UNIVERSIDADE FEDERAL DE SANTA CATARINA

AUDILOG:
Uma ferramenta para auditoria de banco de dados

Guilherme de Mattos do Nascimento

Florianópolis - SC

2011/2
UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

AUDILOG:
Uma ferramenta para auditoria de banco de dados

Guilherme de Mattos do Nascimento

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis – SC

2011/2

Guilherme de Mattos do Nascimento

AUDILOG:
Uma ferramenta para auditoria de banco de dados

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientadora: Profª. Drª.Carina Friedrich Dorneles

Banca Examinadora:

Profª. Drª. Luciana de O. Rech
Prof. M.Sc. Angelo Augusto Frozza

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

SGBD – Sistema de Gerenciamento de Banco de Dados

DBA – *Database Administrator* (Administrador de Banco de Dados)

IDE – *Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)

SQL – *Structured Query Language* (Linguagem de Consulta Estruturada)

XML – *Extensible Markup Language*

DB – *Database* (Base ou banco de dados)

ISO – *International Organization for Standardization*

ERBD – Escola Regional de Banco de Dados

SGSD – Sistema Gerenciador de Stream de Dados

API – *Application Programming Interface*

RESUMO

O presente trabalho apresenta uma solução para auditoria de banco de dados baseada na obtenção das informações de auditoria através dos *logs* gerados pelo SGBD, nomeada DB *Audilog*. Demonstra-se no texto outras abordagens para obtenção das informações de auditoria. Também são apresentadas outras ferramentas para auditoria de banco de dados para que seja possível fazer um breve comparativo. A solução proposta é apresentada através de uma visão geral, arquitetura e forma que foi implementada. A arquitetura do programa desenvolvido foi feita para facilitar a implementação do suporte para *logs* de outros SGBDs, além do atualmente suportado que é o *PostgreSQL*.

Palavras-chave: Auditoria. Banco de Dados. Sistema Gerenciador de Banco de Dados. *PostgreSQL*. DB *Audilog*. *Log*.

1- INTRODUÇÃO

Os bancos de dados costumam armazenar importantes informações confidenciais ou de acesso restrito e que precisam ser mantidas consistentes e íntegras. Para isto, torna-se necessária a segurança dos dados, do banco e do sistema do banco de dados para que estes dados sejam acessados apenas por pessoas que tem a permissão para fazê-lo. Para garantir a segurança e prevenir problemas com acessos indevidos e integridade dos dados, pode-se utilizar a auditoria de banco de dados.

As informações para se fazer a auditoria de banco de dados podem ser obtidas de diversas maneiras, como geração de *logs* das transações, funcionalidades nativas dos SGBDs e gatilhos.

Manter *logs* do sistema também pode ser útil para outras finalidades além da auditoria, como *tunning* ou melhoria de performance das consultas e para identificar erros. Alguns dos dados que podem ser armazenados pelos *logs* gerados pela maioria dos SGBDs são:

- *login* e *logout* de usuário
- início, parada e reinício do sistema de banco de dados
- falhas e erros do sistema
- mudanças de privilégios de usuários
- mudanças na estrutura do banco
- acessos ao banco através de operações como SELECT ou INSERT.

Os bancos de dados comerciais costumam ter funcionalidades implementadas que auxiliam na auditoria. Os bancos da *Oracle* possuem algumas soluções do tipo, como, por exemplo, o comando audit(ORACLE, 2010). O DB2 da IBM possui um

mecanismo de auditoria que gera e permite que sejam mantidas informações para auditoria de eventos previamente definidos(IBM, 2011). Estas informações são armazenadas em um arquivo de *log*.

É importante destacar que em todos estes casos, o administrador de banco de dados, também chamado database administrator, precisa executar comandos para que se possa descobrir o que foi armazenado pelas ferramentas de auditoria. Esta se torna uma tarefa trabalhosa, pois deve ser executada manualmente quando não há ferramentas adequadas para auxiliar no processo.

As funcionalidades e ferramentas descritas acima são características existentes em SGBDs proprietários. Os SGBDs *free*(gratuitos e de código aberto), como o *PostgreSQL* e o *MySQL*, não possuem ferramentas de auditoria nativas ou gratuitas de grande expressão.

Tendo em vista a carência apontada nos SGBDS *free e* pela escassez de ferramentas de auditoria de banco de dados gratuitas, o objetivo do trabalho é apresentar uma solução para auditoria de banco de dados baseada nos *logs* gerados pelos SGBDs e que permita fazer uma análise a partir dos dados coletados, gerando gráficos que possam ajudar o DBA a monitorar o banco. O intuito principal é permitir a auditoria para melhoria de performance das consultas, mas também aliando aspectos da segurança e erros através da análise das consultas SQL.

A estrutura do texto está organizada da forma descrita a seguir. No capítulo 2 é apresentada a auditoria de banco de dados, destacando a sua utilidade e as principais abordagens para gerar dados para se fazer a auditoria. O capítulo 3 apresenta algumas ferramentas de auditoria existentes. O capítulo 4 apresenta o *Audilog*, a ferramenta proposta neste trabalho que se baseia na auditoria pelos *logs* do SGBD. Por fim, o capítulo 5 apresenta a conclusão e trabalhos futuros.

2. AUDITORIA DE BANCO DE DADOS

Auditoria é um processo sistemático, documentado e independente para obter informações para avaliar objetivamente se um conjunto de políticas, procedimentos ou requisitos são atendidos, dependendo do escopo da auditoria. Este processo é importante para que a organização possa analisar se as atividades desenvolvidas estão sendo executadas como planejadas e se os seus objetivos são alcançados de forma satisfatória (ISO 19011, 2002, p. 01).

2.1 O QUE É A AUDITORIA DE BANCO DE DADOS E PARA QUE SERVE

A auditoria de banco de dados procura dar respostas a perguntas como: Quem acessou ou alterou um dado, quando e como? Para isso, os SGBDs costumam prover funcionalidades para criar registros de auditoria para transações no banco, alterações na suas estruturas e eventos de sistema.

A auditoria de banco de dados é feita para evitar e identificar ações indevidas por parte do usuário através de seus acessos aos dados. Informações sobre os acessos da base de dados são coletadas e analisadas para que se possa descobrir falhas de segurança e a origem do problema. Esta auditoria deve ser feita de forma independente e transparente, catalogando todas as informações relevantes (SIMON *et al.,* 2011, p. 02).

Mullins (2002, p. 392) diz que a auditoria é uma funcionalidade do SGBD que permite que o DBA mantenha registro do uso dos recursos e privilégios do banco de dados. Quando a auditoria está habilitada, o SGBD mantém um registro de auditoria das operações do banco de dados. Cada operação auditada produz informações que incluem qual objeto foi afetado, quem fez a operação e quando. Dependendo do

nível de auditoria suportado pelo SGBD, um registro do que foi alterado também pode ser feito. A auditoria é importante para segurança da informação, mas também serve para melhoria de performance do banco, detecção de falhas de consultas e de sistema.

Um dos grandes problemas da auditoria é que as informações de auditoria geradas podem causar um impacto na performance do sistema e fazer com que uma grande quantidade de dados seja armazenada. Por isto, deve-se definir um nível de granularidade da auditoria para que não sejam armazenadas informações desnecessárias. Dependendo deste nível de granularidade, as informações de auditoria podem ser utilizadas para recuperação de dados através dos valores anteriores e posteriores a uma alteração nos dados.

Entre as informações de auditoria que costumam ser coletadas incluem: *login* e *logoff*, reinício do servidor de banco de dados, comandos feitos por usuários com privilégios de administrador de sistema, tentativas de violação de integridade de dados, operações de inserção, alteração, deleção e seleção de dados, execução de *stored procedures*, tentativas de acessos ao banco de dados não permitidos, mudanças na estrutura das tabelas, alterações em um registro (MULLINS, 2002, p. 393).

2.2 COMO IMPLEMENTAR A AUDITORIA

Há diversas maneiras de se obter as informações para auditoria do banco de dados. As seções a seguir apresentam algumas delas.

2.2.1 Ferramentas de auditoria nativas

A auditoria de banco de dados pode ser feita através de ferramentas nativas do

SGBD.

Segundo Chopskie (2011, p. 03), os Sistemas de Gerenciamento de Banco de Dados como *Oracle*, *Sybase*, *Microsoft* SQL *Server*, IBM DB2 e IBM IMS contêm a funcionalidade de criar registros de auditoria das transações que acessam os dados armazenados nos seus bancos de dados. Cada um deles permite um nível diferente de granularidade do que pode ser auditado.

As ferramentas nativas causam um impacto na performance do SGBD se algo que acontece frequentemente é registrado na auditoria.

2.2.2 Triggers de auditoria

A auditoria com *triggers* é feita com a criação de gatilhos que são disparadas a cada alteração nos dados das tabelas através dos comandos INSERT, UPDATE ou DELETE. Geralmente, os dados armazenados pelas *triggers* são os valores da linha alterada, antes e depois da alteração, em uma tabela que espelha a original e mantém dados adicionais como data de atualização, usuário que executou ou tipo de operação. Tendo as informações armazenadas pelos gatilhos, pode-se fazer consultas para recuperar os dados, saber das operações de cada usuário, entre outros, dependendo das informações armazenadas pelos gatilhos. A utilização de *triggers* para auditoria causa um impacto levemente significativo na performance do banco, pois cada operação de INSERT, UPDATE, DELETE acarreta em ao menos uma operação de inserção nas tabelas de espelho.

2.2.3 Camada de Rede: Packet Sniffing

Algumas soluções *third-party* são baseadas no monitoramento dos pacotes que trafegam na camada de rede, vindos das conexões com os clientes para o SGBD.

Este tipo de solução não contempla as operações feitas localmente, pois elas não passam pela camada de rede. Além disso, vários outros problemas técnicos podem dificultar a auditoria através de *packet sniffing*, como a compatibilidade da solução com o dispositivo de rede ou capacidade de inspecionar pacotes encriptados (CHOPSKIE, 2011, p. 06-07).

2.2.4 Ferramentas Third-Party baseadas em agentes

As ferramentas *third-party* podem utilizar agentes que tentam acessar a memória compartilhada do SGBD para adquirir as operações SQL executadas. Este tipo de abordagem consome aproximadamente 5% do uso da CPU segundo os desenvolvedores. Estas ferramentas podem conter funcionalidades que as ferramentas nativas não apresentam, como alertas em tempo-real para que ações suspeitas sejam detectadas e tratadas pelos administradores de bancos de dados mais rapidamente (CHOPSKIE, 2011, p. 08-09).

Algumas ferramentas também mesclam os agentes com a abordagem de *packet-sniffing*.

2.2.5 Logs do SGBD

Os SGBDs permitem que sejam armazenadas informações das ações tomadas no sistema em arquivos de *log,* como eventos de sistema, transações feitas no banco e mudanças de privilégios. Estes arquivos podem ser utilizados para se fazer a auditoria. Para isto, as ferramentas que utilizam os *logs* do próprio SGBD para coletar as informações de auditoria, lêem o arquivo fazendo uma análise do que se trata cada parte do *log* para, então, poder utilizá-las em auditorias.

Com a utilização de *logs*, pode-se fazer a auditoria fora do servidor do banco

de dados, ao contrário de auditorias nativas ou *triggers*. Para ser feito este tipo de auditoria, costuma-se utilizar um *software* para análise dos *logs* gerados. Os *logs* devem estar configurados de acordo com as necessidades da auditoria e do *software*.

Um ponto fraco deste tipo de auditoria é o fato de que algumas informações não são registradas em *log* por certos SGBDs. Além disso, este tipo de análise não costuma dar suporte para alertas em tempo real. O impacto na performance acontece apenas pela geração dos arquivos de *log* que podem tomar muito espaço em disco a longo prazo, pois a análise pode ocorrer fora do servidor.

2.3 GERENCIAMENTO DE LOG DE BANCO DE DADOS

O uso de *logs* para manter registros das ações tomadas em um banco de dados e a análise destes *logs*, permite que seja feita uma auditoria neste banco e se garanta que os requisitos de segurança do banco de dados sejam cumpridos. Os principais requisitos de segurança em banco de dados geralmente contêm os seguintes componentes, segundo CHUVAKIN (2011, p. 01):

- controle de acesso ao *software*, estruturas e dados do banco de dados.
- configuração adequada do banco
- criptografia dos dados
- verificação de vulnerabilidades.

Diferentes SGBDs oferecem opções próprias para configuração de *log*. A maioria pode registrar todos, ou quase todos eventos a seguir, citados por Chuvakin (2011, p. 02):

- *logins* e *logouts* de usuários

- início e paradas do SGBD
- falhas e erros do sistema
- mudanças de privilégios
- mudanças nas estruturas do banco de dados
- a maior parte das ações de DBA
- acessos aos dados.

Estes registros feitos nos *logs* são importantes fontes de informação para auditoria e cumprimento dos requisitos de segurança em banco de dados. Porém, estes registros não costumam ser fáceis de serem analisados por diversos fatores, como disponibilidade do *log*, complexidade do formato de log e volume de dados (CHUVAKIN, 2011, p. 02). Além disso, cada SGBD possui seu próprio formato de *log*. É importante ressaltar que a maioria dos registros não estão nas configurações padrões dos SGBDs.

Ferramentas de análise e gerenciamento de *log* se tornam essenciais para que a auditoria possa ser feita a partir dos *logs*. A maioria das ferramentas permite que os *logs* sejam revistos em um SGBD específico apenas e requer que seja rodada no mesmo servidor do SGBD. Elas também não oferecem nenhum tipo de análise em tempo real através de alertas ou ações de resposta automática (CHUVAKIN, 2011, p. 05).

Algumas ferramentas mais avançadas permitem que sejam executadas em diferentes servidores de bancos de dados. Algumas permitem, até mesmo, uma análise dos logs do banco de dados em conjunto com outros *logs*, como de *firewall* (CHUVAKIN, 2011, p. 05-06). Estas ferramentas não automatizam somente a análise

dos dados do *log*, mas todo ciclo de vida do gerenciamento de *log*, que envolve tarefas como:

- Coletar o *log* remotamente ou através de um agente no local em que é gerado.
- Transferir e manter os dados de forma segura
- Disparar alertas aos DBAs
- Prover relatórios e análises
- Armazenar e apagar os *logs* de acordo com as regras de retenção

A escolha entre uma ferramenta mais simples de análise de *log* ou um mecanismo de gerenciamento de *log* depende de fatores como recursos financeiros e de TI, tamanho da organização e regras que precisam ser seguidas. Outro problema que precisa ser considerado é a performance. Um agente operando no servidor do banco de dados pode tornar o sistema lento durante a própria ação ou por razões como *bugs* ou *memory leaks*. A coleta remota dos *logs* tem menos chances de afetar a performance do banco de dados, mas a coleta inicial tem potencial para isto devido a grande quantidade de dados. Para evitar isto, é interessante agendar as coletas para horários de baixo acesso (CHUVAKIN, 2011, p. 08).

3– TRABALHOS RELACIONADOS

Neste capítulo é apresentada uma proposta de arquitetura para auditoria de banco de dados baseada em registros de *log* de autoria de Fernando Simon, Aldri L. dos Santos e Carmen S. Hara, apresentado na Escola Regional de Banco de Dados com título “Um Sistema de Auditoria baseado na Análise de Registros de Log”. (SIMON et al., 2008) Além disso, são apresentadas outras ferramentas para auditoria de banco de dados e um breve comparativo entre elas e a ferramenta proposta neste trabalho.

3.1 – SISTEMA DE AUDITORIA BASEADO EM SGSD PARA ANÁLISE DE LOGS DO SGBD

Sistemas Gerenciadores de *Streams* de Dados (SGSD) são sistemas que controlam a manutenção e a requisição de fluxos de dados. Os SGSD costumam executar as mesmas consultas, definidas previamente, sobre os dados que chegam ao longo do tempo, ao contrário dos SGBDs que utilizam consultas dinâmicas. (SIMON et al., 2008, p. 01-02)

No trabalho descrito nesta seção, propõe-se uma integração entre SGBD e SGSD para realizar a auditoria dos dados através dos registros de log do SGBD. Para isto, foram escolhidos o SGBD *PostgreSQL* e o SGSD *Borealis* (SIMON *et al.*, 2008, p. 01).

A arquitetura do sistema proposto pode ser visualizada na Figura 1.

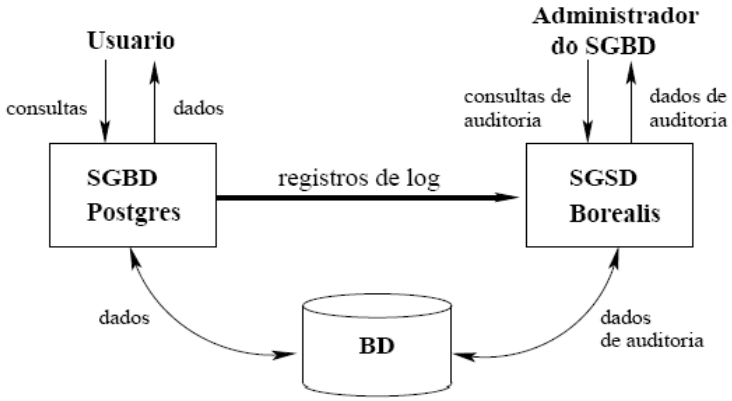

Figura 1: Arquitetura do Sistema de Auditoria (Fonte: SIMON et al., 2008. p. 05)

O SGSD *Borealis* permite ser executado em uma máquina, enquanto um SGBD que serve de fonte dos dados é executado em outra. Cada operação feita no SGBD gera um registro de *log*, que é enviado para o SGSD. Então, esse fluxo de registros é processado pelo SGSD para que se possa obter informações através das consultas previamente definidas pelo DBA. Estas informações também podem ser obtidas em tempo real. Esta arquitetura permite flexibilidade e independência. Ela permite a filtragem dos registros de *log* através das consultas predefinidas e a auditoria é feita por um aplicativo externo ao SGBD, que será executado em outra máquina (SIMON *et al.*, 2008, p. 04-06).

Para utilização do SGSD é necessário definir o esquema dos registros de entrada e saída, as consultas a serem aplicadas aos fluxos de dados e a codificação de um programa para transformar o fluxo de entrada no esquema definido. A definição dos esquemas dos registros de entrada e saída é feita em XML (SIMON *et al.*, 2008, p. 06).

Concluindo o artigo, os autores dizem que o SGDS *Borealis* oferece uma linguagem de alto nível para definição de regras de auditoria através de consultas e que, alterações nas regras e políticas de auditoria requerem apenas alterações nas

consultas registrada neste sistema. Porém, necessita-se de mais estudos de casos para descobrir se a linguagem oferecida pelo SGDS *Borealis* é adequada para expressar as consultas que geralmente são empregadas em auditoria (SIMON *et al.*, 2008, p. 08).

3.2 – FERRAMENTAS E MECANISMOS DE AUDITORIA

Este tópico apresenta alguns mecanismos de auditoria de banco de dados para SGBDs específicos.

3.2.1 – Comando audit no Oracle Database

O comando *audit* é usado para criar informações específicas de auditoria (*audit trail*). Estas informações são armazenadas em tabelas específicas ou arquivos XML de acordo com a configuração, além de outras opções.

Para habilitar o comando *audit* no *Oracle Database* é preciso mudar o parâmetro de configuração de inicialização "*audit_trail*" para "*true*" para armazenar os dados em tabelas no banco de dados ou "xml" para armazená-los em arquivos no formato XML. Com o comando *audit* é possível auditar acessos a tabelas ou atividades de usuários específicos. O *audit* permite armazenar informações sobre operações de acessos às tabelas (INSERT, UPDATE, DELETE, SELECT), alterações das estruturas do banco (CREATE, ALTER, DROP de objetos como tabelas, colunas, usuários, mudanças de privilégios, etc.) e eventos do sistema.

O exemplo de comando a seguir permite auditar todas as alterações de dados feitas pelo usuário JOHN no *Oracle Database*:

```
audit update table, delete table, insert table by JOHN by access;
```

Com as informações armazenadas, pode-se fazer a auditoria através de

consultas SQL. Porém, para um detalhamento maior das consultas e alterações nas linhas das tabelas torna-se necessária a utilização de *triggers*.

3.2.2 – pgFouine para PostgreSQL

O *pgFouine* é um analisador de *log* para *PostgreSQL* utilizado para gerar relatórios. Ele está sob a licença para distribuição GNU e tem suporte apenas para sistema operacional *Linux*. Os relatórios podem ser úteis para determinar quais consultas precisam ser otimizadas para melhorar a performance do banco de dados. O *pgFouine* fornece estatísticas sobre duração das consultas, quais consultas são as mais frequentes, histórico das consultas executadas e relatórios de erros com maior ocorrência. O *pgFouine* exige que o log do *PostgreSQL* esteja configurado de maneira a registrar as consultas de acordo com a duração de suas execuções e precisa ser executado em linha de comando. Também deve ser definido o prefixo da linha de *log* para mostrar o usuário e o banco da consulta caso se deseje fazer relatórios filtrados por estes dados. O resultado da execução do programa para geração de relatório pode ser obtido em arquivo de texto ou em formato HTML construído com tabelas, o que facilita a visualização das informações.

O comando a seguir pode ser utilizado para criação de um relatório em que apenas consultas do tipo SELECT devem ser consideradas.

```
pgfouine.php -file logs_pgbench_20051211063633.log -onlyselect
```

3.2.3 Auditoria com triggers ou gatilhos

A auditoria com *triggers* é feita com a criação de gatilhos que são disparadas a cada alteração nos dados das tabelas através dos comandos INSERT, UPDATE ou DELETE. Geralmente, os dados armazenados pelas *triggers* são os valores da linha

alterada, antes e depois da alteração, em uma tabela que espelha a original e mantém dados adicionais como data de atualização, usuário que executou, tipo de operação. Tendo os dados armazenados pelos gatilhos, pode-se fazer consultas para recuperar os dados, saber as operações de cada usuário, entre outros, dependendo das informações armazenadas pelos gatilhos. As auditoria através das triggers é limitada pois não pode ser feita para consultas do tipo SELECT, alterações nos metadados do banco e nem para eventos do sistema de banco de dados, o que a torna bastante específica.

3.3 COMPARATIVO

A tabela abaixo mostra um comparativo das técnicas apresentadas.

Tabela 1: Comparativo das técnicas apresentadas.

	Audilog	***pgFouine***	***audit* (*Oracle*)**	***Triggers***
Consultas mais frequentes	Sim	Sim	Não	Não
Consultas mais demoradas	Sim	Sim	Não	Não
Lista de Consultas	Sim	Sim	Não	Não
Tipo de Análise	Leitura de XML gerado a partir dos logs	Leitura de log.	Leitura das informações geradas pelo SGBD e armazenadas no banco ou em XML através de consultas	Consulta nas tabelas com dados gerados pelas *triggers*.
Interface de Execução	Sim	linha de comando	consulta no SGBD	consulta no SGBD
Interface Gráfica de Saída	Gráficos e tabelas em j*ava.swing*.	tabelas em HTML	SGBD	SGBD
Conteúdo da Análise	DML	DML	DDL, DML, Eventos de Sistema	*insert*, *update*, *delete*.
SGBD	*PostgreSQL*	*PostgreSQL*	*Oracle*	vários

Além das comparações apresentadas na Tabela 1, é importante destacar que as análises feitas a partir de arquivos de *log* podem ser feitas fora do servidor do SGBD. O comando *audit* do *Oracle* armazena uma variedade de informações de auditoria maior, mas menos detalhada em relação as triggers e análises de *log* para auditoria em consultas DML.

A auditoria por *trigger* fica limitada pelos seus próprios disparos que são feitos apenas em *inserts*, *updates* e *deletes*. Além disso, as *triggers* e tabelas que armazenam as informações de auditoria podem precisar serem alteradas quando há uma alteração na estrutura da tabela que a utiliza. A vantagem da utilização da trigger é que ela permite guardar um histórico dos valores das linhas da tabela através do armazenamento dos parâmetros das consultas.

O DB *Audilog* e o *pgFouine* permitem uma auditoria direcionada a melhoria da performance das consultas, pois demonstram as consultas mais utilizadas e mais lentas de forma bastante fácil de visualizar. Porém, o DB *Audilog* facilita o seu uso através de uma interface gráfica para execução da auditoria e demonstração de gráficos estatísticos. Além disso, o DB *Audilog* foi implementado com uma arquitetura que permite a implementação da análise para outros SGBDs e sua análise é feita pelos arquivos XML, o que o torna mais versátil para futuras implementações para suporte a novos tipos de consultas também. Uma desvantagem da auditoria através dos logs é que a recuperação de dados pode ser feita apenas através da análise das consultas que fizeram as alterações, ao contrário das triggers que podem armazenar diretamente os valores.

4 – DB AUDILOG

Neste capítulo é apresentado o DB *Audilog* e suas funcionalidades como proposta de programa para auditoria de acessos a banco de dados.

4.1 VISÃO GERAL

O DB *Audilog* é um programa para auxílio na auditoria de banco de dados. O programa apresenta gráficos para tabelas mais acessadas e acessos por usuários de acordo com filtros pré-configurados, além de mostrar as consultas em forma de relatório, destacando as mais lentas e as mais frequentes. As informações de acesso ao banco de dados pelos usuários são adquiridas através dos *logs* gerados pelo próprio SGBD. Nesta primeira versão do DB *Audilog*, apenas os *logs* gerados pelo *PostgreSQL* são suportados.

A Figura 2 demonstra a relação do DB *Audilog* com o SGBD.

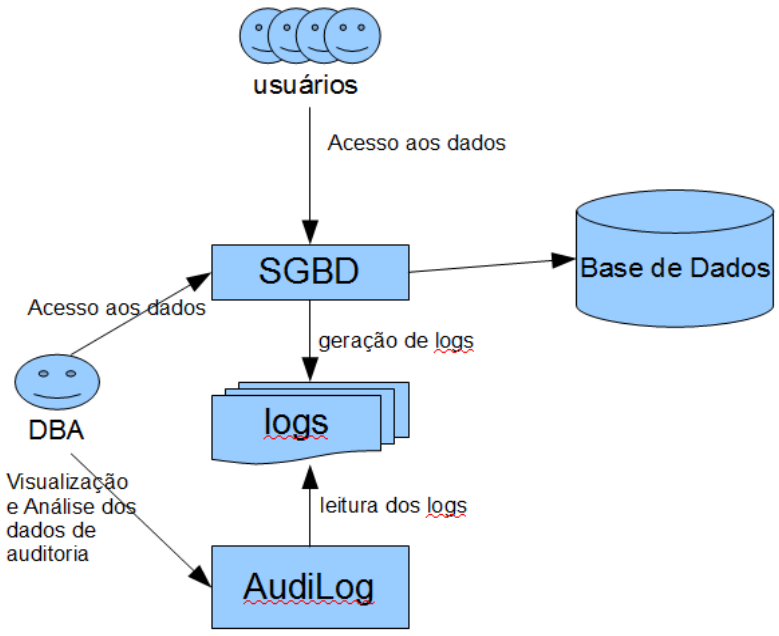

Figura 2: DB Audilog e o SGBD

Para utilização do *Audilog* é necessário que o *log* do SGBD esteja configurado

adequadamente para que os dados utilizados por aquele programa sejam coletados. O arquivo de configuração de *log* do *PostgreSQL* é chamado *postgresql.conf*. Os seguintes atributos devem estar configurados:

log_statement = 'all' : define qual tipo de comando SQL deve ser armazenado. O valor *'all'* denota que todos os tipos devem ser armazenados.

log_line_prefix = '%t user=%u dbname=%d ' : define se algumas informações devem vir antes da linha de *log*. O valor *'%t'* é para *timestamp*, ou seja, data e hora. Já o valor *'%u'* é para usuário. E, por fim, *'%d'* para o nome do banco de dados.

log_duration = 'on' : habilita o armazenamento do tempo de execução da consulta

datestyle = 'iso, dmy' : define o formato da data.

As informações necessárias para auditoria só são armazenadas a partir do momento que as configurações de *log* do SGBD estiverem corretas.

O *Audilog* apresenta funções para configuração de conexão com o banco de dados, abrir, gerar e salvar relatórios. A conexão com o banco é feita para que se possa obter os metadados do banco e, assim, conhecer a sua estrutura e os usuários. Desta forma, torna-se possível filtrar a análise considerando apenas as consultas que acessam ou alteram uma tabela especificada e, também, filtrar a análise por usuário do SGBD. Caso contrário, todas consultas e todos os usuários são consideradas nas análises.

Figura 3: Janela de Configuração de Conexão com o SGBD

A configuração da conexão com o banco de dados é opcional e pode ser observada na Figura 3. Porém, sem a conexão não se pode obter a estrutura dos metadados nem obter a lista de usuários do SGBD. Esta configuração é feita através do item *Conexão* presente no menu *Configuração*. Para fazer a conexão é necessário escolher o SGBD (no momento apenas o *PostgreSQL* é suportado) e inserir nome do usuário do SGBD, senha do usuário, endereço IP do servidor do banco de dados, número da porta utilizada pelo SGBD e nome do banco de dados.

É importante ressaltar que o *PostgreSQL* por padrão, não permite conexão remota com o banco de dados. Para que isto seja possível, é necessário adicionar o IP da máquina cliente (a que está executando o DB *Audilog*) ao arquivo *pg_hba.conf* do *PostgreSQL*.

Figura 4: Componente de visualização de metadados

Tendo os metadados, o *Audilog* carrega e apresenta os metadados em forma de árvore para que os relatórios possam ter restrições por tabela, ou seja, gerar relatório para acessos a uma tabela específica. A Figura 4 apresenta os metadados carregados.

Para que seja feita a análise dos *logs*, é obrigatório definir em qual pasta estão localizados os arquivos de *log* do SGBD para que o programa faça a leitura e armazene dados para auditoria do banco. A Figura 5 demonstra a janela de configuração e leitura dos *logs* e deve ser acessada através do item *Leitura de Log* no menu *Configuração*.

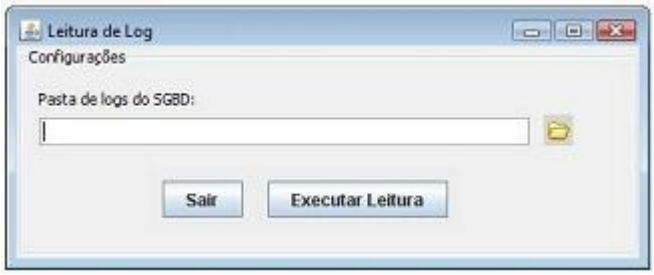

Figura 5: Janela de Leitura de Log

Deve-se selecionar a pasta em que os *logs* a serem lidos se encontram e clicar no botão *Executar Leitura* para que os *logs* sejam lidos, as informações sejam extraídas e armazenadas em um arquivo XML de forma a serem mais facilmente acessadas. Além de facilitar a análise pela estrutura do armazenamento das informações no arquivo XML, guardá-las neste formato permite que a análise seja independente do *log* que foi utilizado para obter as informações para auditoria. Sendo assim, torna-se possível que a análise para auditoria e a leitura do *log* sejam implementadas independentemente uma da outra. Isto faz com que a implementação da leitura e extração de informações de um novo formato de *log* seja suficiente para dar suporte a auditoria para o SGBD deste novo formato.

Figura 6: Painel de Filtro e Execução.

Para realização da auditoria o usuário deve clicar no botão *Auditar* que se encontra no canto esquerdo da tela principal. A análise pode ser restrita por um intervalo de tempo, por usuário, por tabela e por tipos de operação. A Figura 6 mostra o painel de filtro da auditoria e de execução da mesma. Este painel se encontra no canto esquerdo da tela principal.

A saída da auditoria é apresentada através de um gráfico de barras que apresentam as tabelas mais acessadas de acordo com a auditoria e um gráfico de pizza que apresenta os acessos por usuários. A Figura 7 apresenta a tela principal do DB *Audilog* mostrando os gráficos gerados. Quando a filtragem da análise é feita por alterações e selecionando uma tabela específica, também é mostrado um gráfico que apresenta as alterações por coluna da tabela.

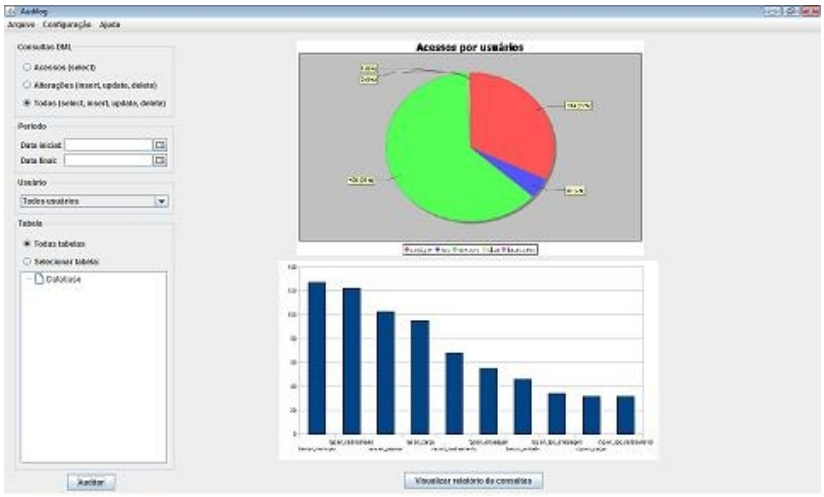

Figura 7: Tela Principal com Gráficos Gerados

Pode-se também obter um relatório com as consultas utilizadas na auditoria clicando no botão *Visualizar Relatório de Consultas* abaixo dos gráficos. Este relatório apresenta as consultas de três formas: ordenadas por data de execução, apresentando as mais lentas e, por fim, destacando as consultas mais frequentes. O relatório por data de execução e o relatório por consultas mais lentas apresentam as consultas informando a data da execução, o tempo ou duração de execução da consulta, o usuário que fez a consulta e a própria consulta. Já o relatório por frequência das consultas é feito desconsiderando os valores das comparações das consultas, montando uma consulta genérica. Um exemplo pode ser visualizado a seguir, em que as duas primeiras consultas resultam na mesma consulta genérica.

Consulta 1	select * from bas.en_pessoa where id_pessoa=2
Consulta 2	select * from bas.en_pessoa where id_pessoa=5
Consulta genérica	select * from bas.en_pessoa where id_pessoa=?

Este tipo de relatório apresenta a frequência ou número de vezes que a consulta foi executada, a duração média da consulta e a consulta genérica.

Os relatórios podem ser visualizados na Figura 8.

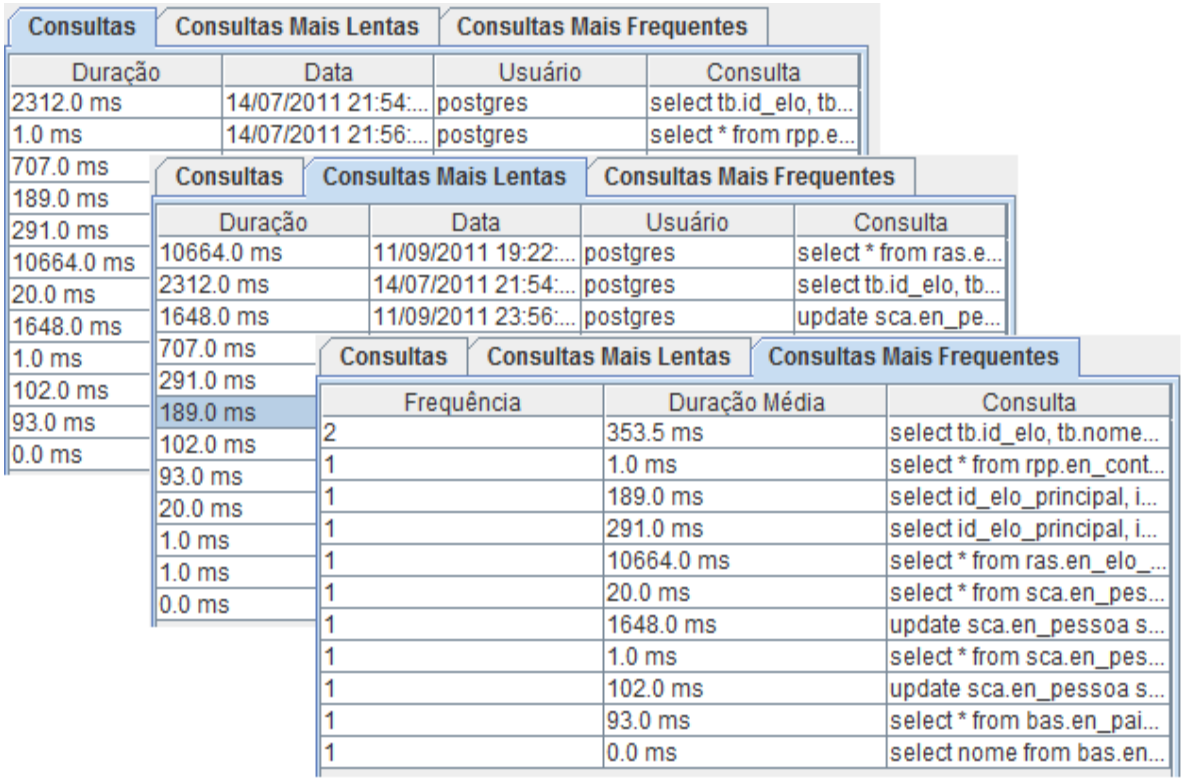

Figura 8: Relatórios de consultas

4.2 ARQUITETURA E IMPLEMENTAÇÃO

A implementação do *Audilog* foi feita na linguagem de programação *Java* através da IDE *Eclipse* e a versão atual dá suporte para o *PostgreSQL*. No entanto, outros tipos de *log* e, consequentemente, outros tipos de SGBD poderão ser suportados futuramente através da reimplementação de interfaces que lidam com a leitura do *log*.

Ao configurar uma conexão, os metadados são carregados pelo programa através de uma conexão com o JDBC e armazenados em um arquivo XML. A estrutura dos metadados é apresentada no programa no painel inferior esquerdo. Após a criação do arquivo de XML, os *logs* do SGBD podem ser lidos, sendo extraídas as informações utilizadas na auditoria e armazenadas em formato XML, também para posterior análise e auditoria.

O *Document Type Definition* (DTD) da Tabela 2 apresenta a estrutura do arquivo XML que armazena os metadados do banco de dados.

De acordo com o DTD, o XML apresenta a estrutura do banco de dados em forma de árvore, com esquemas contendo tabelas e tabelas contendo colunas. Cada item da estrutura tem também seu nome armazenado como atributo.

Tabela 2: metadata.dtd

```
<!DOCTYPE metadata [
<!ELEMENT database (schemas)>
<!ELEMENT schemas ((schema+))>
<!ELEMENT schema (tables)>
<!ELEMENT tables ((table+))>
<!ELEMENT table (columns+)>
<!ELEMENT columns (column+)>
<!ELEMENT column EMPTY>

<!ATTLIST schema name CDATA #REQUIRED>
<!ATTLIST table name CDATA #REQUIRED>
<!ATTLIST column name CDATA #REQUIRED>
]>
```

Os usuários acessam os dados através do SGBD. Cada acesso gera um registro de *log* armazenado em um arquivo de *log*. O *Audilog* faz uma leitura destes arquivos de *logs* para armazenas as informações dos acessos em um arquivo XML que pode ser usado para analisar cada comando e gerar relatórios e gráficos para auxílio na auditoria do banco de dados.

O próximo DTD define a estrutura do XML que armazena as informações das consultas obtidas através da leitura dos arquivos de *log*:

Tabela 3: queries.dtd

```
<!DOCTYPE queries [
<!ELEMENT queries (instruction*)>
<!ELEMENT instruction ((sql))>
<!ELEMENT sql ((command,tabelas))>
<!ELEMENT command (#PCDATA)>
<!ELEMENT tabelas ((accessed,altered))>
<!ELEMENT accessed (table*)>
<!ELEMENT table EMPTY>
<!ELEMENT altered (table*)>
<!ELEMENT table (column)>
<!ELEMENT column EMPTY>

<!ATTLIST instruction date CDATA #REQUIRED>
<!ATTLIST instruction user CDATA #REQUIRED>
<!ATTLIST sql operation CDATA #REQUIRED>
<!ATTLIST sql duration CDATA #REQUIRED>
<!ATTLIST table name CDATA #REQUIRED>
<!ATTLIST table schema CDATA #REQUIRED>
<!ATTLIST column name CDATA #REQUIRED>
]>
```

O XML das consultas apresenta a instrução ou comando que gerou a informação, a data da execução do comando, o usuário que o executou, o tipo de operação (até o momento apenas consultas DML: SELECT, INSERT, UPDATE,

DELETE), a duração da consulta, as tabelas acessadas com seus nomes e esquemas, as tabelas alteradas com nome e esquema e, por fim, as colunas alteradas.

A Figura 9 apresenta um fluxograma que representa a leitura do arquivo de *log*. De acordo com a figura, o *Audilog* faz a leitura do arquivo de *log* armazenando as informações para posterior análise. A leitura é feita registro por registro. Se o registro de *log* for um comando, informações sobre o comando são adquiridas e armazenadas em arquivos no formato XML.

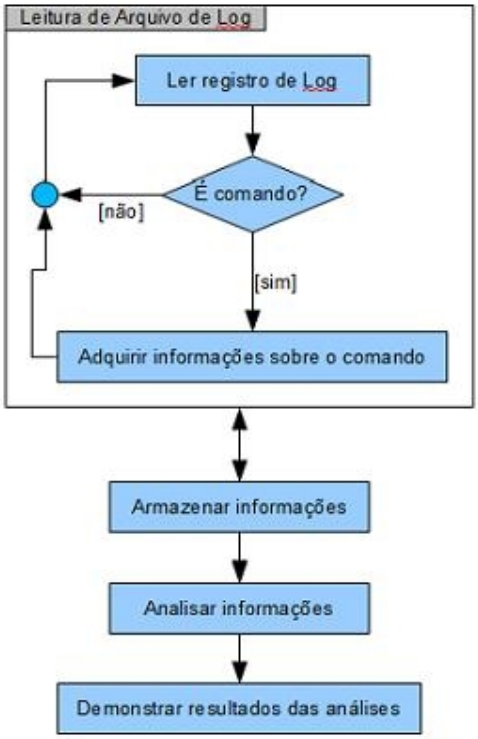

Figura 9: Fluxograma que descreve o funcionamento do DB Audilog

Tabela 4: Exemplo de registro de acesso ao banco de dados encontrado no log do PostgreSQL.

2010-11-01 14:40:03 BRT postgres basename LOG: comando: select * from rpp.en_rastreamento rpp inner join rpp.en_carga car on rpp.id_rastreamento=car.id_rastreamento where rpp.id_rastreamento=979035 2010-11-01 14:40:03 BRT postgres basename LOG: duraÃ§Ã£o: 9.000 ms

No exemplo de registro de acesso da Tabela 4, as seguintes informações são obtidas:

Data e hora: 2010-11-01 14:40:03

usuário: postgres

nome do banco de dados: basename

consulta: select * from rpp.en_rastreamento rpp inner join rpp.en_carga car on rpp.id_rastreamento=car.id_rastreamento where rpp.id_rastreamento=979035

operação: select

tabelas acessadas: rpp.en_rastreamento, rpp.en_carga

duração: 9.000 ms

Os dados coletados do arquivo de *log* são armazenados num arquivo XML para tornar mais fácil a implementação de suporte a outros formatos de *logs* gerados por outros SGBDs e facilitar as análises. Desta forma, apenas o módulo de leitura de *log* para o novo tipo de *log* precisa ser implementado, sem precisar reimplementar as análises feitas com os dados.

Quando o DBA ordena uma análise dos dados coletados, o programa faz a

análise pelos arquivos XML e gera relatórios e gráficos de acesso aos dados.

O programa é composto pelos módulos apresentados na Figura 10. O módulo *gui*, ou *Módulo de Interface Gráfica com o Usuário,* é responsável pela interface gráfica do programa. O módulo *control*, ou *Módulo de Controle*, é responsável por fazer o gerenciamento do controle geral da execução do programa. É chamado *db* o módulo que faz a conexão com o banco de dados e requisita os metadados. O módulo log é responsável pela leitura de log extraindo as informações necessárias para auditoria. O pacote *sql* é responsável por fazer a leitura e análise das consultas para definir que tipo de operação se trata, tabelas e colunas acessadas ou alteradas. O pacote *xml* é responsável pela escrita e leitura nos arquivos XML. Os módulos *xml.metadata* e *xml.data* possuem as entidades que serão transformadas em nodos nos arquivos XML de metadados e do *log*, respectivamente.

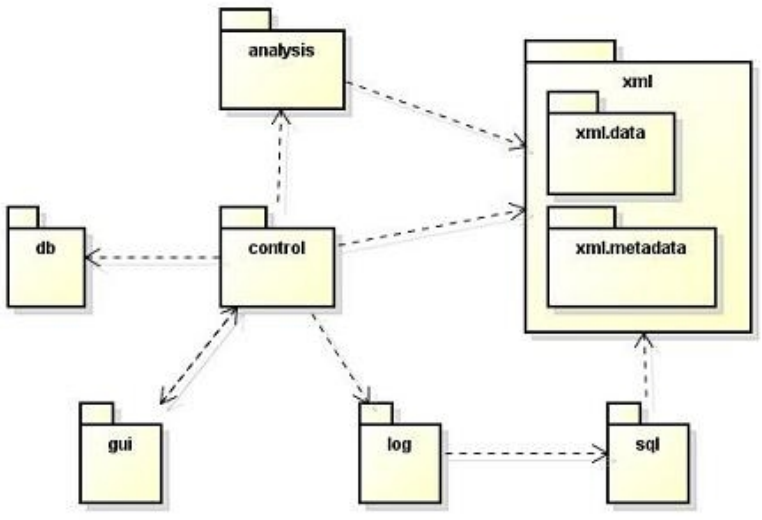

Figura 10: Módulos do DB Audilog

A Figura 11 apresenta o diagrama das classes do pacote de *Leitura de Log*. A classe *LogReader* é uma interface que deve ser implementada para cada tipo ou

formato diferente de log ao qual o DB *Audilog* dá suporte. Como nesta primeira versão, apenas o *PostgreSQL* é suportado, foi implementada a classe *PostgreSQLLogReader*, que implementa *LogReader*.

O método *isNewCel*() serve para avaliar se a linha do arquivo que foi lida é um novo registro do *log*. Para descobrir se o registro lido se trata de um comando, o método *isNewCommand*() deve ser utilizado. Os métodos *getDate*(), *getUser*(), *getComando*() extraem os dados do registro de *log*.

Após um registro ser lido, uma classe *LogCel* é instanciada para armazenar os dados do registro. Entre estes dados, está o comando SQL, que é mapeado por uma classe chamada *ComandoSQL* através da chamada do *SQLReader*, que transforma um *String* de comando SQL lido no *log* em um objeto da classe *ComandoSQL*. A classe *SQLReader* do pacote *sql* avalia quais tipos de consultas SQL devem ser considerados para análise de forma a escrever no XML apenas aqueles que são úteis nas análises. É ela que chama o *SQLDeParser* para análise da consulta SQL e extração das informações para armazenamento no objeto *ComandoSQL*.

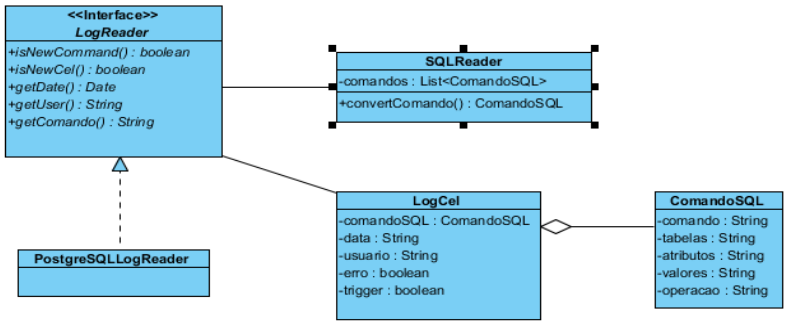

Figura 11: Diagrama de Classes do Pacote de Leitura de Log

Os atributos das classes *LogCel* e *ComandoSQL* são mapeados em um arquivo XML.

Toda manipulação dos arquivos XML, desde a criação até a leitura, foi feita com uma biblioteca de classes chamada *XStream*. Esta biblioteca pode ser utilizada

para serializar e desserializar um arquivo XML.

O pacote *sql* reimplementa algumas classes do *framework JSQLParser* para extrair as informações necessárias para auditoria, como tabelas acessadas ou colunas alteradas.

O *JSQLParser* é um *framework* que analise a consulta e a transforma numa hierarquia de classes que pode ser navegada através do padrão de projeto *Visitor*.

O módulo *db* contém as classes que fazem conexão com o banco de dados. A classe *ConnectionManager* faz a conexão com o banco de dados. Já a classe *MetadataManager* obtém os metadados do banco através da API chamada *SchemaCrawler*, e os armazena num arquivo XML.

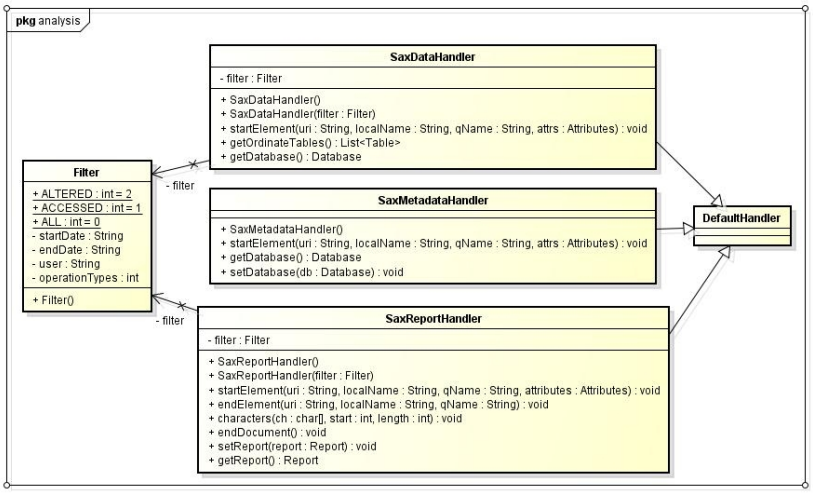

Figura 12: Módulo analysis

O módulo *analysis* pode ser visto na Figura 12. As classes *SaxDataHandler*, *SaxReportHandler*, *SaxMetadataHandler* são responsáveis pela leitura e extração de informação dos XML mapeando as informações para objetos que serão utilizados para criação dos gráficos da auditoria, de relatórios ou da apresentação dos metadados, respectivamente. Estas classes herdam de *DefaultHandler*, que é uma

classe da API chamada SAX, que é uma API para leitura de XML. A classe *Filter* mantém os tipos de filtros que podem ser feitos na auditoria para que sejam utilizados na leitura dos arquivos XML na auditoria de forma que apenas os dados requisitados sejam obtidos.

No pacote *analysis.report* se encontram as classes *GenericQuery* e *Query*, que são utilizadas para criação da estrutura para criação do relatório, que é mantida em um objeto da classe *Report*.

Nos pacotes *xml.data* e *xml.metadata* encontram-se as classes *Database*, *Schema*, *Table* e *Column* que são utilizadas pelo *XStream* para geração dos XML e, também, para criação de uma estrutura para criação dos gráficos da auditoria.

5. CONCLUSÃO E TRABALHOS FUTUROS

O DB *Audilog* é uma ferramenta que dá suporte a auditoria de banco de dados com uma interface de fácil uso. Além disso, o *Audilog* foi implementado de forma a facilitar futuras implementações de leituras de *logs* diferentes para que se possa fazer as mesmas análises a partir de *logs* de SGBDs diferentes.

O trabalho desenvolvido atende bem a auditoria para melhoria de performance do banco e parcialmente na questão da segurança do banco, pois trata apenas das consultas DML que são feitas no banco.

A escolha da ferramenta para auditoria ou técnica utilizada para implementar a mesma depende do nível de granularidade desejado de acordo com os objetivos definidos para auditoria.

Para uma auditoria mais completa deve-se utilizar técnicas como *triggers* para manter os valores anteriores e posteriores a alterações nas tabelas e agentes para implementação de alertas em tempo real. Para garantia da segurança do banco de dados também torna-se necessário o tratamento de todos os outros tipos de consultas SQL e de eventos de sistema na auditoria. Em próximas versões do programa todas as consultas SQL serão consideradas nas análises.

O DB *Audilog* pode ser bastante útil para Sistemas de Gerenciamento de Banco de Dados gratuitos como *PostgreSQL* e *MySQL* que não têm auditoria nativa.

Uma das grandes vantagens da geração dos dados de auditoria ser baseada nos arquivos de *log* é que, tendo os arquivos de *log*, pode-se fazer a auditoria fora do servidor do banco de dados. Para facilitar que isto seja feito, pretende-se estudar mais detalhadamente uma maneira de obter os *logs* do SGBD do servidor através de

um Sistema Gerenciador de Streams de Dados, como apresentado no item 3.1 do capítulo 3.

Como trabalho futuro pretende-se implementar a leitura de *log* para o *MySQL,* para que se possa fazer as análises a partir dos *logs* gerados pelo MySQL. Futuramente o Audilog também fará análise sob consultas DDL, eventos de sistema e transações sem sucesso na execução.

Em uma versão próxima do programa, a obtenção da estrutura do banco e dos usuários do banco de dados poderá ser obtida através da varredura do XML gerado ou pela inserção do filtro desejado através de um campo de texto, desta forma será possível fazer filtros por tabela e usuário sem a necessidade de conexão com o banco.

REFERÊNCIAS BIBLIOGRÁFICAS

ARASU, Arvind et al. **STREAM**: The Stanford Data Stream Management System. Disponível em: http://ilpubs.stanford.edu:8090/641/ . Acesso em: 13 março 2011.

CHOPSKIE, Ed. **Database Auditing Tools and Strategies**. Disponível em: http://www.meritalk.com/uploads_legacy/whitepapers/Introduction_to_Database_Auditing-May_2008.pdf . Acesso em: 12 junho 2011.

CHUVAKIN, Anton. **Introduction to Database Log Management**. Disponível em: http://www.infosecwriters.com/. Acesso em 02 fevereiro 2011.

IBM. **Introduction to the DB2 audit facility**. Disponível em: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.sec.doc%2Fdoc%2Fc0005483.html . Acesso em: 03 março 2011.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). **ISO 19011**: Guidelines for quality and/or environmental management systems auditing. ISO, 2002. Disponível em: http://chemeng.hut.edu.vn/images/stories/iso/tc-tk/ISO19011_2002_Ban%20tieng%20Anh.pdf . Acesso em: 21 março 2011.

MULLINS, Craig. **Database Administration:** The Complete Guide to Practices and Procedures. Boston: Addison-Wesley, 2002.

ORACLE. **Oracle Database SQL Reference**. Disponível em: http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_4007.htm . Acesso em 20 dezembro 2010.

SIMON, Fernando; DOS SANTOS, Aldri L.; HARA, Carmem S. Um Sistema de Auditoria baseado na Análise de Registros de Log. In: ESCOLA REGIONAL DE BANCO DE DADOS, IV, 2008, Florianópolis-SC. **Anais da Escola Regional de Banco de Dados 2008**. Disponível em: *www.inf.ufpr.br/carmem/pub/erbd08.pdf* . Acesso em: 13 janeiro 2011.