Um Sistema de Auditoria baseado na Análise de Registros de Log

Fernando Simon[1], **Aldri L. dos Santos**[1], **Carmem S. Hara**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-900 – Curitiba – PR – Brasil

{fernandos, aldri, carmem}@inf.ufpr.br

Trabalho de Pós-graduação

Abstract. *Database Management Systems are essential in many applications. They are fundamental to guarantee the safety of stored data, and thus it is important to monitor when and which data has been read or written in order to prevent and detect misuse. This is usually achieved by auditing a log of records generated for this purpose. Nevertheless, if a record is generated for every operation on the database, the volume of log data may become overwhelming. In this case, it is important to define more specific auditing policies, including the selection of data to audit. This paper proposes the use of a Data Stream Management System (DSMS) to audit a database. The advantages of this approach are twofold. First, the database administrator can filter out auditing records, and define policies as DSMS queries. Second, auditing results are generated in real time, minimizing the amount of storage needed for logging records.*

Resumo. *Gerenciadores de banco de dados são fundamentais em diversas aplicações, e garantir que os dados armazenados permaneçam seguros é um desafio. Controlar quando e como o dado foi acessado é possível através de auditoria. A auditoria em geral é baseada em registros de log gerados para este fim, que cresce em volume à medida que aumenta a utilização do banco de dados. Dependendo deste volume, é necessário definir políticas mais específicas sobre quais tabelas e colunas auditar, bem como os tipos de consultas. Este artigo propõe a utilização de um Sistema Gerenciador de* Streams *de Dados (SGSD) para realizar a auditoria em um banco de dados. Esta estratégia tem como vantagem permitir que o administrador do banco de dados defina os parâmetros de auditoria e os dados a serem auditados, bem como a obtenção de resultados em tempo real.*

1. Introdução

A evolução dos aplicativos nos últimos anos tornou o uso de sistemas gerenciadores de banco de dados (SGBD) peça fundamental nas suas arquiteturas. Características antes presentes e necessárias somente em grandes sistemas, como acesso direto aos dados, múltiplas conexões, integridade e durabilidade, passaram a fazer parte como requisitos da maioria dos sistemas. O uso de um SGBD tem reduzido o tempo de desenvolvimento de um projeto, sendo responsável por prover funcionalidades antes implementadas pela aplicação. Por outro lado, o uso de um banco de dados exige a realização de tarefas adicionais, como a gerência, configuração e manutenção do mesmo.

Da mesma forma que os aplicativos evoluíram na última década, os SGBDs também tiveram um salto de qualidade significativo. Alguns diminuíram a sua dependência do administrador, criando soluções automatizadas para recuperação de falhas, gerenciamento de *cache* de disco, atualização de parâmetros, dentre outras funcionalidades. Com a explosão do uso da *Internet*, os aplicativos passaram a ser utilizados globalmente, com usuários conectados de diferentes partes do mundo, acessando os mais diferentes dados e ao mesmo tempo. Assim, os problemas relacionados com segurança de acesso aos dados, cresceram proporcionalmente ao número de requisições e usuários. O controle sobre os dados armazenados em um SGBD passou a ser crucial. A identificação de quem acessa os dados, o momento do acesso e ações realizadas pelo usuário criaram demanda para um novo tipo de ação, a auditoria [Pavlou e Snodgrass 2006].

A auditoria procura identificar e evitar ações suspeitas e fraudulentas por parte do usuário, coletando dados sobre suas atividades no banco de dados. As informações coletadas são então analisadas a fim de descobrir problemas de segurança e sua origem. A necessidade de identificar quais foram as ações e quais os padrões suspeitos são importantes requisitos para a segurança do sistema.

[Hawthorn et al. 2006] destaca que a auditoria deve ser realizada de maneira independente e transparente, de forma que todas as informações relevantes devem ser catalogadas. A maioria dos SGBDs apresenta uma forma de catalogar estas ações, gerando *logs* de auditoria. Infelizmente poucos apresentam métodos transparentes, sendo que muitos necessitam da criação de *triggers* para cada objeto analisado. O uso de *triggers* é inadequado, pois onera o uso do banco de dados por adicionar rotinas que devem ser executadas a cada ação realizada [Sallachl 1992]. A geração de dados de auditoria pode ser implementada através de funções genéricas ou através de políticas de uso do banco e *logs* automáticos.

[Sandhu e Samarati 1996] dividem a auditoria em dois tipos, sendo o primeiro a *intrusão passiva*, na qual a análise é feita com dados coletados anteriormente, e o segundo o *sistema ativo*, no qual a análise é realizada em tempo real. O sistema ativo tem a vantagem de possibilitar reações imediatas, tomando ações visando a proteção dos dados. A desvantagem é que esse tipo de auditoria pode sobrecarregar o sistema gerenciador de banco de dados para a realização da auditora em tempo real. Uma forma de obter análises rápidas sem afetar o desempenho do SGBD é direcionar o fluxo de registros de *log* para processamento por outro sistema.

Neste artigo é proposta a utilização de um Sistema Gerenciador de *Streams* de Dados (SGSD) para este propósito. Neste tipo de sistema, o processamento de fluxos de dados (*streams*) é realizado através de máquinas de processamento de *streams* (MPE), que dão suporte a uma nova classe de processamento, chamada de aplicações baseadas em fluxos de dados. Neste tipo de aplicação, os dados são enviados para o sistema na forma de um fluxo de registros e as consultas são continuamente aplicadas sobre estes fluxos [Hwang et al. 2005].

A integração destes dois métodos, auditoria e processamento de *streams*, pode garantir que os registros de *log* sejam processados em tempo real, possibilitando que os registros possam ser filtrados e as análises dos dados que o administrador de banco de dados achar necessárias sejam expressas através de consultas definidas no SGSD.

Este trabalho descreve o desenvolvimento de um aplicativo que utiliza o SGSD Borealis [Abadi et al. 2005] para ler e processar registros de auditoria oriundos do SGBD Postgres [Postgres 2006].

As próximas seções estão organizadas da seguinte forma: a seção 2 apresenta os trabalhos relacionados e algumas características dos SGSDs; a seção 3 descreve o desenvolvimento do sistema de auditoria; a seção 4 conclui o artigo e apresenta alguns trabalhos futuros.

2. Trabalhos relacionados e os SGSDs

Segundo [Sallachl 1992] auditoria é uma fonte de informação necessária em sistemas específicos, como financeiros, ou nos quais dados confidenciais são armazenados. Ainda segundo este autor, quando a auditoria é realizada, discrepâncias nos padrões de utilizaçcão dos dados tornam-se visíveis e podem ser identificados. Desta forma, um SGBD deve manter catalogadas todas as alterações realizadas sobre o mesmo. [Haraty 1999] comparou alguns métodos utilizados por sistemas comerciais para catalogar as alterações, onde em sua maioria estes são armazenados na forma de *logs*, sendo sua posterior análise uma etapa importante para a identificação de problemas. [Debar et al. 1999] define uma taxonomia para ser empregada na detecção de intrusão em sistemas de rede, apresentando uma divisão e categorização de métodos de auditoria que podem ser empregados de maneira genérica. Já [Lunt 1993] apresenta algumas definições quanto ao uso de auditoria para a detecção de intrusão, focando nos modos corretos de catalogar os registros e de quais consultas podem ser aplicadas. [Chung et al. 1999] é mais específico na área de banco de dados e apresenta um aplicativo próprio para a realização de auditoria, e a arquitetura de um sistema para tal fim.

Porém, nenhum dos autores pesquisados apresenta de maneira genérica como realizar a auditoria, identificando quais os padrões mais comuns para ações de usuários, ou quais as regras que devem ser seguidas para auditar um banco de dados. Outro detalhe observado é que nenhum autor cita o uso de métodos como a auditoria através da análise de *streams*. Cada artigo apresenta métodos próprios ou descrevem ferramentas que realizam a coleta dos dados e a auditoria.

Os Sistemas Gerenciadores de *Streams* de Dados (SGSDs) foram propostos para prover as funcionalidades de um SGBD sobre fluxos contínuos de dados, fornecendo respostas em tempo real, bem como resultados aproximados. Uma discussão sobre processamento de *streams* em tempo real pode ser encontrada em [Stonebraker et al. 2005]. A principal diferença entre os SGBDs e os SGSDs consiste em como os dados e as consultas persistem no sistema [Koudas e Srivastava 2003]. O primeiro contém informação estática e consultas dinâmicas, enquanto o último tem o comportamento inverso. Ou seja, os bancos de dados tradicionais normalmente executam consultas diferentes sobre o mesmo conjunto de dados. Já os SGSDs executam as mesmas consultas sobre dados que chegam ao longo do tempo. Em algumas situações, como no caso da auditora, é interessante integrar os dois sistemas. Desta forma é possível armazenar os resultados obtidos com o processamento de *streams* para consultá-los posteriormente.

Os trabalhos de pesquisa envolvendo SGSDs são recentes, e a maioria dos sistemas desenvolvidos ainda são protótipos. Dentre eles podem ser citados: o Borealis [Abadi et al. 2005] - o qual foi baseado nos sistemas Aurora [Abadi et al. 2003] e Medusa

[Balazinska et al. 2004] desenvolvidos pelo mesmo grupo; o TelegraphCQ [Chandrasekaran et al. 2003] - que foi implementado de forma a ser uma extensão do SGBD Postgres; o STREAM [Arasu et al. 2003] - que foi um dos pioneiros nesta área de pesquisa; e por fim, o Gigascope [Cranor et al. 2003] - que é um projeto comercial com resultados significativos no monitoramento de redes.

Estudos feitos sobre estes protótipos [Plagemann et al. 2004, Ahmad et al. 2005, Abadi et al. 2004] apresentaram resultados encorajadores. O Borealis foi escolhido para implementar o sistema de auditoria porque, dentre os SGSDs acima citados, ele é o único sistema distribuído. Assim, é possível que o SGBD a ser auditado esteja sendo executado em uma máquina e o SGSD em outra. Além disso, o Borealis possui características inovadoras, tais como *registros de revisão*, *viagem no tempo* e *linhas de controle* [Ahmad et al. 2005]. Tal como em qualquer banco de dados distribuído [Lima et al. 2003], o Borealis também permite a integração dos dados e o compartilhamento de recursos, e também implementa mecanismos de tolerância a falhas, processamento distribuído, escalabilidade, e balanceamento e dispersão de carga [Ahmad et al. 2005].

3. Auditoria baseada em um SGSD

Nesta seção será exposta a arquitetura do sistema de auditoria, bem como os métodos utilizados para o seu desenvolvimento. Como requisito, existe a necessidade de realizar a auditoria em *logs* de ações do PostgreSQL, tal que os padrões de análise sejam genéricos para serem utilizados independente do esquema do banco de dados.

3.1. Arquitetura

A arquitetura do sistema é ilustrada na Figura 1. O usuário interage com o SGBD enviando consultas e obtendo seus resultados. Cada operação sobre o SGBD gera um registro de *log*, que é enviado para o SGSD. Este fluxo de registros de *log* é processado pelo SGSD para responder às consultas de auditoria registradas pelo administrador do SGBD a fim de obter informações necessárias para a auditoria do sistema. Os resultados das consultas podem ser obtidas tanto em tempo real, como podem ser armazenadas no próprio SBGD para posterior consulta.

As vantagens da arquitetura proposta são:

1. flexibilidade: ela permite a filtragem dos registros de *log* e consultas arbitrárias podem ser registradas no SGSD;
2. independência: a auditoria é realizada por um aplicativo externo ao SGBD, além de poder ser executada em uma máquina distinta, aumentando assim a segurança e controle do sistema.

Na sequência, detalhes sobre os componentes do sistema são apresentados.

3.2. O SGBD PostgreSQL

O PostgreSQL foi utilizado como o SGBD sobre a qual a análise e a auditoria são realizadas. Ele foi escolhido por permitir a customização dos dados gerados para possibilitar a auditoria do sistema, porém ele não apresenta facilidades para a análise destas informações [Postgres 2006]. Ou seja, o SGBD permite que no seu arquivo de configurações possa ser definido que o sistema gere saídas sobre suas atividades. Dentre os

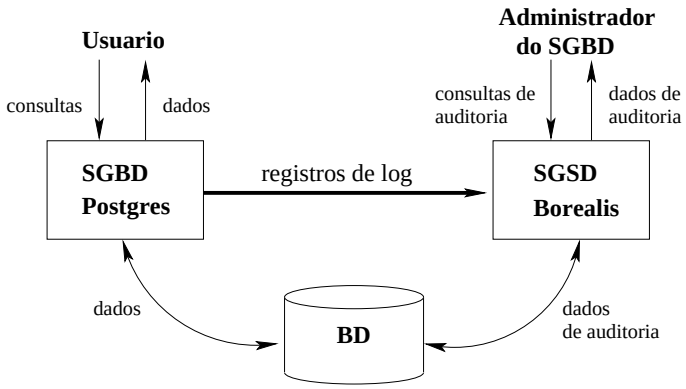

Figura 1. Arquitetura do sistema.

parâmetros presentes neste arquivo está a criação de um *log* para cada atividade realizada pelo SGBD. Este *log* pode conter, além de informações sobre acesso aos dados, diversas outras, como as mensagens de conexões, erros de autenticação e erros de consultas SQL.

Um exemplo contendo registros de *log* gerados pelo PostgreSQL é apresentado na Figura 2. Neste caso, o arquivo de configurações foi definido para gerar somente informações sobre acessos aos dados. Como pode ser observado, os registros são gerados a cada consulta realizada pelo cliente, nos quais estão presentes as informações sobre os tipos de consultas e duração das mesmas. São estes os parâmetros que serão utilizados como base para a realização da auditoria.

Org:127.0.0.1(57303);Tsmp:2007-06-24 17:04:02.291BRT;DB:db-curso; Usr:us-curso;Cmd:SELECT;ID:467ece31.1601-5;LOG: duration: 9.861 ms
Org:127.0.0.1(57303);Tsmp:2007-06-24 17:04:02.669 BRT;DB:db-curso; Usr:us-curso;Cmd:UPDATE;ID:467ece31.1601-6;LOG: duration: 376.788 ms
Org:127.0.0.1(57303);Tsmp:2007-06-24 17:04:02.697 BRT;DB:db-curso; Usr:us-curso;Cmd:SELECT;ID:467ece31.1601-7;LOG: duration: 27.878 ms

Figura 2. Exemplo de *log*.

Embora o SGBD possa ser configurado para gerar estas informações, ele não oferece nenhum aplicativo para fazer a análise das mesmas. Como já foi citado, a auditoria tem a função de identificar problemas relacionados à segurança dos dados, que incluem detecção de intrusão e abuso de privilégios. Além disso, ela também pode ser utilizada para a identificação de problemas no desempenho do sistema. [Debar et al. 1999] apresenta as seguintes características para um sistema de auditoria, dentre as quais as principais são:

1. método de deteção: é a forma como detecção é realizada;
2. comportamento: define as ações tomadas pelo sistema na presença de problemas;
3. origem: determina a origem do problema e
4. freqüência: caracteriza o modo como é aplicada a análise.

O método proposto neste artigo, que é a integração das tecnologias de SGBDs com SGSDs faz a análise de maneira contínua sobre o fluxo de registros de *log*, apresentando um comportamento passivo no caso de detecção de algum problema.

[Chung et al. 1999] destaca que para realizar a auditoria é necessário repassar as políticas e regras de uso para os mecanismos de segurança. Esta tradução de regras para um aplicativo é difícil de ser obtida e necessita que os padrões já estejam definidos. [Lunt 1993] destaca que a auditoria também pode ser realizada para analisar o sistema, sem a necessidade de identificar questões de segurança ou abuso de privilégios. Este autor também apresenta um conjunto de regras que podem ser utilizadas para a gerência do sistema. Qualquer que seja o propósito da auditoria, é necessário definir as regras e políticas para a mesma. Na abordagem proposta neste artigo, estas regras são expressas através de consultas registradas no SGSD, como descrito na próxima seção.

3.3. O SGSD Borealis

Borealis é um sistema distribuído para processamento de *streams*. O desenvolvimento de uma aplicação utilizando o SGSD consiste nos seguintes passos: definição do esquema dos registros de entrada e saída, definição das consultas a serem aplicadas sobre os fluxos de dados e codificação de um programa para transformar o fluxo de entrada no esquema definido.

Tanto a definição dos esquemas dos registros de entrada e de saída, quanto as consultas são feitas através de um arquivo no formato XML, como mostrado na Figura 3. O arquivo XML deve conter as definições dos *streams* de entrada e saída (Linhas 1 e 2), bem como o esquema dos registros que compõe o fluxo de entrada (Linhas 3 a 13) e saída (Linhas 14 a 17). Para campo do registro são definidos o nome, tipo e tamanho.

As consultas a serem registradas no SGSD são também expressas em XML. O Borealis dá suporte a diversos tipos de operadores, divididos em dois tipos:

- *sem estado*: são operadores que realizam a computação sobre um único registro por vez. Como exemplo existem: *Filter* (seleção), *Map* e *Union*(união);
- *baseado em estado*: são operadores que realizam a computação sobre um grupo de entradas. O número de entradas que compõem o grupo é definido através de uma janela, que pode ser definida por tempo ou número registros. O operador mais comum é o de agregação.

Neste artigo é apresentada uma consulta simples, que obtém o tempo médio de execução de cada tipo de comando, que consistem em *SELECT, INSERT, UPDATE, DELETE e INTERNAL*. Este último contempla todos os comandos internos do banco de dados, como indexações e alterações de estruturas. Porém, cabe ressaltar que consultas bastante complexas podem ser expressas na linguagem oferecida pelo sistema. Por exemplo, é possível determinar se o padrão de utilização do SGBD apresentado por um determinado usuário em uma sessão difere dos padrões armazenados no SGBD no último mês. Estes padrões podem ser obtidos através de dados de auditoria armazenados no SGBD.

A consulta sobre médias de comandos é ilustrada na Figura 3 (Linhas 18 a 32). O operador de agregação permite gerar uma média dos valores de entrada em uma determinada janela de tempo. O conceito de janela é importante no processamento de fluxos contínuos de dados, pois não é possível ler toda a entrada de dados para obter o resultado de uma operação de agregação. Os parâmetros utilizados na consulta são detalhados abaixo.

```
1.  <input stream="LogReg"      schema="LogRegTuple"/>
2.  <output stream="Aggregate"schema="AggregateTuple"/>
3.  <schema name="LogRegTuple">
4.  <field name="time"        type="int"/>
5.  <field name="origem"      type="string"size="32"/>
6.  <field name="tsmp"        type="string"size="27"/>
7.  <field name="dbname"      type="string"size="32"/>
8.  <field name="usrname"     type="string"size="32"/>
9.  <field name="cmd"         type="string"size="10"/>
10. <field name="id"          type="string"size="32"/>
11. <field name="duration"    type="double"/>
12. <field name="erro"        type="int"/>
13. </schema>
14. <schema name="AggregateTuple">
15. <field name="cmd"         type="string"size="10"/>
16. <field name="avg"         type="double"/>
17. </schema>
18. <query name="sqlquery">
19. <box name="sqlavg"type="aggregate">
20. <in   stream="LogReg"/>
21. <out   stream="Aggregate"/>
23. <parameter  name="aggregate-function.0"                  value="avg(duration)"/>
24. <parameter  name="aggregate-function-output-name.0"value="avg"/>
25. <parameter  name="window-size-by"                        value="TUPLES"/>
26. <parameter  name="window-size"                           value="10"/>
27. <parameter  name="advance"                               value="5"/>
28. <parameter  name="group-by"                              value="cmd"/>
29. <parameter  name="order-by"                              value="TUPLENUM"/>
30. <parameter  name="drop-empty-outputs"                    value="1"/>
31. </box>
32. </query>
```

Figura 3. Definições em XML para o SGSD Borealis.

- `aggregate-function.0`: este parâmetro define a função utilizada para a agregação. Neste exemplo é utilizada a média sobre o tempo de duração dos tipos de consultas;
- `window-size-by`: define o método utilizado para marcar a janela de tempo; neste exemplo ele foi definido como `TUPLES`, gerando assim janelas sobre o número de entradas;
- `window-size`: especifica o número de registros que compõem a janela. No exemplo este número foi definido como 10;
- `advance`: este parâmetro define o tamanho do passo da janela sobre as entradas. Assim, mesmo que o tamanho da janela seja 10 o passo é de somente 5, permitindo que as janelas sejam sobrepostas;
- `group-by`: estabelece os campos que definem os agrupamentos sobre os quais

1. Tempo medio para a execucao dos comando abaixo listados
2. Os tempos estao expressos em ms
3. Select Insert Update Delete Internal
4. 8.60 198.00 641.70 260.00 166.50

Figura 4. Resultados obtidos.

são aplicadas as operações de agregação.

Associado ao arquivo XML contendo os esquemas e consultas, é necessário o desenvolvimento de um programa em C++ que recebe o fluxo de dados, transforma-os em registros de acordo com o esquema de entrada pré-definido, e envia-os para o Borealis. As interfaces das funções que devem ser implementadas são geradas pelo Borealis quando os esquemas e as consultas são registrados no sistema. Assim, algumas funções, como a de captação da entrada, sua interpretação e formatação de acordo com o esquema de entrada e envio dos registros ao Borealis têm que ser codificados pelo usuário. As saídas do sistema também devem ser tratadas pelo programa, que podem ser exibidas diretamente para o administrador do sistema, bem como ser armazenadas no SGBD para posterior consulta.

Com a compilação deste programa é gerado um aplicativo que quando executado lê o arquivo de *logs* e envia ao Borealis seus registros para a análise. Neste exemplo, as saídas de sua execução são exibidas diretamente no monitor. A Figura 4 apresenta o resultado de uma execução do programa. Os valores expressam os tempos médios para a janela de tempo definida, sendo que os valores são atualizados em tempo real.

4. Conclusão e trabalhos futuros

Com o desenvolvimento deste sistema pode-se observar que a realização de auditoria de banco de dados através de *streams* é possível, permitindo uma redução da complexidade do aplicativo a ser desenvolvido. Essas características somente puderam ser obtidas pelo uso do Borealis, por ele apresentar uma forma precisa e rápida para o processamento de dados. Além disso, como ele oferece uma linguagem de alto nível, alterações em regras e políticas de auditoria requerem apenas alterações nas consultas registradas no sistema.

As tecnologias empregadas para a auditoria e análise de registros permitiram a criação de um aplicativo que pode agregar novas funcionalidades ao PostgreSQL, tais como a análise em tempo real do desempenho, os tempos de acesso, além de outras formas de auditoria. Mesmo com o uso de uma auditoria simples, o desenvolvimento deste sistema permitiu a identificação de novos mecanismos de análise, a demonstração de novos métodos para a coleta, e a possibilidade de análise distribuída de dados, métodos estes que estão cada vez mais presentes em sistemas de bancos de dados.

Apesar do sistema desenvolvido ser bastante simples, ele mostrou que as tecnologias de SGBD e SGSD podem ser empregadas em conjunto, dando suporte a uma grande demanda e permitindo respostas rápidas. A viabilidade da integração destas duas tecnologias, processamento de *stream* e auditoria, lança novos desafios de desenvolvimento, abrindo novas possibilidades quanto ao emprego não somente em banco de dados mas em áreas que necessitem de análise de *logs* em busca de padrões.

Dois pontos serão abordados em trabalhos futuros. O primeiro é a própria auditoria, com um estudo mais detalhado sobre o poder de expressão necessário para definir consultas que expressem as regras e políticas mais comumente empregadas nesta área. O segundo ponto consiste em tornar o sistema mais genérico, não havendo a necessidade de compilação do sistema sempre que uma nova consulta é registrada no sistema. Outro trabalho futuro é uma extensão do SGSD que possibilite o registro e a alteração de consultas dinamicamente.

Referências

Abadi, D. J., Ahmad, Y., Balazinska, M., Çentintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A. S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., e Zdonik, S. (2005). The design of the borealis stream processing engine. In *Proceedings of the 2nd Conference on Classless Inter-Domain Routing (CIDR'05)*, pages 277–289.

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E. F., Hatoun, M., Hwang, J.-H., Maskey, A., Rasin, A., Singer, A., Stonebraker, M., Tatbul, N., Xing, Y., Yan, R., e Zdonik, S. (2003). Aurora: A data stream management system. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03)*.

Abadi, D. J., Lindner, W., Madden, S., e Schuler, J. (2004). An integration framework for sensor networks and data stream management systems. In *Proceedings of 30th International Conference on Very Large Data Bases (VLDB'04)*, pages 1361–1364.

Ahmad, Y., Berg, B., Çetintemel, U., Humphrey, M., Hwang, J.-H., Jhingran, A., Maskey, A., Papaemmanouil, O., Rasin, A., Tatbul, N., Xing, W., Xing, Y., e Zdonik, S. (2005). Distributed operation in the borealis stream processing engine. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*, pages 882–884.

Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., e Widom, J. (2003). Stream: The stanford data stream management system. *IEEE Data Engineering Bulletin*, 26(1).

Balazinska, M., Balakrishnan, H., e Stonebraker, M. (2004). Load management and high availability in the medusa distributed stream processing system. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD'04)*, pages 929–930.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., e Shah, M. (2003). Telegraphcq: Continuous dataflow processing for an uncertain world. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR'03)*, pages 269–280.

Chung, C. Y., Gertz, M., e Levvit, K. (1999). Demids: A misuse detection system for database systems. In *Working Conference on Integrity and Internal Control in Information Systems*, pages 159–178.

Cranor, C., Johnson, T., Spatscheck, O., e Shkapenyuk, V. (2003). The gigascope stream database. *IEEE Data Engineering Bulletin*, 26(1):27–32.

Debar, H., Decier, M., e Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, (31):805–822.

Haraty, R. A. (1999). C2 secure database management systems: a comparative study. In *SAC*, pages 216–220.

Hawthorn, P., Simons, B., Clifton, C., Wagner, D., Bellovin, S. M., Wright, R. N., Rosenthal, A., Poore, R. S., Coney, L., Gellman, R., e Hochheiser, H. (2006). Statewide databases of registered voters: a study of accuracy, privacy, usability, security, and reliability issues. *Communications of the ACM*, 49(4):26–28.

Hwang, J.-H., Balazinska, M., Rasin, A., Cetintemel, U., Stonebraker, M., e Zdonik, S. (2005). High-availability algorithms for distributed stream processing. In *International Conference on Data Engineering (ICDE'05)*.

Koudas, N. e Srivastava, D. (2003). Data stream query processing: A tutorial. In *Proceedings of 29th International Conference on Very Large Data Bases (VLDB'03)*, pages 1149–1149.

Lima, A. A. B., Matoso, M. L. Q., e Esperança, C. (2003). Efficient processing of heavyweight queries in database clusters. Technical Report 001, UFRJ.

Lunt, T. L. (1993). Detecting intruders in computer systems. In *Conference on Auditing and Computer Technology*.

Pavlou, K. e Snodgrass, R. T. (2006). Forensic analysis of database tampering. In *SIGMOD*, pages 109–120.

Plagemann, T., Goebel, V., Bergamini, A., Tolu, G., Urvoy-Keller, G., e Biersack, E. W. (2004). Using data stream management systems for traffic analysis - a case study. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM'04)*, pages 215–226.

Postgres (2006). *PostgreSQL 8.2.3 Documentation*. PostgreSQL Global Development Group.

Sallachl, D. L. (1992). A deductive database audit trail. *Communications of ACM*, pages 314–319.

Sandhu, R. e Samarati, P. (1996). Authentication, access control, and audit. *ACM Computing Surveys*, 28(1):241–243.

Stonebraker, M., Çetintemel, U., e Zdonik, S. (2005). The 8 requirements of real-time stream processing. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*.