Auditoria de sistemas

Deixe que seu banco de dados faça o trabalho por você

por ***Cristóferson Guimarães Magalhães Bueno e Odilon Corrêa da Silva***

O objetivo deste artigo é demonstrar como criar um sistema de auditoria que se ajuste facilmente a qualquer tipo de aplicação que utilize banco de dados relacional, sem a necessidade de grandes alterações na mesma. No sistema proposto 99% do trabalho de auditoria será realizado pelo SGDB utilizado pela aplicação.

Conceito de auditoria

Auditoria consiste em um exame cuidadoso, sistemático e independente, cujo objetivo seja averiguar se as atividades desenvolvidas em determinada empresa ou setor estão de acordo com as disposições planejadas e/ou estabelecidas previamente, se estas foram implementadas com eficácia e se estão adequadas (em conformidade) à consecução dos objetivos [1].

Como irá funcionar o sistema de auditoria?

O conceito aplicado neste artigo é simples, mas bastante eficaz. A auditoria deve ser capaz de registrar quaisquer alterações feitas nos dados de um sistema, o dado original antes da manipulação, quem o manipulou e finalmente quando este dado foi manipulado.

Seguindo esta linha de raciocínio, nosso sistema de auditoria terá que receber algumas informações do usuário que acessa a base. O envio dessas informações é única parte da auditoria que deve ser implementado no sistema que se deseja auditar.

No caso de nossa aplicação exemplo, será passado como parâmetro o nome do usuário e o IP. Esses parâmetros serão armazenados nas tabelas de auditoria à medida que o usuário efetua as operações no banco.

Os tópicos a seguir descrevem os recursos dos SGBDs que serão necessários para a implementação da auditoria e a explicação de como os parâmetros serão armazenados para serem utilizados posteriormente.

Todas os exemplos deste artigo serão implementados utilizando o MySQL 5 e o mecanismo de armazenamento InnoDB, no entanto todas as soluções apresentadas aqui podem ser facilmente implementadas em outros bancos de dados relacionais, sem a necessidade de grandes mudanças, com exceção do PostgreSQL devido sua diferença na implementação das Triggers, mas nada que impeça o mesmo de ser utilizado com um pouco mais de esforço.

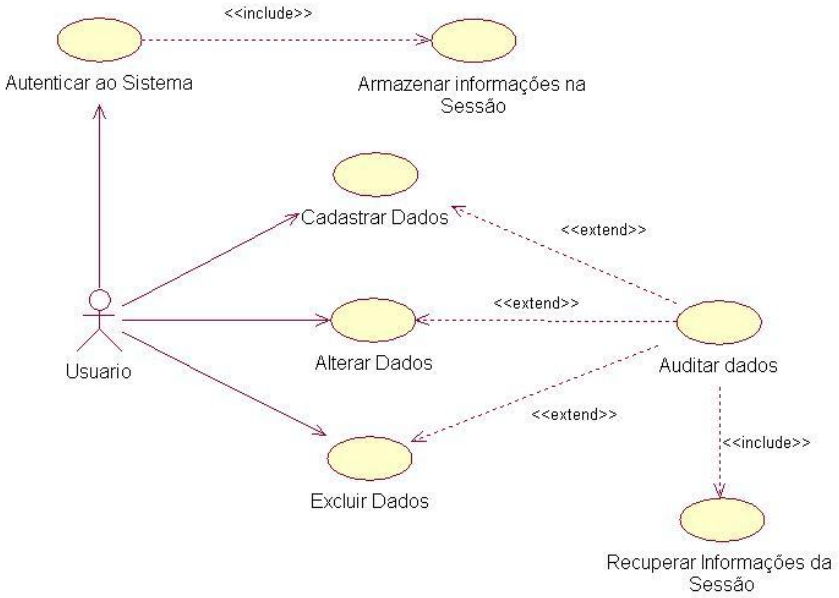

Figura 1. Diagrama de Caso de Uso do Sistema de Auditoria.

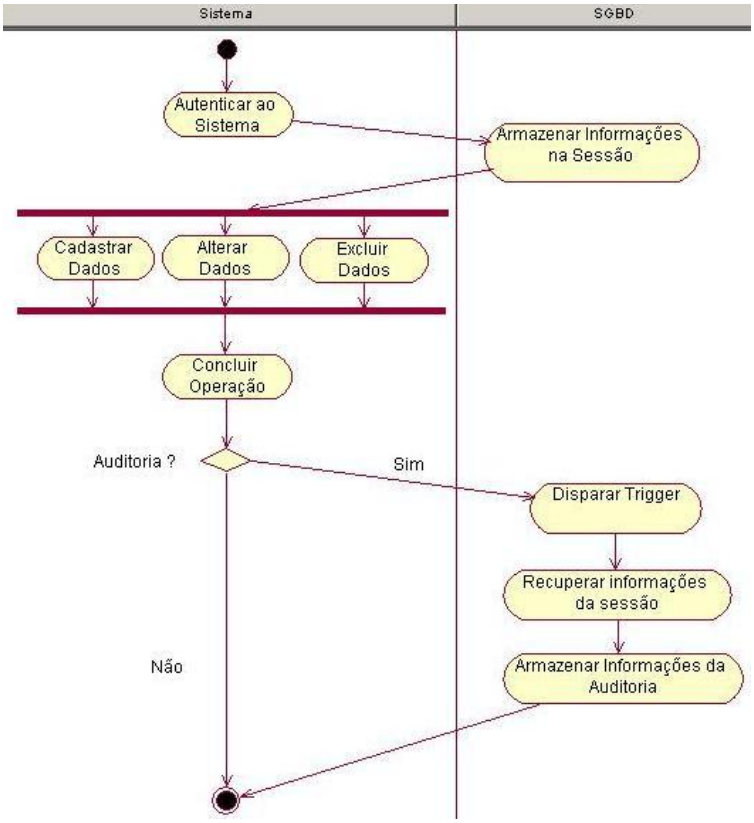

Figura 2. Diagrama de Atividades do Sistema de Auditoria.

Os diagramas acima descrevem as interações do sistema de auditoria com o SGBD, e armazena os eventos efetuados na base da aplicação.

Variáveis de Sessão, Triggers, Stored Procedures

Estes são os principais recursos necessários para criação do sistema de auditoria proposto neste artigo. Com eles e um pouco de criatividade seremos capazes de criar uma solução para auditoria que resida praticamente toda no banco de dados.

- **Variáveis de Sessão**

  São variáveis temporárias, criadas para armazenar dados, que possam ser posteriormente recuperados e utilizados em rotinas no banco. Essas rotinas podem ser simples consultas ou Stored Proceduress.

  Sempre que a aplicação efetuar uma conexão com o banco será necessário criar algumas variáveis de sessão e armazenar nelas os parâmetros da auditoria. No caso deste artigo os parâmetros armazenados nas variáveis de sessão serão, nome e IP, como citado acima.

- **Trigger**

  É um recurso de programação presente na maioria dos sistemas de gerenciamento de banco de dados. Ele é utilizado para associar um procedimento a um evento do banco de dados (inclusão, exclusão, atualização de registro, por exemplo) de modo que o procedimento seja executado automaticamente sempre que o evento associado ocorrer.

  As Triggers serão utilizadas na nossa aplicação para disparar as *Stored Procedures* que farão os registros dos eventos executados no banco, assim como povoar o dicionário de dados necessário para o sistema de auditoria.

- ***Stored Procedures***

  São programas ou procedimentos armazenados no banco de dados. Este recurso permite encapsular tarefas repetitivas, aceita a passagem de parâmetros e de entrada e retorna um valor de *status* (para indicar sucesso ou falha na execução). Stored Procedures permite reduzir tráfego de rede, melhorar a performance da aplicação e criar mecanismos de segurança.

Estrutura da aplicação modelo

A estrutura de tabelas a seguir servirá como modelo para exemplificar a criação e uso do sistema de auditoria.

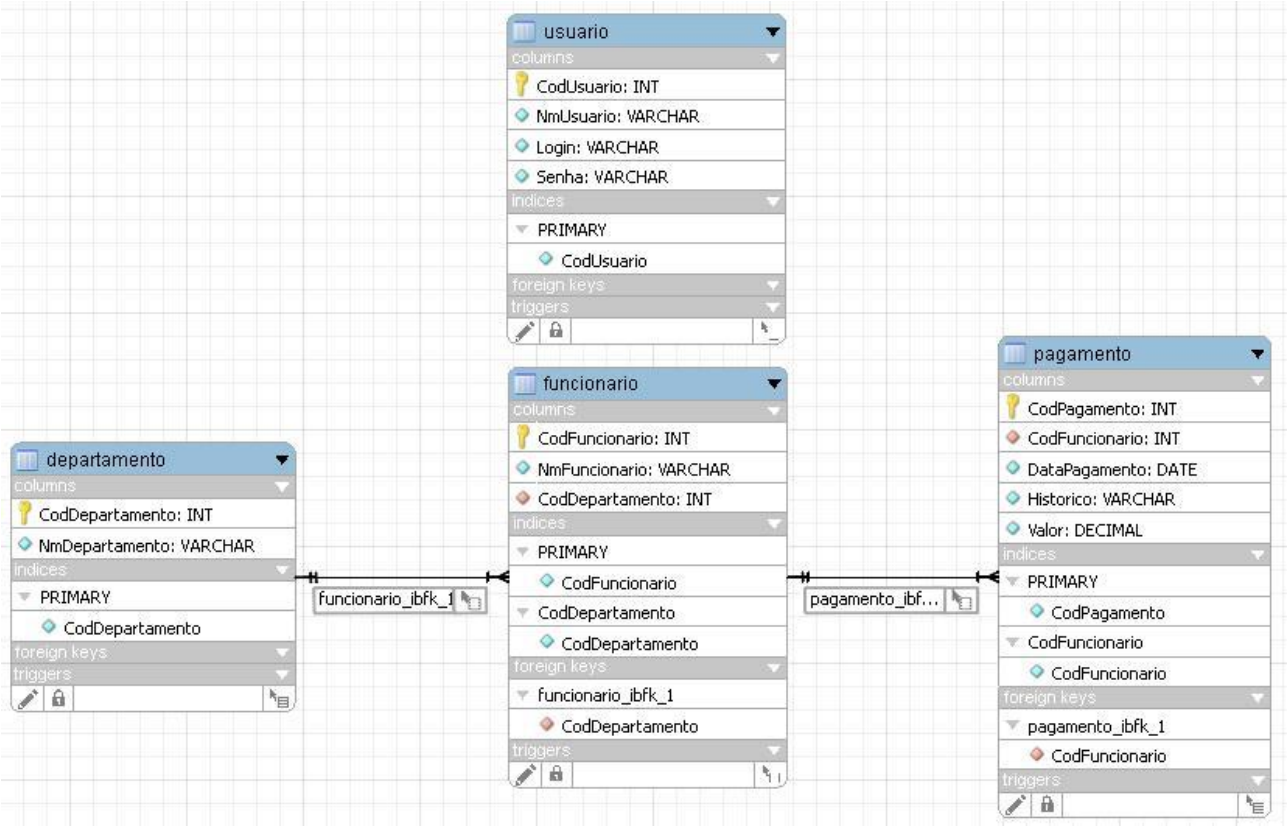

Figura. 3. Diagrama ER da aplicação modelo.

Código 1. SQL para a geração das tabelas da aplicação exemplo.

```
SET FOREIGN_KEY_CHECKS = 0;

CREATE DATABASE IF NOT EXISTS `FolhaPagamento`
  CHARACTER SET latin1;

DROP TABLE IF EXISTS `FolhaPagamento`.`departamento`;
CREATE TABLE `FolhaPagamento`.`departamento` (
  `CodDepartamento` INT(10) unsigned NOT NULL AUTO_INCREMENT,
  `NmDepartamento` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`CodDepartamento`)
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

DROP TABLE IF EXISTS `FolhaPagamento`.`funcionario`;
CREATE TABLE `FolhaPagamento`.`funcionario` (
  `CodFuncionario` INT(10) unsigned NOT NULL AUTO_INCREMENT,
  `NmFuncionario` VARCHAR(50) NOT NULL,
  `CodDepartamento` INT(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`CodFuncionario`),
  INDEX `CodDepartamento` (`CodDepartamento`),
  CONSTRAINT `funcionario_ibfk_1` FOREIGN KEY `funcionario_ibfk_1` (`CodDepartamento`)
    REFERENCES `FolhaPagamento`.`departamento` (`CodDepartamento`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

DROP TABLE IF EXISTS `FolhaPagamento`.`pagamento`;
CREATE TABLE `FolhaPagamento`.`pagamento` (
```

```
  `CodPagamento` INT(10) unsigned NOT NULL AUTO_INCREMENT,
  `CodFuncionario` INT(10) unsigned NOT NULL DEFAULT '0',
  `DataPagamento` DATE NOT NULL DEFAULT '0000-00-00',
  `Historico` VARCHAR(100) NOT NULL,
  `Valor` DECIMAL(10, 0) NOT NULL DEFAULT '0',
  PRIMARY KEY (`CodPagamento`),
  INDEX `CodFuncionario` (`CodFuncionario`),
  CONSTRAINT `pagamento_ibfk_1` FOREIGN KEY `pagamento_ibfk_1` (`CodFuncionario`)
    REFERENCES `FolhaPagamento`.`funcionario` (`CodFuncionario`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

DROP TABLE IF EXISTS `FolhaPagamento`.`usuario`;
CREATE TABLE `FolhaPagamento`.`usuario` (
  `CodUsuario` INT(10) unsigned NOT NULL AUTO_INCREMENT,
  `NmUsuario` VARCHAR(50) NOT NULL,
  `Login` VARCHAR(20) NOT NULL,
  `Senha` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`CodUsuario`)
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

SET FOREIGN_KEY_CHECKS = 1;
```

Dicionário de Dados

O dicionário de dados é uma estrutura de tabelas, cujo objetivo, será armazenar as informações do sistema a ser auditado (Figura 3).

Para uma melhor performance, é aconselhável separar as tabelas do sistema de auditoria, sejam criadas em um disco diferente do que está o esquema de banco a ser auditado, evitando assim concorrência de acesso a disco.

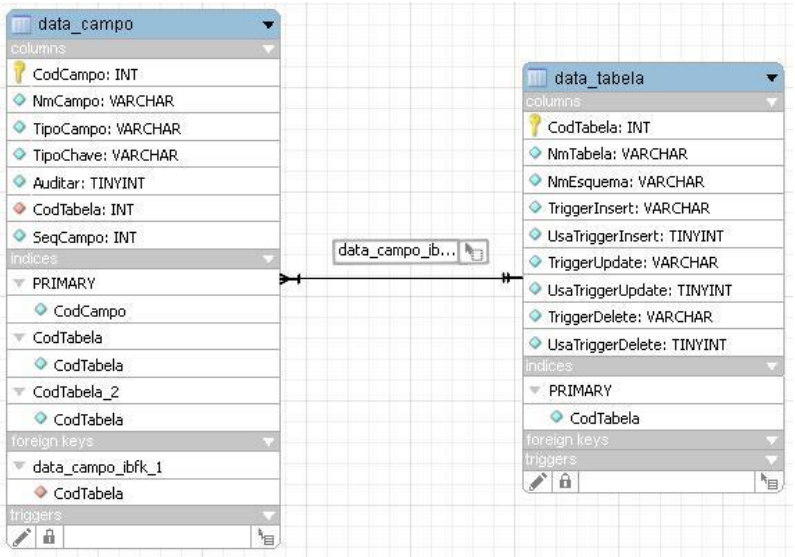

Figura 4. Estrutura do dicionário de dados.

Código 2. SQL para a geração das tabelas do Dicionário de Dados.

```
SET FOREIGN_KEY_CHECKS = 0;

CREATE DATABASE IF NOT EXISTS `Auditoria`
CHARACTER SET latin1;
-- Tables
DROP TABLE IF EXISTS `Auditoria`.`data_campo`;
CREATE TABLE `Auditoria`.`data_campo` (
  `CodCampo` INT(10) unsigned NOT NULL AUTO_INCREMENT,
  `NmCampo` VARCHAR(45) NOT NULL,
  `TipoCampo` VARCHAR(45) NOT NULL,
```

```
  `TipoChave` VARCHAR(45) NOT NULL,
  `Auditar` TINYINT(1) NOT NULL DEFAULT '0',
  `CodTabela` INT(10) unsigned NOT NULL DEFAULT '0',
  `SeqCampo` INT(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`CodCampo`),
  INDEX `CodTabela` (`CodTabela`),
  INDEX `CodTabela_2` (`CodTabela`),
  CONSTRAINT `data_campo_ibfk_1` FOREIGN KEY `data_campo_ibfk_1` (`CodTabela`)
    REFERENCES `Auditoria`.`data_tabela` (`CodTabela`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

DROP TABLE IF EXISTS `Auditoria`.`data_tabela`;
CREATE TABLE `Auditoria`.`data_tabela` (
  `CodTabela` INT(10) unsigned NOT NULL AUTO_INCREMENT,
  `NmTabela` VARCHAR(45) NULL,
  `NmEsquema` VARCHAR(45) NULL,
  `TriggerInsert` VARCHAR(45) NULL,
  `UsaTriggerInsert` TINYINT(1) NULL,
  `TriggerUpdate` VARCHAR(45) NULL,
  `UsaTriggerUpdate` TINYINT(1) NULL,
  `TriggerDelete` VARCHAR(45) NULL,
  `UsaTriggerDelete` TINYINT(1) NULL,
  PRIMARY KEY (`CodTabela`)
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

SET FOREIGN_KEY_CHECKS = 1;
```

Tabela 1. Detalhamento das tabelas do Dicionário de Dados.

Tabela	**Descrição**
Data_Tabela	Tabela irá armazenar as informações referente as tabelas do nosso dicionário de dados.
Campo	**Descrição**
CodTabela	Chave de identificação da tabela.
NmTabela	Nome da tabela.
NmEsquema	Esquema que a tabela pertence.
TriggerInsert	Nome da trigger para auditoria de insert.
UsatriggerInsert	Campo lógico que define a utilização da trigger de insert.
TriggerUpdate	Nome da trigger para auditoria de update.
UsaTriggerUpdate	Campo lógico que define a utilização da trigger de update.
TriggerDelete	Nome da trigger para auditoria de delete.
UsaTriggerDelete	Campo lógico que define a utilização da trigger de delete.
Tabela	**Descrição**
Data_Campo	Tabela irá armazenar as informações referente as colunas das tabelas do nosso dicionário de dados.
Campo	**Descrição**
CodCampo	Chave do campo.
NmCampo	Nome do campo.
TipoCampo	Tipo de dado que o campo armazena.
TipoChave	Tipo da chave do campo.
Auditar	Status para verificar se o campo será auditado ou não. Valido apenas para as operações de update.
CodTabela	Chave estrangeira da tabela que o campo pertence.
SeqCampo	Seqüência do campo na tabela.

Os procedimentos (Stored Procedures) a seguir são responsáveis por povoar o dicionário de dados, com os parâmetros do sistema que se pretendente auditar. Estes parâmetros serão utilizados para criação das triggers de auditoria. O status delas (se estão ativadas, ou não) será definido pelo administrador da auditoria e também ficará armazenado no dicionário de dados.

Código 3. Stored Procedures para de povoamento do Dicionário de Dados.

```
DELIMITER ||

DROP PROCEDURE IF EXISTS `Auditoria`.`SP_Data_tabela`;
CREATE DEFINER=`root`@`localhost` PROCEDURE `SP_Data_tabela`(in Pschema varchar(45))
begin
DECLARE PNmTabela VarCHAR(45);
DECLARE FINAL INT;
DECLARE CNmTabela CURSOR FOR select distinct table_name from information_schema.columns where
table_schema=Pschema;
DECLARE exit HANDLER FOR NOT FOUND SET FINAL=0;
OPEN CNmTabela;
REPEAT
FETCH CNmTabela INTO PNmTabela;
INSERT INTO Auditoria.Data_tabela
VALUES(null,PNmTabela,Pschema,concat('Aud_InS_',PNmTabela),false,concat('Aud_Upd_',PNmTabela),false,concat('A
ud_Del_',PNmTabela),false);
until(FINAL=0)
END REPEAT;
CLOSE CNmTabela;
commit;

END;||
DELIMITER ;

DELIMITER ||

DROP PROCEDURE IF EXISTS `Auditoria`.`SP_Data_Campo`;
CREATE DEFINER=`root`@`localhost` PROCEDURE `SP_Data_Campo`()
begin
DECLARE PNmColuna VarCHAR(45);
DECLARE PNmTipoColuna VarCHAR(45);
DECLARE PNmTipoChave VarCHAR(45);
DECLARE PCodTabela int;
DECLARE PSeqColuna int;
DECLARE FINAL INT;
DECLARE CNmColuna CURSOR FOR select C.Column_name,C.data_type,C.column_key,T.CodTabela,C.ordinal_position
from information_schema.columns C , Auditoria.Data_tabela T where   C.table_schema=T.NmEsquema
and C.Table_name=T.NmTabela order by T.CodTabela,C.ordinal_position;
DECLARE exit HANDLER FOR NOT FOUND SET FINAL=0;
OPEN CNmColuna;
REPEAT
FETCH CNmColuna INTO PNmColuna,PNmTipoColuna,PNmTipoChave,PCodTabela,PSeqColuna;
INSERT INTO Auditoria.Data_Campo
VALUES(null,PNmColuna,PNmTipoColuna,PNmTipoChave,true,PCodTabela,PSeqColuna);
until(FINAL=0)
END REPEAT;
CLOSE CNmColuna;

END;||
DELIMITER ;
```

No script acima foram criadas duas procedures, a primeira `SP_Data_Campo` é responsável por obter os nomes das tabelas da base de dados da aplicação que se deseja auditar, através das tabelas de sistema do SGBD, no caso específico deste artigo o MySQL, a segunda `SP_Data_Campo` varre todas as tabelas da base a ser auditada capturando as informações de cada coluna da tabela (nome, tipo, atributo chave e a posição da coluna na tabela).

Tabelas de auditoria

A estrutura a seguir permite armazenar todos os dados manipulados das tabelas auditadas de forma genérica. Para tratarmos o campo do tipo Blob foi necessário criar uma tabela a parte, com a mesma estrutura.

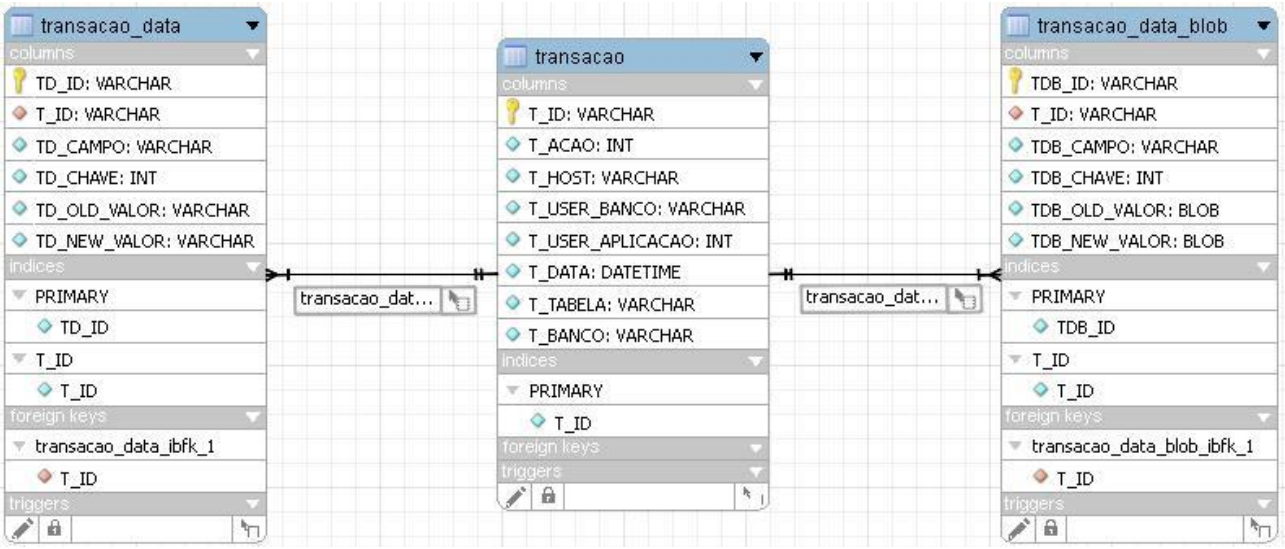

Figura 5. Estrutura de tabelas de auditoria.

Código 4. SQL para geração da estrutura das tabelas de auditoria.

```
SET FOREIGN_KEY_CHECKS = 0;

CREATE DATABASE IF NOT EXISTS `auditoria`
  CHARACTER SET latin1;

DROP TABLE IF EXISTS `auditoria`.`transacao`;
CREATE TABLE `auditoria`.`transacao` (
  `T_ID` VARCHAR(37) NOT NULL DEFAULT '',
  `T_ACAO` INT(11) NOT NULL DEFAULT '0',
  `T_HOST` VARCHAR(15) NOT NULL,
  `T_USER_BANCO` VARCHAR(30) NOT NULL,
  `T_USER_APLICACAO` INT(11) NOT NULL DEFAULT '0',
  `T_DATA` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  `T_TABELA` VARCHAR(45) NOT NULL,
  `T_BANCO` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`T_ID`)
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

DROP TABLE IF EXISTS `auditoria`.`transacao_data`;
CREATE TABLE `auditoria`.`transacao_data` (
  `TD_ID` VARCHAR(37) NOT NULL DEFAULT '',
  `T_ID` VARCHAR(37) NOT NULL DEFAULT '',
  `TD_CAMPO` VARCHAR(45) NOT NULL DEFAULT '',
  `TD_CHAVE` INT(11) NOT NULL DEFAULT '0',
  `TD_OLD_VALOR` VARCHAR(255),
  `TD_NEW_VALOR` VARCHAR(255),
  PRIMARY KEY (`TD_ID`),
  INDEX `T_ID` (`T_ID`),
  CONSTRAINT `transacao_data_ibfk_1` FOREIGN KEY `transacao_data_ibfk_1` (`T_ID`)
    REFERENCES `auditoria`.`transacao` (`T_ID`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

DROP TABLE IF EXISTS `auditoria`.`transacao_data_blob`;
CREATE TABLE `auditoria`.`transacao_data_blob` (
  `TDB_ID` VARCHAR(37) NOT NULL DEFAULT '',
  `T_ID` VARCHAR(37) NOT NULL DEFAULT '',
  `TDB_CAMPO` VARCHAR(45) NOT NULL DEFAULT '',
  `TDB_CHAVE` INT(11) NOT NULL DEFAULT '0',
  `TDB_OLD_VALOR` BLOB,
  `TDB_NEW_VALOR` BLOB,
  PRIMARY KEY (`TDB_ID`),
  INDEX `T_ID` (`T_ID`),
  CONSTRAINT `transacao_data_blob_ibfk_1` FOREIGN KEY `transacao_data_blob_ibfk_1` (`T_ID`)
```

```
    REFERENCES `auditoria`.`transacao` (`T_ID`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT
)
ENGINE = InnoDB
ROW_FORMAT = Compact
CHARACTER SET latin1 COLLATE latin1_swedish_ci;

SET FOREIGN_KEY_CHECKS = 1;
```

Tabela 2. Detalhamento das tabelas de auditoria.

Tabela	**Descrição**
Transacao	Tabela que irá armazenar as transações do banco auditado.
Campo	**Descrição**
T_ID	Chave de identificação da transação
T-ACAO	Ação que auditoria esta registrando. Insert=1; Update =2, Delete=3
T_HOST	Endereço IP da estação
T_USER_BANCO	Usuário do banco de dados que executou a ação
T_USER_APLICAÇÃO	Usuário da aplicação que executou a ação
T_DATA	Data e hora da execução da ação
T_TABELA	Nome da tabela
T_BANCO	Nome do esquema que a tabela pertence
Tabela	**Descrição**
Transacao_data	Tabela que irá armazenar os dados referente as transações do banco auditado.
Campo	**Descrição**
TD_ID	Chave de identificação da transação do dado
T_ID	Chave estrangeira da transação
TD_CAMPO	Nome do campo da tabela
TD_CHAVE	Tipo chave do campo: Primaria=1; Estrangeira=2; Nenhum=3
TD_OLD_VALOR	Valor atual do campo
TD_NEW_VALOR	Valor alterado do campo
Tabela	**Descrição**
Transacao_Data_Blob	Tem a mesma função da Tabela Transacao_data, mas armazenará campo do tipo Blob.
Campo	**Descrição**
TDB_ID	Chave de identificação da transação do dado
T_ID	Chave estrangeira da transação
TDB_CAMPO	Nome do campo da tabela
TDB_CHAVE	Tipo chave do campo: Primaria=1; Estrangeira=2; Nenhum=3
TDB_OLD_VALOR	Valor atual do campo
TDB_NEW_VALOR	Valor alterado do campo

Construindo as Triggers de auditoria

Precisamos criar as triggers que vão fazer o serviço para nós. Todo o trabalho será automatizado, mas para melhor entendimento vamos criar um exemplo manualmente. As *triggers* a seguir correspondem à auditoria da tabela de funcionário.

Quando um evento ocorrer na tabela de funcionário, a informação manipulada e sua chave de identificação serão armazenadas nas tabelas de auditoria. Com esses dados, será possível localizar o registro original e obter às informações que não foram manipuladas, sem a necessidade de armazenar todo registro. É importante lembrar que as informações do usuário de sistema devem estar armazenadas na transação, assim como o evento ocorrido, *host* da estação, usuário do SGBD, data e a hora do evento.

Código 5. SQL da Trigger de Update, responsável por registrar as alterações na tabela de funcionário.

```
DELIMITER ||
 CREATE TRIGGER Aud_Upd_funcionario BEFORE UPDATE ON FolhaPagamento.funcionario
 FOR EACH ROW BEGIN

 set @T_ID = UUID();

 INSERT INTO auditoria.transacao(T_ID,T_ACAO,T_HOST,T_USER_BANCO,T_USER_APLICACAO,T_DATA,T_TABELA,T_BANCO)
 values(@T_ID,2,@host,user(),@iduser,now(), 'funcionario' , 'FolhaPagamento');

 INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'CodFuncionario',1,OLD.CodFuncionario,NEW.CodFuncionario);

 IF OLD.NmFuncionario <> NEW.NmFuncionario then
 INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'NmFuncionario',3,OLD.NmFuncionario,NEW.NmFuncionario);
 end if ;

 IF OLD.CodDepartamento <> NEW.CodDepartamento then
 INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'CodDepartamento',3,OLD.CodDepartamento,NEW.CodDepartamento);
 end if ;

END;||
DELIMITER ;
```

Analisando o código acima podemos ver claramente que sempre vamos registrar os campos chaves, e apenas serão armazenados os dados caso acorra uma alteração.

Código 6. SQL da Trigger de Insert, responsável por registrar todas as inserções na tabela de funcionário.

```
DELIMITER ||
 CREATE TRIGGER Aud_Ins_funcionario BEFORE INSERT ON FolhaPagamento.funcionario
 FOR EACH ROW BEGIN

 set @T_ID = UUID();

 INSERT INTO auditoria.transacao(T_ID,T_ACAO,T_HOST,T_USER_BANCO,T_USER_APLICACAO,T_DATA,T_TABELA,T_BANCO)
 values(@T_ID,1,@host,user(),@iduser,now(), 'funcionario' , 'FolhaPagamento');

 INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'CodFuncionario',1,NULL,NEW.CodFuncionario);

  INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'NmFuncionario',3,NULL,NEW.NmFuncionario);


  INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'CodDepartamento',3,NULL,NEW.CodDepartamento);

END;||
DELIMITER ;
```

Código 7. SQL da Trigger de Delete, responsável por registrar todas as exclusões na tabela de funcionário.

```
DELIMITER ||
 CREATE TRIGGER Aud_Del_funcionario BEFORE DELETE ON FolhaPagamento.funcionario
 FOR EACH ROW BEGIN

 set @T_ID = UUID();

 INSERT INTO auditoria.transacao(T_ID,T_ACAO,T_HOST,T_USER_BANCO,T_USER_APLICACAO,T_DATA,T_TABELA,T_BANCO)
 values(@T_ID,3,@host,user(),@iduser,now(), 'funcionario' , 'FolhaPagamento');

 INSERT INTO auditoria.transacao_data
 (TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
 values(UUID(),@T_ID,'CodFuncionario',1,OLD.CodFuncionario,NULL);
```

```
 INSERT INTO auditoria.transacao_data
(TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
values(UUID(),@T_ID,'NmFuncionario',3,OLD.NmFuncionario,NULL);


 INSERT INTO auditoria.transacao_data
(TD_ID,T_ID,TD_CAMPO,TD_CHAVE,TD_OLD_VALOR,TD_NEW_VALOR)
values(UUID(),@T_ID,'CodDepartamento',3,OLD.CodDepartamento,NULL);

END;||
DELIMITER ;
```

Analisando o código das *triggers* de Insert e Delete acima, podemos ver que todas as informações sempre serão registradas.

Automatizando o processo

Toda a estrutura necessária para o sistema de auditoria já foi descrita nos tópicos acima. Inclusive as *triggers* responsáveis por executar as rotinas de auditoria correspondentes aos eventos disparados no banco. Porem a criação e gestão das *triggers* descritas acima para um sistema de médio grande porte seria algo inviável. Por isso vamos demonstrar uma simples aplicação para realizar a gestão da auditoria, esta aplicação automatizaria o processo de criação e/ou desativação (exclusão) das *triggers* de auditoria, de forma fácil.

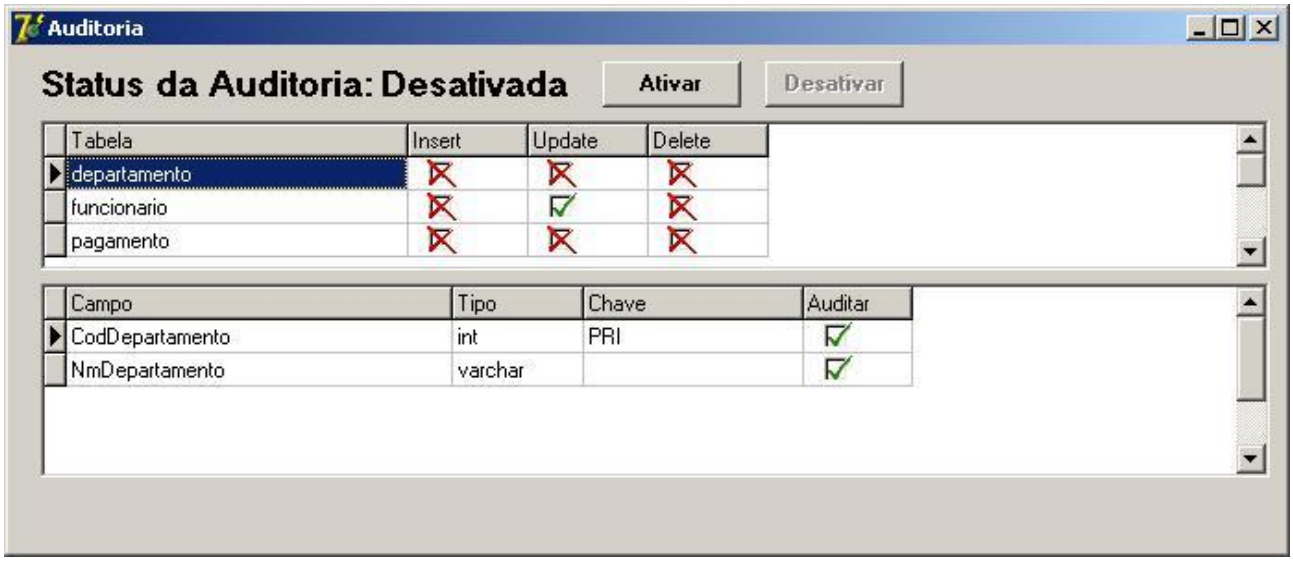

Figura 6. Tela principal de uma simples aplicação de gestão de auditoria.

A aplicação acima funcionaria da seguinte maneira:

O administrador da auditoria selecionaria as opções que ele deseja auditar (Insert, Update, Delete) para cada tabela do sistema auditado e os campos das respectivas tabelas.

Ativando a auditoria em uma determinada tabela, a aplicação criaria as *triggers* para cada evento que se deseja auditar, por sua vez, desativando a auditoria todas as *triggers* seriam excluídas. Em outros SGBDs seria possível apenas desativa-las, no entanto este recurso ainda não está disponível para o MySQL. Esta limitação de recurso não causará impacto, pois a rotina de configuração da auditoria não pode ser uma atividade rotineira, para uma auditoria consistente é necessário definir as regras de auditoria e elas nunca devem ser alteradas ou o menos possível.

Como realizar a auditoria dos dados

Para simplificar a explicação realizaremos todos os testes via console do MySQL.

As rotinas SQL a seguir, demonstram a execução de um evento nas tabelas da aplicação auditada, em seguida mostram uma consulta nas tabelas de auditoria para demonstrar os registros dos eventos executados. A última consulta de cada bloco corresponde a uma exibição mais detalhada das informações auditadas.

Código 8. SQL que declara as variáveis de sessão atribuindo valores. Esta atividade deve ser realizada pela aplicação a ser auditada, quando em um ambiente real. Os parâmetros devem ser passados quando o usuário do autenticar na aplicação auditada, registrando a assim os dados necessários para vincular os eventos de banco com o usuário que os disparou.

```
mysql> set @Iduser=1;
Query OK, 0 rows affected (0.00 sec)

mysql> set @Host='192.168.170.25';
Query OK, 0 rows affected (0.00 sec)
```

Código 9. Evento de inserção na tabela de funcionário.

```
mysql> insert into departamento values(1,'DEPTO. COMPUTAÇÃO');
Query OK, 1 row affected (0.03 sec)

mysql> insert into usuario values (1, 'Administrador', 'aluno', 'aluno');
Query OK, 1 row affected (0.03 sec)

mysql> insert into funcionario values(1,'JOAO',1);
Query OK, 1 row affected (0.05 sec)

mysql> select * from transacao;
+--------------------------------------+--------+---------------+--------------+-----------------+---------------------+------------+----------------+
| T_ID                                 | T_ACAO | T_HOST        | T_USER_BANCO | T_USER_APLICACAO | T_DATA              | T_TABELA   | T_BANCO        |
+--------------------------------------+--------+---------------+--------------+-----------------+---------------------+------------+----------------+
| 5a848571-6d5d-1029-9b50-53dd5926b416 |      1 | 192.168.170.25 | root@localhost |               1 | 2006-07-25 15:41:23 | funcionario | FolhaPagamento |
+--------------------------------------+--------+---------------+--------------+-----------------+---------------------+------------+----------------+
1 row in set (0.00 sec)

mysql> select * from transacao_data;
+--------------------------------------+--------------------------------------+---------------+----------+--------------+--------------+
| TD_ID                                | T_ID                                 | TD_CAMPO      | TD_CHAVE | TD_OLD_VALOR | TD_NEW_VALOR |
+--------------------------------------+--------------------------------------+---------------+----------+--------------+--------------+
| 5a848c7b-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | CodFuncionario |        1 | NULL         | 1            |
| 5a848fd8-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | NmFuncionario  |        3 | NULL         | JOAO         |
| 5a849194-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | CodDepartamento |        3 | NULL         | 1            |
+--------------------------------------+--------------------------------------+---------------+----------+--------------+--------------+
3 rows in set (0.00 sec)

mysql> SELECT L.NmUsuario, T.T_Data, T.T_Tabela, D.Td_campo, D.Td_New_valor
    -> FROM auditoria.transacao as T, auditoria.transacao_data as D, FolhaPagamento.Usuario as L
    -> where
    ->     T.T_Id=D.T_Id
    -> and T.T_User_Aplicacao=L.CodUsuario
    -> and T.T_acao=1;
+------------+---------------------+------------+---------------+-------------+
| NmUsuario  | T_Data              | T_Tabela   | Td_campo      | Td_New_valor |
+------------+---------------------+------------+---------------+-------------+
| Admintrador | 2006-07-25 15:41:23 | funcionario | CodFuncionario | 1           |
| Admintrador | 2006-07-25 15:41:23 | funcionario | NmFuncionario  | JOAO        |
| Admintrador | 2006-07-25 15:41:23 | funcionario | CodDepartamento | 1           |
+------------+---------------------+------------+---------------+-------------+
3 rows in set (0.00 sec)
```

Código 10. Evento de alteração na tabela de funcionário.

```
mysql> update funcionario set nmfuncionario='Joao Paulo';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from transacao;
+--------------------------------------+--------+---------------+--------------+-----------------+---------------------+------------+----------------+
| T_ID                                 | T_ACAO | T_HOST        | T_USER_BANCO | T_USER_APLICACAO | T_DATA              | T_TABELA   | T_BANCO        |
+--------------------------------------+--------+---------------+--------------+-----------------+---------------------+------------+----------------+
| 35c453e6-6d62-1029-9b50-53dd5926b416 |      2 | 192.168.170.25 | root@localhost |               1 | 2006-07-25 16:16:09 | funcionario | FolhaPagamento |
| 5a848571-6d5d-1029-9b50-53dd5926b416 |      1 | 192.168.170.25 | root@localhost |               1 | 2006-07-25 15:41:23 | funcionario | FolhaPagamento |
+--------------------------------------+--------+---------------+--------------+-----------------+---------------------+------------+----------------+
2 rows in set (0.00 sec)

mysql> select * from transacao_data;
+--------------------------------------+--------------------------------------+---------------+----------+--------------+--------------+
| TD_ID                                | T_ID                                 | TD_CAMPO      | TD_CHAVE | TD_OLD_VALOR | TD_NEW_VALOR |
+--------------------------------------+--------------------------------------+---------------+----------+--------------+--------------+
| 35c45af3-6d62-1029-9b50-53dd5926b416 | 35c453e6-6d62-1029-9b50-53dd5926b416 | CodFuncionario |        2 | 1            | 1            |
| 35c45e74-6d62-1029-9b50-53dd5926b416 | 35c453e6-6d62-1029-9b50-53dd5926b416 | NmFuncionario  |        1 | JOAO         | Joao Paulo   |
| 5a848c7b-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | CodFuncionario |        1 | NULL         | 1            |
| 5a848fd8-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | NmFuncionario  |        3 | NULL         | JOAO         |
| 5a849194-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | CodDepartamento |        3 | NULL         | 1            |
+--------------------------------------+--------------------------------------+---------------+----------+--------------+--------------+
5 rows in set (0.00 sec)

mysql> SELECT L.NmUsuario, T.T_Data, T.T_Tabela, D.Td_campo, D.Td_Old_valor,D.Td_New_valor
    -> FROM auditoria.transacao as T, auditoria.transacao_data as D, FolhaPagamento.Usuario as L
    -> where
    ->     T.T_Id=D.T_Id
    -> and T.T_User_Aplicacao=L.CodUsuario
```

```
    -> and T.T_acao=2;
+------------+---------------------+------------+---------------+-------------+-------------+
| NmUsuario  | T_Data              | T_Tabela   | Td_campo      | Td_Old_valor | Td_New_valor |
+------------+---------------------+------------+---------------+-------------+-------------+
| Admintrador | 2006-07-25 16:16:09 | funcionario | CodFuncionario | 1           | 1           |
| Admintrador | 2006-07-25 16:16:09 | funcionario | NmFuncionario  | JOAO        | Joao Paulo  |
+------------+---------------------+------------+---------------+-------------+-------------+
2 rows in set (0.00 sec)
```

Código 11. Evento de exclusão na tabela de funcionário.

```
mysql> delete from funcionario;
Query OK, 1 row affected (0.05 sec)

mysql> select * from transacao;
+--------------------------------------+-------+---------------+---------------+-----------------+---------------------+------------+---------------+
| T_ID                                 | T_ACAO | T_HOST        | T_USER_BANCO  | T_USER_APLICACAO | T_DATA              | T_TABELA   | T_BANCO       |
+--------------------------------------+-------+---------------+---------------+-----------------+---------------------+------------+---------------+
| 35c453e6-6d62-1029-9b50-53dd5926b416 |     2 | 192.168.170.25 | root@localhost |               1 | 2006-07-25 16:16:09 | funcionario | FolhaPagamento |
| 3afbe49d-6d71-1029-9b50-53dd5926b416 |     3 | 192.168.170.25 | root@localhost |               1 | 2006-07-25 18:03:40 | funcionario | FolhaPagamento |
| 5a848571-6d5d-1029-9b50-53dd5926b416 |     1 | 192.168.170.25 | root@localhost |               1 | 2006-07-25 15:41:23 | funcionario | FolhaPagamento |
+--------------------------------------+-------+---------------+---------------+-----------------+---------------------+------------+---------------+
3 rows in set (0.00 sec)

mysql> select * from transacao_data;
+--------------------------------------+--------------------------------------+---------------+----------+--------------+-------------+
| TD_ID                                | T_ID                                 | TD_CAMPO      | TD_CHAVE | TD_OLD_VALOR | TD_NEW_VALOR |
+--------------------------------------+--------------------------------------+---------------+----------+--------------+-------------+
| 35c45af3-6d62-1029-9b50-53dd5926b416 | 35c453e6-6d62-1029-9b50-53dd5926b416 | CodFuncionario |        2 | 1            | 1           |
| 35c45e74-6d62-1029-9b50-53dd5926b416 | 35c453e6-6d62-1029-9b50-53dd5926b416 | NmFuncionario  |        1 | JOAO         | Joao Paulo  |
| 3afbebba-6d71-1029-9b50-53dd5926b416 | 3afbe49d-6d71-1029-9b50-53dd5926b416 | CodFuncionario |        1 | 1            | NULL        |
| 3afbeea4-6d71-1029-9b50-53dd5926b416 | 3afbe49d-6d71-1029-9b50-53dd5926b416 | NmFuncionario  |        3 | Joao Paulo   | NULL        |
| 3afbf071-6d71-1029-9b50-53dd5926b416 | 3afbe49d-6d71-1029-9b50-53dd5926b416 | CodDepartamento |        3 | Joao Paulo   | NULL        |
| 5a848c7b-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | CodFuncionario |        1 | NULL         | 1           |
| 5a848fd8-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | NmFuncionario  |        3 | NULL         | JOAO        |
| 5a849194-6d5d-1029-9b50-53dd5926b416 | 5a848571-6d5d-1029-9b50-53dd5926b416 | CodDepartamento |        3 | NULL         | 1           |
+--------------------------------------+--------------------------------------+---------------+----------+--------------+-------------+
8 rows in set (0.00 sec)

mysql> SELECT L.NmUsuario, T.T_Data, T.T_Tabela, D.Td_campo, D.Td_Old_valor
    -> FROM auditoria.transacao as T, auditoria.transacao_data as D, FolhaPagamento.Usuario as L
    -> where
    ->      T.T_Id=D.T_Id
    -> and T.T_User_Aplicacao=L.CodUsuario
    -> and T.T_acao=3;
+------------+---------------------+------------+---------------+--------------+
| NmUsuario  | T_Data              | T_Tabela   | Td_campo      | Td_Old_valor |
+------------+---------------------+------------+---------------+--------------+
| Admintrador | 2006-07-25 18:03:40 | funcionario | CodFuncionario | 1           |
| Admintrador | 2006-07-25 18:03:40 | funcionario | NmFuncionario  | Joao Paulo  |
| Admintrador | 2006-07-25 18:03:40 | funcionario | CodDepartamento | 1           |
+------------+---------------------+------------+---------------+--------------+
3 rows in set (0.00 sec)
```

Conclusão

Implementando solução proposta neste artigo, será possível ter um eficiente sistema de auditoria, que pode ser facilmente adicionado a qualquer aplicação, uma vez que poucos parâmetros devem ser passados por ela para o banco. A utilização deste sistema também reduzirá linhas de código na aplicação em desenvolvimento assim como outras vantagens, redução no tráfego de rede e melhor desempenho da aplicação e atendendo a todos os requisitos necessários para uma auditoria eficiente.

Esperamos que este artigo tenha lhe proporcionado uma boa leitura, e seja útil no dia a dia. Até a próxima!

Referencia

[1] Wikipedia <http://pt.wikipedia.org/wiki/Auditoria>;

MySQL <http://www.mysql.com>;

MySQL Workbench <http://dev.mysql.com/downloads/workbench/>;

Revistas SQL Magazine <http://www.sqlmagazine.com.br>;

(cbueno@cientec.net) Bacharel em Sistemas de Informação pelo Centro Universitário do Leste de Minas Gerais. Atualmente trabalha como Analista de Sistemas na Cientec Consultoria e Desenvolvimentos de Sistemas. Possui 3 anos de experiência com desenvolvimento Web.

(odilon@cientec.net) é Bacharel em Sistemas de Informação formado pelo Centro Universitário do Leste de Minas Gerais, atualmente está concluindo o curso de especialização em Analise de Sistema da DCC/UFMG. Possui 10 anos de experiência na área, hoje voltados para levantamento de requisitos, analise, e desenvolvimento de software, trabalha como Analista de Sistemas na Cientec Consultoria e Desenvolvimentos de Sistemas.