somente por usuários com permissões de administrador do banco de dados. Já na seção de consulta, a conexão pode ser feita por todos os usuários do banco de dados que tiverem permissão de consulta as tabelas de auditoria.

---

[3] Conjunto ou grupo de privilégios que pode ser concedido a usuários ou outros roles.

Tanto a seção de gerenciamento, quanto a seção de consulta são subdivididas em telas, sendo que em ambas, a primeira tela permite a seleção do banco de dados a ser conectado, podendo ser Oracle ou Firebird.

A seção de gerenciamento é composta de quatro telas para conexões com Firebird e cinco telas para conexões com Oracle, apresentadas de forma seqüencial, no estilo *wizard*[4], e não é permitido que nenhuma delas seja acessada sem que se tenha passado pelas telas anteriores.

As telas disponíveis para conexões com Firebird são: Credenciais, Tabelas, Permissões e Gráficos. Já as telas disponíveis para conexões com Oracle são: Credenciais, Auditoria, Tabelas, Permissões e Gráficos.

Na tela de Credenciais são solicitadas as informações para possibilitar a conexão com o banco de dados, como usuário, senha, base de dados, dentre outros elementos. Só é possível conectar com usuário que tenha permissão de administração do banco de dados.

É na tela de Auditoria, que está disponível apenas para as conexões com Oracle, que é solicitado o nome do usuário que será o proprietário das tabelas de controle da auditoria, bem como das tabelas auxiliares, aquelas que serão geradas para cada tabela a ser auditada. É nela também que é informado o *tablespace*[5] onde estas tabelas ficarão armazenadas.

A informação do usuário proprietário das tabelas de auditoria é solicitada somente no primeiro acesso a uma base de dados, que é o momento em que a ferramenta cria a estrutura de controle de auditoria.

Diferentemente do usuário, a informação do *tablespace* pode ser alterada em qualquer outro acesso. Isso permite que as tabelas de auditoria possam ser alocadas em mais de um *tablespace*. Assim, o administrador do banco de dados pode optar por distribuir o armazenamento das tabelas de auditoria como achar apropriado. Por exemplo: as tabelas auxiliares para auditoria de um sistema de estoque podem estar alocadas em um *tablespace* diferente das tabelas auxiliares para auditoria de um sistema de recursos humanos.

Destaca-se que tanto o usuário que será escolhido para ser o proprietário das tabelas de auditoria, como o *tablespace* onde elas serão armazenadas, devem estar previamente criados no banco de dados, bem como as permissões de quotas de utilização da *tablespace* pelo usuário.

Embora a ferramenta permita apenas a seleção de usuários cadastrados no banco de dados e que tenham permissão para criação de objetos, impedindo assim que seja informado, por engano, um usuário inexistente ou sem permissão de criação de objetos, a informação do usuário proprietário das tabelas de auditoria não poderá ser alterada posteriormente. Portanto, um bom planejamento por parte do administrador do banco de dados é fundamental.

---

[4] Assistente utilizado em interface gráfica que provê um meio simples de realizar tarefas complexas em sistemas computacionais, fazendo uso de um esquema passo-a-passo.

[5] Unidade lógica de armazenamento dentro do banco de dados Oracle.

Na tela de Tabelas, a ferramenta permite a seleção das tabelas a serem auditadas, a seleção dos campos a auditar para a tabela selecionada e a seleção das ações a auditar (inserções, alterações e exclusões) na tabela. Ainda nesta tela, são apresentadas as tabelas que já estão preparadas para auditoria bem como seus respectivos campos. Isto permite que o usuário possa interromper a auditoria de determinados campos, de toda a tabela, reiniciar a auditoria de um campo ou reiniciar a auditoria de uma tabela que tenha sido interrompida anteriormente. Ainda, pode alterar as ações que estão sendo auditadas para uma determinada tabela.

Nesta tela as tabelas a auditar e seus campos ainda não auditados são apresentados à direita, enquanto as tabelas já auditadas e os campos auditados são listados a esquerda, como mostra a Figura 2.

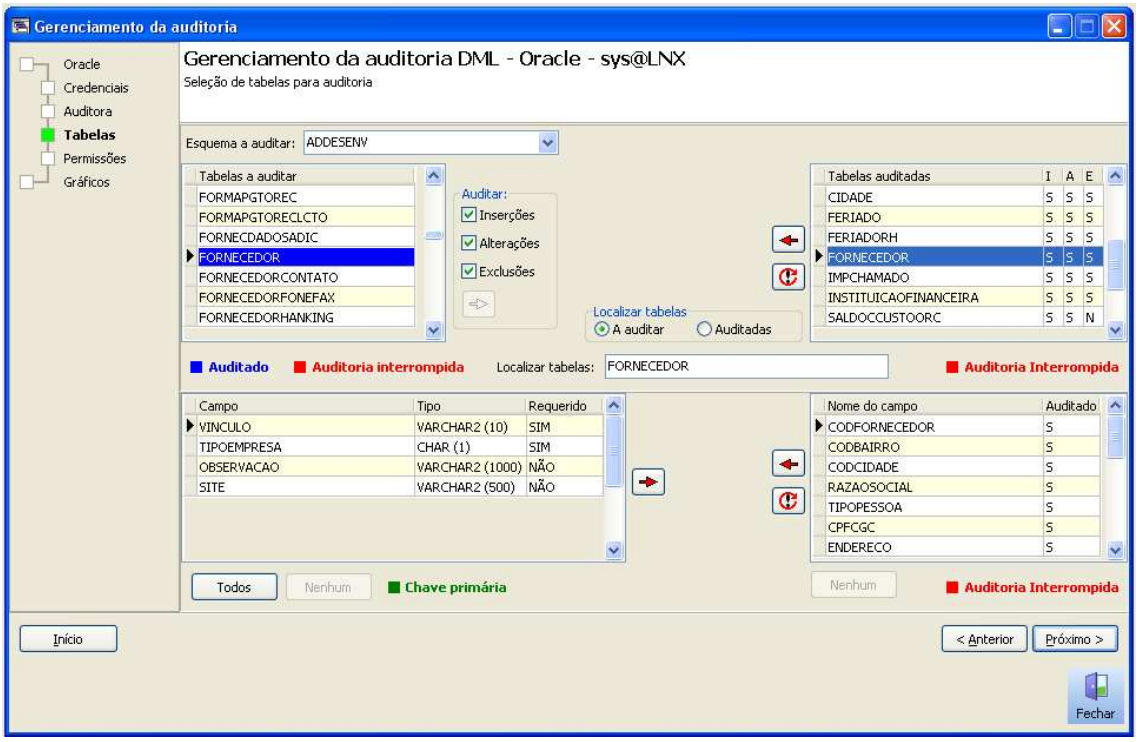

Figura 2. Tela de tabelas da seção de gerenciamento de auditoria

A tela de Permissões possibilita a concessão (ou retirada) de permissões de consulta aos dados auditados para os usuários cadastrados no banco de dados. Tais permissões podem ser concedidas ou retiradas a qualquer momento. Isso permite ao administrador do banco de dados um fácil controle destes acessos.

A última tela da seção de gerenciamento é a tela de Gráficos. Ela fornece uma combinação de gráficos para facilitar a visualização do crescimento da quantidade de registros nas tabelas de auditoria. Estes gráficos podem ser gerados através da combinação de três fatores: (i) quantidade de registros por ação (inserção, alteração e exclusão) ou independente da ação; (ii) periodicidade anual ou mensal; e (iii) abrangência geral, por tabela ou por esquema, sendo que a abrangência por esquema somente está disponível para auditorias em bancos de dados Oracle.

A Figura 3 apresenta um gráfico de quantidade de registros por ações, com periodicidade anual e abrangência geral. Ele pode ser útil para detectar um comportamento anormal no crescimento dos registros de auditoria. Quando isso ocorrer,

a geração de um novo gráfico, com periodicidade mensal, pode dar uma posição mais aproximada do período em que o comportamento anormal aconteceu, facilitando sua identificação.

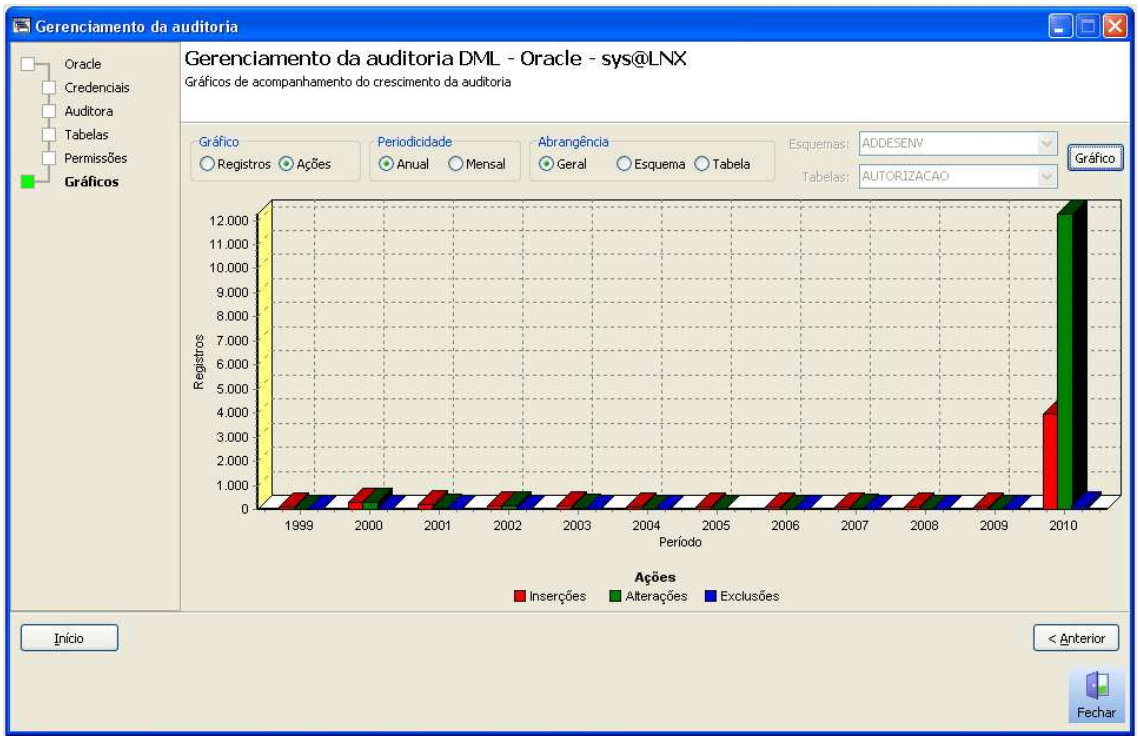

Figura 3. Gráfico de registros de auditoria por ações, anual e geral

A seção de consulta é composta de três telas, tanto para as conexões com Firebird quanto para as conexões com Oracle, apresentadas de forma seqüencial, no estilo *wizard*, onde não é permitindo que nenhuma tela seja acessada sem que se tenha passado pelas anteriores. As telas disponíveis para a seção de consulta, tanto para Firebird quanto para Oracle são: Credenciais, Consulta e Verificação.

Na tela de Credenciais são solicitadas informações para possibilitar a conexão com o banco de dados, como usuário, senha, base de dados, dentre outros elementos. Só é possível conectar com usuário que tenha permissão de consulta a auditoria, sendo esta permissão concedida pelo administrador do banco de dados na seção de gerenciamento da auditoria.

A tela de Consulta permite que o usuário construa consultas mesmo sem ter conhecimento de linguagem SQL. Nela, o usuário pode selecionar a tabela auditada que desejar consultar os dados históricos, selecionar quais campos de dados desta tabela ele deseja utilizar como filtro para a pesquisa, selecionar a condição para comparação (dependendo do tipo do dado) e informar o valor que será utilizado como parâmetro para a consulta daquele campo de dados. A ferramenta permite a combinação de todos os campos auditados da tabela para a construção dos filtros, permitindo duplicação, o que possibilita a implementação de filtros com intervalos.

Além dos campos de dados das tabelas, ainda é possível selecionar os campos adicionais para construção de filtros, como usuário, ação, *timestamp* e endereço IP. O

usuário ainda pode informar se deseja que todas as regras selecionadas sejam atendidas (*and*) ou se apenas uma das regras precisa ser atendida (*or*) para executar a pesquisa.

A Figura 4 demonstra uma consulta aos dados históricos da tabela Cidade, cujo objetivo é procurar alterações feitas no cadastro de uma determinada cidade.

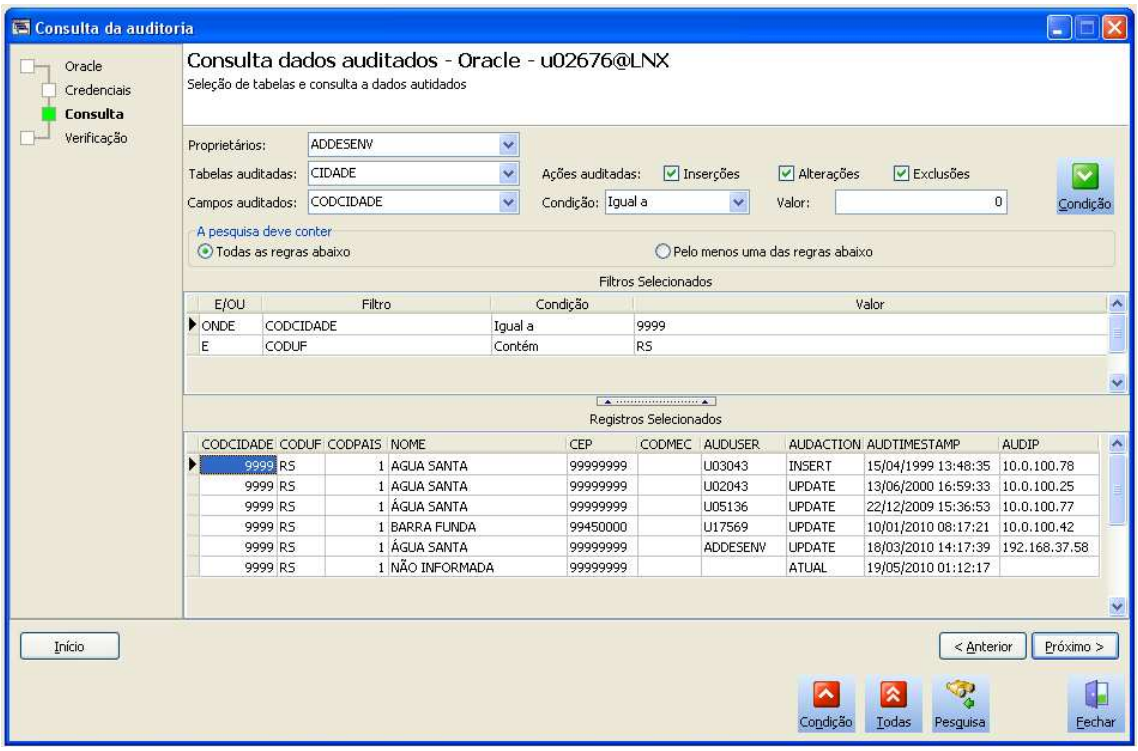

Figura 4. Consulta a dados auditados

A tabela auxiliar, responsável pelo armazenamento dos dados históricos, armazena o conteúdo dos campos, anterior a ação executada. Dessa forma, para saber o que exatamente foi alterado em determinado momento, é necessário a comparação de dois registros, aquele selecionado e o imediatamente posterior. Para facilitar este trabalho, a ferramenta possui uma tela de verificação que demonstra as alterações ao usuário.

Na última tela da seção de consulta (tela de Verificação), a ferramenta apresenta os registros resultantes da consulta efetuada, em ordem cronológica. Neste momento, o usuário pode escolher um registro para que a ferramenta demonstre com mais destaque os dados alterados naquela ocorrência, bem como os dados adicionais que foram gerados. Esses dados correspondem ao usuário que executou a ação, qual ação foi efetuada (inserção, alteração ou exclusão), o momento da ação (*timestamp*) e o local da ação (endereço IP ou nome da estação de trabalho).

A Figura 5 mostra o detalhamento de uma alteração em um registro do cadastro de cidades.

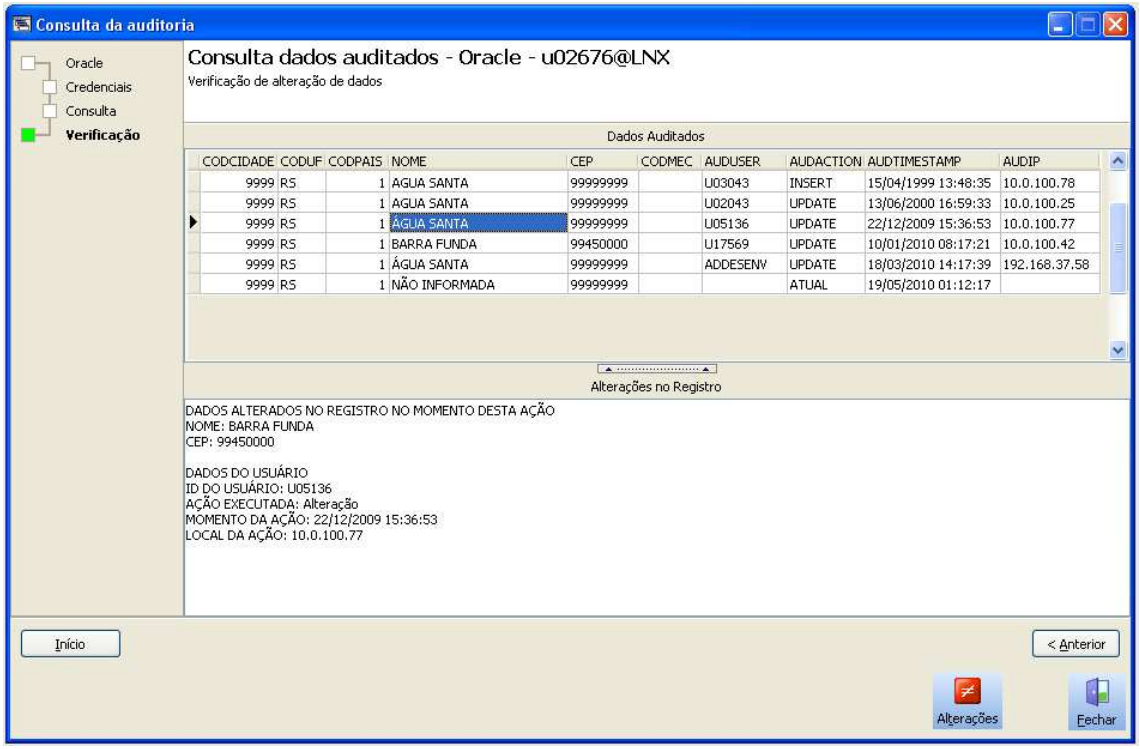

Figura 5. Verificação de alteração em registro

3.2. Comparação Com Outras Ferramentas

Existem outras maneiras de armazenar dados históricos, para verificações futuras, em um banco de dados. Algumas delas são recursos distribuídos com o próprio banco de dados, como é o caso da opção *audit_trail*[6] do banco de dados Oracle. Outras são ferramentas de terceiros, como o IB LogManager (Upscene Productions 2010), desenvolvido para os bancos de dados InterBase e Firebird.

A opção de auditoria do banco de dados Oracle, *audit_trail*, deve ser ativada utilizando os parâmetros de inicialização do banco de dados, que ficam no arquivo de configuração `init.ora`. Ele é um parâmetro estático, ou seja, para alterá-lo, deve-se parar a execução do banco de dados (*shutdown*), alterar o valor do parâmetro e reiniciá-lo novamente (*startup*).

As opções do parâmetro de auditoria para *audit_trail* são: (i) NONE, para desabilitar a auditoria; (ii) OS, habilita a auditoria e direciona os registros de auditoria para arquivos do sistema operacional; (iii) DB, habilita a auditoria e direciona os registros de auditoria para uma tabela na base de dados (tabela `sys.aud$`); (iv) DB EXTENDED, idem a DB, porém popula também os campos `sqlbind`[7] e `sqltext`[8]; (v) XML, habilita a auditoria e direciona os registros de auditoria para arquivos XML; (vi) XML EXTENDED, idem a XML porém inclui os campos `sqlbind` e `sqltext`.

---

[6] Parâmetro de inicialização do banco de dados Oracle, que indica se a auditoria está ou não ativa.

[7] Campo da tabela de auditoria do banco de dados Oracle, nela são armazenadas as variáveis de ligação dos comandos SQL.

[8] Campo da tabela de auditoria do banco de dados Oracle, nela são armazenados os comandos SQL auditados.

A opção de auditoria *audit_trail* permite gravar as seguintes informações de ações: *login*[9] e *logoff*[10]; comandos DML (incluindo o comando SELECT); comandos DDL (*Data Definition Language* – Linguagem de Definição de Dados); auditoria de ações e comandos bem ou mal sucedidos; armazenamento de sentenças SQL e suas variáveis de ligação (*bind variables*) quando o valor do parâmetro de inicialização for DB EXTENDED ou XML EXTENDED. Porém, não armazena os dados históricos dos campos de tabelas. Para poder armazenar estas informações adicionais é necessário a criação de *triggers* persolnalizadas (Oracle, 2008). Há, portanto, a necessidade da construção de soluções próprias para armazenar dados históricos de campos de tabelas e também para poder disponibilizar a consulta a esses dados para o usuário final.

As diferenças entre a Audi-DML e o *audit_trail* são diversas, a finalidade do *audit_trail* é armazenar dados históricos de comandos, bem ou mal sucedidos, enquanto a Audi-DML tem por finalidade o armazenamento do histórico de dados alterados em colunas de tabelas, quando as operações DML são bem sucedidas.

A solução comercial IB LogManager é utilizada para auditoria de dados em bancos de dados InterBase e Firebird. Ela foi desenvolvida pela empresa Holandesa Upscene Productions, cujo foco principal é criar software específico para desenvolvedores de aplicações em bancos de dados (Upscene Productions 2010). Esta solução, assim como a Audi-DML, também utiliza *triggers* para geração das auditorias, porém sua estrutura de controle consiste em: quatro tabelas, uma *view*[11], dois *generators*[12], duas *stored procedures*[13], *alguns índices além de outros objetos*. A cada tabela auditada são criados entre um e três *triggers*, dependendo das ações que serão auditadas para a tabela (inserção, alteração ou exclusão). Já na Audi-DML, são criadas apenas duas tabelas para controle da estrutura de auditoria, mais uma tabela auxiliar e um *trigger* para cada tabela auditada.

O IB LogManager, diferente da Audi-DML, permite a auditoria de ações diferenciadas por campo de tabela, ou seja, enquanto um campo de uma tabela está sendo auditado para ações de inserção e exclusão, outro campo da mesma tabela pode estar sendo auditado para ações de alteração, ou qualquer outra combinação de ações.

Enquanto na Audi-DML são criadas tabelas auxiliares que armazenam os dados históricos (anteriores as ações efetuadas sobre as tabelas originais, e ainda os dados adicionais, gerando um registro para cada ação), o IB LogManager armazena em uma de suas tabelas de controle um registro para cada campo de tabela auditada, contendo o valor anterior e o valor posterior a ação. Isso pode gerar mais de um registro por ação, desde que haja mais de um campo auditado na tabela original. Os dados adicionais são armazenados em outra tabela de controle e há um relacionamento entre ambas para poder identificar que dados pertencem à determinada ação.

---

[9] Ação de acessar um sistema computacional restrito, normalmente inserindo uma palavra de identificação e uma senha.

[10] Ação de interromper um acesso a um sistema computacional.

[11] Visão ou imagem de uma ou mais tabelas, gerada dinamicamente por uma consulta. Armazena somente a estrutura e não armazena dados.

[12] Gerador seqüencial de números inteiros.

[13] Rotinas armazenadas no banco de dados, pré-compiladas, que podem ser executadas por aplicações de usuários.

Outra diferença importante é que na Audi-DML as tabelas auxiliares são criadas com seus campos mantendo o mesmo tipo de dados da tabela original. Já no IB LogManager, os dados são armazenados em campos do tipo *varchar*[14], com exceção dos campos *blob*[15]. Uma vantagem em se manter o tipo de dado da tabela original é a possibilidade de criação de filtros por intervalos de dados no momento da criação de uma consulta aos dados auditados. A Tabela 1 apresenta as principais diferenças entre a Audi-DML e o IB LogManager.

Tabela 1. Principais diferenças entre a Audi-DML e o IB LogManager

	Audi-DML	**IB LogManager**
Bancos de dados	Oracle e Firebird	Firebird e InterBase
Gerenciamento	Conexão do administrador do banco de dados	Conexão do administrador do banco de dados ou usuário
	Cria estrutura do controle de auditoria no primeiro acesso	Cria estrutura do controle de auditoria no primeiro acesso
	Não permite ao usuário alterar os nomes das tabelas e do *role* criados para a estrutura de controle da auditoria	Permite que o usuário altere o nome dos objetos (tabelas, *views*, *stored procedures*, índices, etc.) criados para estrutura de controle da auditoria no momento do primeiro acesso
	A estrutura de controle da auditoria consiste duas tabelas e um *role*	A estrutura de controle da auditoria consiste em quatro tabelas, uma *view*, dois *generators*, duas *stored procedures e* alguns índices
	A cada tabela auditada é criada uma tabela auxiliar para armazenar os dados auditados e um *trigger* para gerar as informações	A cada tabela auditada são criados entre um e três triggers para gerar as informações, sendo elas para *insert*, *update* e *delete* (separadamente)
	Permite selecionar tabelas a auditar	Permite selecionar tabelas a auditar
	Permite selecionar as ações a auditar por tabela (*insert, update, delete*)	Permite selecionar as ações a auditar por tabela (*insert, update, delete*)
	Permite parar a auditoria de tabela	Permite parar a auditoria de tabela
	Permite alterar as ações a auditar por tabela	Permite alterar as ações a auditar por tabela
	Permite reiniciar auditoria em tabela	Permite reiniciar auditoria em tabela
	As ações a auditar dos campos são as mesmas da tabela	Permite selecionar as ações a auditar por campo (*insert, update, delete*)
	Permite selecionar campos a auditar	Permite selecionar campos a auditar
	Permite parar auditoria em campo	Permite parar auditoria em campo
	Permite reiniciar auditoria em campo	Permite reiniciar auditoria em campo
	Não permite exclusão de dados auditados	Permite exclusão de dados auditados
	Não permite alteração de dados auditados	Não permite alteração de dados auditados
	Permite concessão e retirada de permissões de consulta a auditoria para usuários do banco de dados, a qualquer momento	Permite concessão de permissões de consulta a auditoria para usuários do banco de dados no momento do registro da base de dados
	Não permite concessão de acesso a PUBLIC[16]	Permite concessão de acesso a PUBLIC
	Possui gráficos para o acompanhamento do crescimento da auditoria	Não possui acompanhamento do crescimento da auditoria
Consulta	Acesso somente a usuários com permissão de consulta a auditoria	Acesso somente a usuários com permissão de consulta a auditoria
	Permite que o usuário construa consultas aos dados auditados sem a necessidade de ter conhecimento de programação	Permite que o usuário construa consultas aos dados auditados sem a necessidade de ter conhecimento de programação
	O filtro para as consultas pode ser qualquer campo auditado da tabela selecionada mais usuário, ação, momento e local da operação, permitindo combinações e intervalos	O filtro para as consultas é composto por tabela, operação, usuário e intervalo de tempo
	Mostra as alterações feitas nos registros, destacando os campos alterados e informando qual o usuário executou a ação, que ação foi executada, o momento da ação (*timestamp*) e o local (máquina/IP) na qual a ação foi disparada	Mostra as alterações feitas nos registros, destacando os campos alterados e informando qual o usuário executou a ação, que ação foi executada, o momento da ação (*timestamp*) e o local (máquina/IP) na qual a ação foi disparada

[14] Tipo de dado de armazenamento alfanumérico de tamanho variável.

[15] Tipo de dado para armazenamento de campos muito grandes e pode conter texto ou binários.

[16] Usuário especial que representa todos os usuários do banco de dados.

A concessão de privilégios de acesso a PUBLIC não é permitida na Audi-DML, isto porque a intenção é de diminuir a probabilidade de conceder acesso por descuido a usuários indesejados, já que PUBLIC indica todos os usuários do banco de dados. Deve-se ter cautela ao conceder privilégios a PUBLIC, uma vez que todos os usuários (atuais e futuros) receberão estes privilégios.

Enquanto o IB LogManager permite a exclusão de dados auditados, a Audi-DML não permite. Ela foi concebida desta forma para reduzir a possibilidade da perda acidental de dados. Os registros auditados só podem ser excluídos por ação do administrador do banco de dados diretamente sobre as tabelas de auditoria, sem a utilização da ferramenta.

4. Experimentos e Análise de Resultados

A ferramenta Audi-DML utiliza *triggers* para inserir dados históricos em tabelas auxiliares, que são disparados automaticamente pelas ações exercidas na tabela original. Independente da ferramenta que gerou estas ações, o fato de inserir o registro na tabela auxiliar pode alterar o tempo de processamento (geralmente para cima).

Esta alteração pode não ser percebida quando for realizado um cadastramento em uma tabela básica de um determinado sistema, como o cadastramento de uma nova cidade na tabela de cidades. Porém, quando a auditoria for aplicada em uma tabela que recebe uma grande quantidade de dados envolvendo um número significativo de operações de inserção, alteração e exclusão, que são provenientes, por exemplo, de uma rotina de atualização de saldos, a alteração no tempo de processamento pode ser significativa.

Para verificar o impacto que as rotinas de auditoria podem exercer sobre uma aplicação, foram efetuados testes em tabelas nos bancos de dados Oracle e Firebird. No Oracle, os testes foram feitos para comparar os tempos de processamento entre certo número de operações em uma tabela sem a auditoria e a mesma tabela com a auditoria, gerada pela Audi-DML. No Firebird, os testes realizados foram para comparar os tempos de processamento entre uma tabela sem a auditoria e a mesma tabela com a auditoria gerada pela Audi-DML e pelo IB LogManager.

Nos testes realizados com banco de dados Oracle, foi utilizada uma máquina virtual. A máquina real possui um processador Intel Xeon Dual Quad Core com 2,83GHz, 32 GB de memória RAM, sendo que a máquina virtual aloca dois núcleos e 4 GB de memória. O sistema operacional utilizado foi o Oracle Enterprise Linux e a versão do banco de dados foi a Oracle Database 10g Enterprise Edition Release 10.2.0.4.0.

Para realizar estes testes foram criadas duas tabelas, uma para as operações a serem auditadas e outra para armazenar os tempos das execuções. A tabela a ser auditada, chamada `alunotst`, é composta de quatro campos: (i) `codaluno`, tipo *number*, tamanho 6, chave primária; (ii) `datanascimento`, tipo *date*; (iii) `nome`, tipo *varchar2*, tamanho 100; e (iv) `nomepai`, tipo *varchar2*, tamanho 60.

Para a tabela de armazenamento dos tempos de execução, chamada `tomadatempo`, foram adicionados 6 campos: (i) `tomada`, tipo *number*, tamanho 4, chave primária; (ii) `inicio`, tipo *timestamp*, precisão 6; (iii) `fim`, tipo *timestamp*, precisão 6; (iv) `operacao`, tipo *varchar2*, tamanho 20; (v) `auditado`, tipo *char*,

tamanho 1; e (vi) `tempo`, tipo *interval day to second*, com precisão 2 para *day* e 6 para *second*. As tabelas foram armazenadas em um *tablespace* e os índices em outro.

Com a auditoria desativada, foram inseridos 97.303 registros na tabela `alunotst` e os tempos gravados na tabela `tomadatempo`. Logo após, foram realizadas alterações em todos os registros inseridos, onde o valor do campo `nome` foi concatenado com valor do campo `codaluno` e o campo `datanascimento` teve seu valor somado a 2, o que acrescenta dois dias à data. Novamente os tempos foram gravados. Após as alterações, foram excluídos todos os registros da tabela `alunotst` e os tempos novamente gravados. Estes procedimentos foram repetidos cem vezes. Em seguida, a auditoria foi ativada com a ferramenta Audi-DML, e mais cem iterações foram feitas.

Em cada iteração foram inseridos três registros na tabela `tomadatempo`, totalizando 600 registros, a metade para as iterações auditadas e outra metade para as iterações não auditadas. No campo `tomada` foi armazenado o número seqüencial da tomada de tempo. Nos campos `inicio` e `fim` ficaram gravados, respectivamente, os instantes de início e fim da cada conjunto de operação. No campo `operacao` ficaram os valores *insert*, *update* ou *delete* (para que os tempos pudessem ser classificados por tipo de operação e geral). Finalmente, no campo `auditado,` foram informados os caracteres ‘S’ ou ‘N’, indicando se a iteração estava sendo auditada ou não.

Após o término de todas as iterações foi calculado o campo `tempo`, para cada um dos registros contidos na tabela `tomadatempo`. Nele, foi inserida a diferença dos valores dos campos `inicio` e `fim`, para facilitar a geração das tabelas comparativas.

A Tabela 2 apresenta o resultado das tomadas de tempo, por operações, no banco de dados Oracle. As colunas `Menor` demonstram o menor tempo decorrido, entre as cem iterações para aquela operação, para um conjunto de 97.303 registros por iteração. O mesmo serve para as colunas `Maior`, só que para o maior tempo decorrido. Já as colunas `Média`, apresentam o tempo médio levando em consideração todas as cem iterações.

Tabela 2. Tempos em segundos por operações no banco de dados Oracle

Operações	Sem auditoria			Com auditoria Audi-DML		
	Tempos em segundos (100 iterações)			Tempos em segundos (100 iterações)		
	Menor	Maior	Média	Menor	Maior	Média
INSERT	4,183225	14,243513	6,72617931	9,243304	17,435554	12,50225
UPDATE	3,646537	9,459542	5,44970007	9,122904	16,607238	12,163941
DELETE	1,083591	12,318784	3,14927521	7,231098	15,673307	10,517574
INSERT UPDATE DELETE	1,083591	14,243513	5,108385	7,231098	17,435554	11,72792

Com a verificação dos tempos colhidos nos testes, observou-se um aumento significativo no tempo de processamento em todas as operações, quando estas são auditadas pela ferramenta Audi-DML, conforme apresentado na Figura 6.

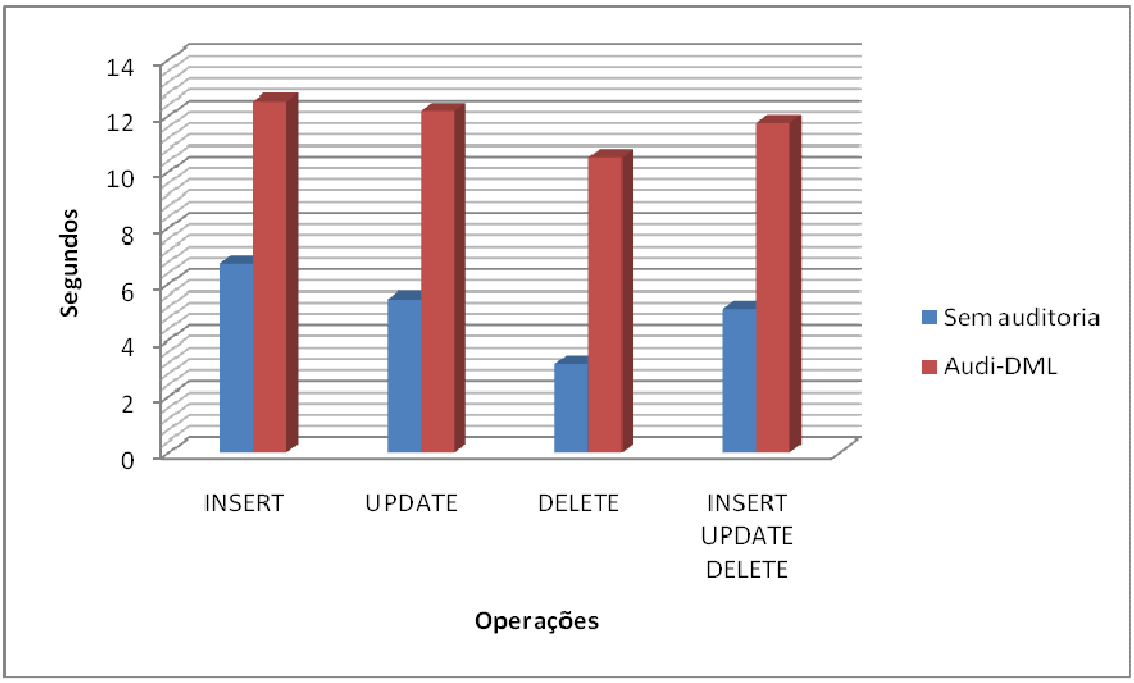

Figura 6. Gráfico das médias dos tempos de processamento no banco de dados Oracle

No banco de dados Oracle, após as cem iterações com a auditoria feita pela ferramenta Audi-DML, foram gerados 29.190.900 registros na tabela auxiliar, ocupando aproximadamente 3,2GB de espaço.

Para os testes realizados com o banco de dados Firebird também foi utilizada uma máquina virtual. A máquina real possui um processador Intel Xeon Dual Quad Core com 2,83GHz, 32 GB de memória RAM, sendo que a máquina virtual aloca um núcleo e 1 GB de memória. O sistema operacional utilizado foi o Ubuntu versão 8.04 e a versão do banco de dados foi a Firebird 2.1.

As tabelas criadas para os testes com o Firebird foram semelhantes às criadas para os testes com o banco de dados Oracle, com a mesma nomenclatura e campos de dados, diferenciando apenas os tipos de dados específicos de cada banco.

Os testes procederam de forma semelhante aos feitos com o banco de dados Oracle. Foram utilizados os mesmos registros e feitas as mesmas operações, porém foram executadas cem iterações a mais com auditoria feita pela ferramenta IB LogManager, totalizando trezentas iterações e gerando novecentos registros de tomada de tempo. Para identificar as tomadas de tempo para cada conjunto de operações, no campo `auditado` da tabela `tomadatempo`, foram informados os caracteres ‘N’, ‘S’ ou ‘I’, indicando se a iteração não estava sendo auditada, estava sendo auditada com Audi-DML, ou auditada com IB LogManager, respectivamente.

Da mesma forma que os testes anteriores, o campo `tempo`, em `tomadatempo`, foi calculado após o término de todas as iterações.

A Tabela 3 apresenta o resultado das tomadas de tempo para os testes com o banco de dados Firebird.

Tabela 3. Tempos em segundos por operações no banco de dados Firebird

Operações	Sem Auditoria			Com Auditoria Audi-DML			Com auditoria IB LogManager		
	Tempos em segundos (100 iterações)			Tempos em segundos (100 iterações)			Tempos em segundos (100 iterações)		
	Menor	Maior	Média	Menor	Maior	Média	Menor	Maior	Média
INSERT	2	15	4,95	5	84	9,727	38	58	45,71
UPDATE	2	9	4,17	3	47	7,449	29	48	33,2
DELETE	1	4	1,656	2	40	5,16	37	52	42,1
INSERT UPDATE DELETE	1	15	3,638	2	84	7,438	29	58	40,337

Semelhante aos testes feitos com o banco de dados Oracle, os testes com o Firebird também apresentaram um aumento significativo no tempo das operações, quando estas estão sendo auditadas tanto com a ferramenta Audi-DML quanto com o IB LogManager. No entanto, a auditoria gerada pela ferramenta IB LogManager ocasionou um aumento de tempo muito superior ao apresentado pela ferramenta Audi-DML, como demonstra a Figura 7.

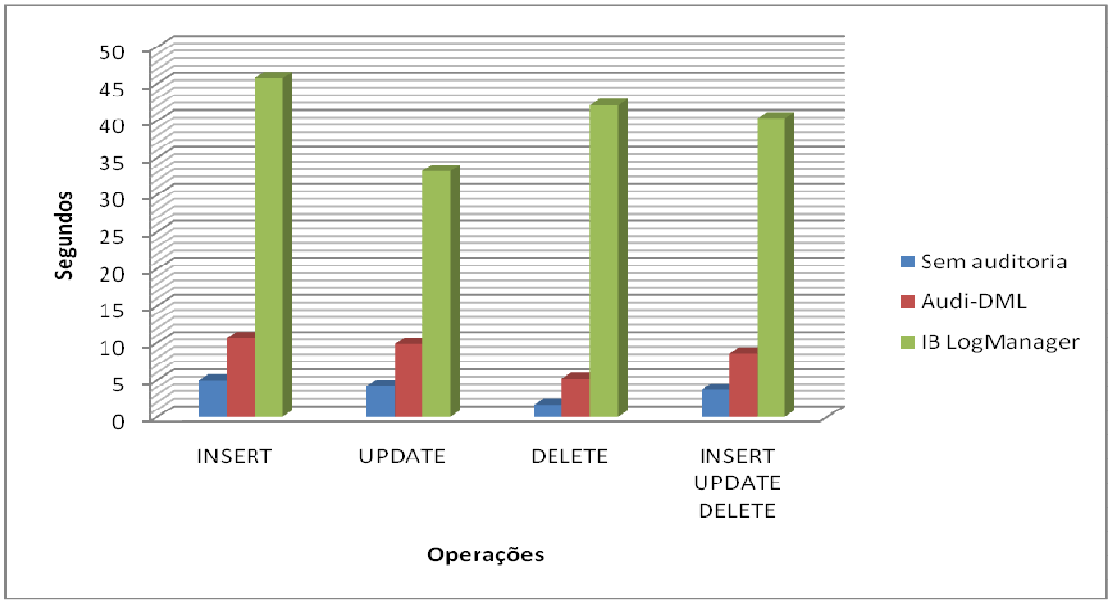

Figura 7. Gráfico das médias dos tempos de processamento no banco de dados Firebird

Com base nos testes realizados com Firebird, observou-se que a ferramenta Audi-DML apresentou melhor desempenho que a ferramenta comercial IB LogManager. Além do menor tempo de processamento, também foi observado uma importante diferença no crescimento da base de dados. Após cem iterações auditadas pela Audi-DML, foram inseridos 29.190.900 registros na tabela auxiliar. Para as mesmas cem iterações auditadas com a ferramenta IB LogManager, foram gerados 97.303.000 registros na tabela `IBLM$COLUMLOG`, onde são armazenados os dados dos campos alterados, e mais 29.190.900 registros na tabela `IBLM$OPERATIONLOG`, onde são armazenados os dados adicionais da operação, totalizando assim 126.493.900 registros inseridos na auditoria. Esta diferença é melhor visualizada na Figura 8.

Figura 8. Número de registros inseridos por ferramenta

Assim como a quantidade de registros inseridos, a diferença no crescimento da base de dados também foi considerável. Com a auditoria feita pela ferramenta Audi-DML a base de dados cresceu aproximadamente 4GB. Já com a auditoria feita pelo IB LogManager, para o mesmo conjunto de operações, o tamanho da base de dados aumentou para cerca de 17,2GB, diferença esta mostrada pela Figura 9.

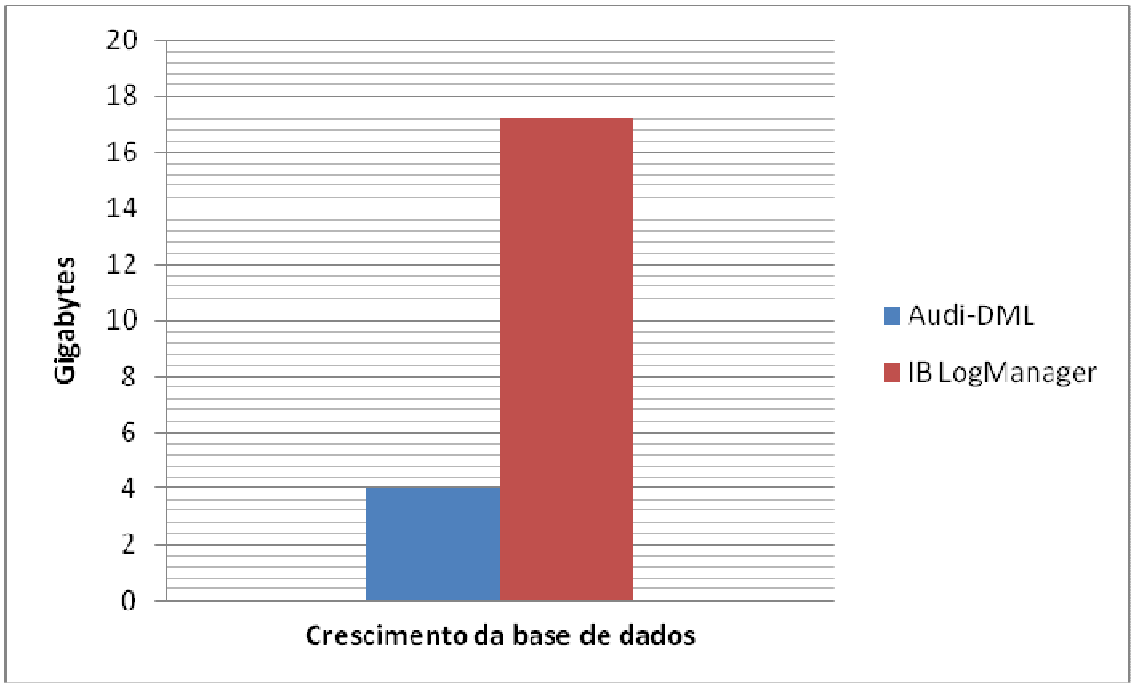

Figura 9. Crescimento da base de dados por ferramenta

5. Considerações Finais

Este artigo discutiu o problema da auditoria de dados manipulados por comandos DML em banco de dados, sua importância para a identificação da origem de erros de informação de dados e a detecção de problemas em aplicativos computacionais.

Levando-se em consideração a lacuna existente nessa área, a ferramenta Audi-DML, proposta neste trabalho, tem como finalidade a geração, o gerenciamento e a consulta de dados históricos, armazenados em tabelas auxiliares e organizados cronologicamente, com o propósito de facilitar o trabalho de auditoria nestes dados. A ferramenta foi desenvolvida para ser utilizada em bancos de dados Oracle e Firebird, e está dividida em duas partes, uma para gerenciamento e outra para consulta.

Testes foram realizados para analisar o impacto sobre o tempo de execução dos comandos DML e os resultados obtidos. Tanto no Oracle quanto no Firebird, houve um aumento significativo no tempo de processamento, quando estes comandos são executados em uma tabela que está sendo auditada.

Uma comparação entre a ferramenta Audi-DML e a ferramenta IB LogManager também foi realizada. Os resultados dos experimentos indicaram que a Audi-DML apresentou um menor acréscimo no tempo de processamento e menor crescimento da base de dados auditada.

Como sugestão de trabalhos futuros, pretende-se acrescentar novas funcionalidades na ferramenta Audi-DML, como a possibilidade de conexões a outros bancos de dados, a identificação dos campos auditados que tenham seu tipo de dado ou tamanho alterados na tabela original, permitindo que esta alteração seja passada para a tabela auxiliar, e ainda a possibilidade de salvar, nos *logs*, as sentenças SQL que geraram as alterações de dados.

Referências

Oracle. (2008) “Oracle Database Security Guide 10g Release 2 (10.2).” Oracle Corporation, Novembro 2008.

Cantu, C. H. (2009) “Auditoria em BDs Firebird.” Acesso em maio/2010. Disponível em <http://www.firebase.com.br/fb/artigo.php?id=2027>

Oliveira, D. C. M. de. (2008) “Implementando Logs de Auditoria em Banco de Dados Oracle.” Acesso em maio/2010. Disponível em <http://www.devmedia.com.br/articles/viewcomp.asp?comp=9992&hl=>

Upscene Productions. (2010) “IB LogManager Product Family – a suite of tools to keep track of data changes in Interbase and Firebird databases.” < http://www.upscene.com>, Junho 2010.