Relatório Detalhado do Banco de Dados: Sistema de Monitoramento de Crimes

Diego Ribeiro Chaves
Lucas Xavier Pairé

1 Objetivo do Banco de Dados

O sistema de monitoramento de crimes foi desenvolvido para gerenciar dados sobre tipos de crimes, cidades e as ocorrências de crimes, com foco na auditoria e na rastreabilidade das operações realizadas nas tabelas do sistema, a partir do estudo de caso dos dados fornecidos pela Secretaria de Segurança Pública do Rio Grande do Sul. O banco de dados implementa recursos como **triggers** para registrar alterações em tempo real, **procedures** para automação de processos e **delete cascade** para garantir a integridade referencial entre as tabelas. O banco de dados utilizado foi o MariaDB.

2 Estrutura do Banco de Dados

O banco de dados contém as seguintes tabelas principais:

2.1 Tabela `TipoCrime`

Armazena os tipos de crimes, com a possibilidade de indicar se o crime é violento e letal (CVLI).

- **Campos**:
  - `IdTipoCrime`: Identificador único do tipo de crime.
  - `Nome`: Nome do tipo de crime.
  - `CVLI`: Indicador se o crime é considerado Violento Letal Intencional (Sim/Não).

2.2 Tabela `Cidade`

Contém as cidades onde os crimes ocorrem.

- **Campos**:
  - `IdCidade`: Identificador único da cidade.
  - `Nome`: Nome da cidade.

2.3 Tabela `Crime`

Registra as ocorrências de crimes, incluindo dados como tipo de crime, cidade e mês/ano da ocorrência.

- **Campos**:
  - `IdCrime`: Identificador único do crime.
  - `IdTipoCrime`: Relacionamento com a tabela `TipoCrime` para indicar o tipo de crime.
  - `IdCidade`: Relacionamento com a tabela `Cidade` para indicar onde o crime ocorreu.
  - `Mes`: Mês em que o crime ocorreu.
  - `Ano`: Ano em que o crime ocorreu.
  - `TotalVitimas`: Total de vítimas do crime.

2.4 Tabela `CVLI`

Relaciona crimes com dados de vítimas, utilizada para Crimes Violentos Letais Intencionais (CVLI).

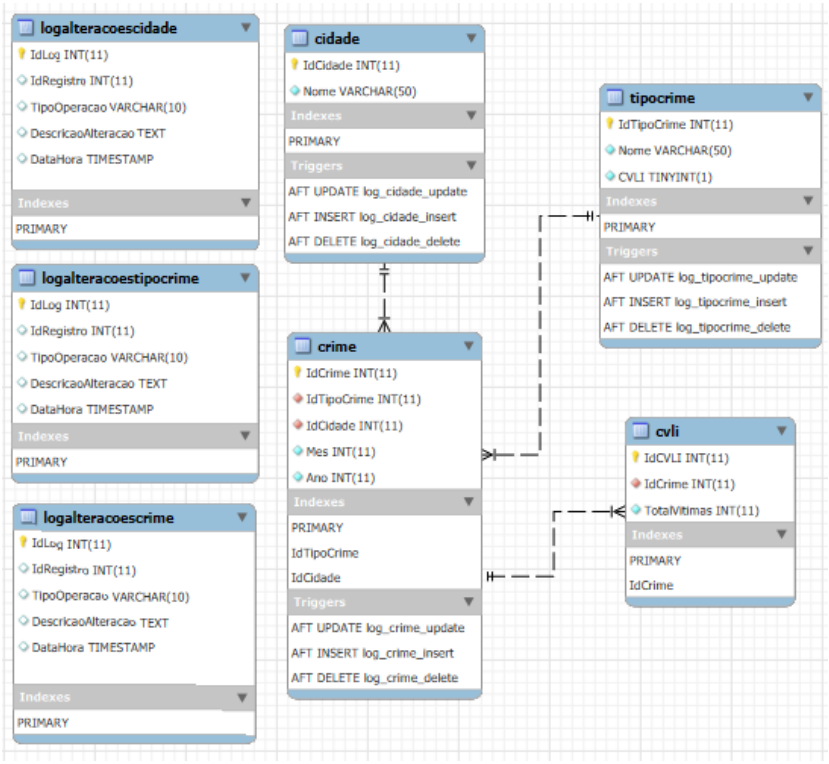
- **Campos**:
  - `IdCrime`: Relacionamento com a tabela `Crime`.
  - `TotalVitimas`: Total de vítimas para crimes classificados como CVLI.

2.5 Tabelas de Log de Auditoria

São usadas para registrar as alterações realizadas nas tabelas principais (`TipoCrime`, `Cidade`, `Crime`), garantindo a rastreabilidade das modificações.

- **Tabelas**:
  - `LogAlteracoesTipoCrime`
  - `LogAlteracoesCidade`
  - `LogAlteracoesCrime`
- **Campos (para todas as tabelas de log)**:
  - `IdLog`: Identificador único do log.
  - `IdRegistro`: Identificador do registro modificado.
  - `TipoOperacao`: Tipo de operação realizada (INSERT, UPDATE, DELETE).
  - `DescricaoAlteracao`: Descrição da modificação realizada.
  - `DataHora`: Data e hora da operação.

A estrutura pode ser observada visualmente através do diagrama de classes:



3 Implementação de Triggers

As **triggers** são usadas para monitorar e registrar automaticamente as alterações nas tabelas principais. Elas são acionadas após a execução de operações de inserção, atualização ou exclusão. Abaixo estão as triggers implementadas:

3.1 Triggers para `TipoCrime`

- `log_tipocrime_insert`: Acionada após a inserção de um novo tipo de crime na tabela `TipoCrime`.
- `log_tipocrime_update`: Acionada após a atualização de um registro na tabela `TipoCrime`.
- `log_tipocrime_delete`: Acionada após a exclusão de um tipo de crime.

3.2 Triggers para `Cidade`

- `log_cidade_insert`: Registra a inserção de uma nova cidade na tabela `Cidade`.
- `log_cidade_update`: Acionada quando um registro de cidade é alterado.
- `log_cidade_delete`: Acionada quando uma cidade é excluída da tabela `Cidade`.

3.3 Triggers para `Crime`

- `log_crime_insert`: Registra a inserção de um novo crime na tabela `Crime`.
- `log_crime_update`: Registra qualquer atualização realizada em registros de crimes.
- `log_crime_delete`: Acionada após a exclusão de um registro na tabela `Crime`.

4 Procedures

Procedures (**procedimentos armazenados**) são usados para automação de algumas operações, como a inserção de dados em tabelas de crimes ou cidades.

4.1 AdicionarCrime

Este procedimento recebe como parâmetros os dados necessários para inserir um novo crime na tabela `Crime`. O procedimento **verifica se o tipo de crime é classificado como CVLI (Crime Violento Letal Intencional)**. Se for, ele automaticamente cria uma entrada correspondente na tabela `CVLI` para registrar o número de vítimas desse crime.

4.2 AtualizarTipoCrime

Este procedimento facilita a atualização do nome de um tipo de crime na tabela `TipoCrime`. Ele realiza a atualização e chama a trigger de auditoria para registrar a modificação.

5 Constraints e Integridade Referencial

O banco de dados implementa **constraints** para garantir a integridade dos dados:

- **Chaves primárias**: As tabelas `TipoCrime`, `Cidade`, `Crime`, `CVLI` e os logs possuem chaves primárias que garantem que cada registro tenha um identificador único.
- **Chaves estrangeiras**: A tabela `Crime` possui chaves estrangeiras que garantem que cada crime esteja vinculado a um tipo de crime válido na tabela `TipoCrime` e a uma cidade válida na tabela `Cidade`.
- **Delete Cascade**: Para garantir a integridade referencial, foi implementado o **delete cascade** em algumas relações. Por exemplo, quando um tipo de crime é excluído da tabela `TipoCrime`, todos os registros associados na tabela `Crime` também são automaticamente excluídos. Além disso, quando uma cidade é excluída da tabela `Cidade`, todos os crimes ocorridos naquela cidade são excluídos.

6 Views

As **views** são utilizadas para facilitar a visualização e consulta dos dados de forma agregada. Elas oferecem uma interface simplificada para acesso a informações complexas.

- `LogsCidades`: Exibe as alterações realizadas na tabela `Cidade`.
- `LogsCrimes`: Exibe as alterações feitas na tabela `Crime`.
- `LogsTipoCrime`: Exibe todas as alterações realizadas na tabela `TipoCrime`.
- `ViewCidade`: Exibe os dados de cidades de maneira simplificada.
- `ViewCrime`: Combina informações das tabelas `Crime`, `TipoCrime`, `Cidade` e `CVLI`.
- `ViewTipoCrime`: Exibe os tipos de crimes de forma agregada, incluindo a transformação do campo `CVLI` em valores legíveis (Sim ou Não).

7 Facilidades e dificuldades no desenvolvimento

Durante o desenvolvimento deste banco de dados, algumas dificuldades e facilidades se destacaram no processo de implementação.

7.1 Dificuldades

A principal dificuldade enfrentada foi a criação de **procedures** e **triggers** que garantissem a integridade das informações do banco de dados. Especialmente, foi desafiador pensar na lógica necessária para assegurar que as operações de inserção, atualização e exclusão de registros fossem realizadas de forma consistente, sem comprometer a integridade referencial e os dados relacionados.

A complexidade surgiu quando foi necessário implementar as **triggers** de auditoria e as **procedures** que, além de realizar as operações de forma eficiente, também precisavam verificar condições específicas, como o tipo de crime em uma operação de inserção. A implementação de um `delete cascade`, por exemplo, exigiu um cuidadoso planejamento para garantir que a exclusão de um registro em uma tabela não gerasse inconsistências nas tabelas relacionadas.

Além disso, a criação de procedimentos que inserissem automaticamente dados relacionados, como na tabela `CVLI` para crimes do tipo CVLI, foi uma tarefa que exigiu raciocínio lógico aprofundado para garantir que todas as dependências fossem corretamente tratadas.

7.2 Facilidades

Em contrapartida, algumas facilidades tornaram o processo mais ágil e eficiente. A linguagem SQL se destacou por sua clareza e simplicidade. Sua sintaxe intuitiva permitiu uma implementação rápida das operações básicas de manipulação de dados, como inserção, atualização e consulta. A organização das instruções SQL e a possibilidade de trabalhar com `joins` para combinar tabelas facilitou a construção de consultas complexas.

Outro ponto positivo foi a interface do MariaDB no terminal. A instalação e o uso do MariaDB foram bastante diretos, sem complicações. A interface de linha de comando oferece uma maneira eficiente de interagir com o banco de dados, permitindo uma administração rápida e precisa, sem necessidade de ferramentas adicionais.

Essas facilidades, somadas à flexibilidade da linguagem SQL e à boa usabilidade do MariaDB, permitiram que o processo de construção do banco de dados fosse eficiente, apesar das dificuldades enfrentadas com a lógica de procedimentos e triggers.

8 Conclusão

O banco de dados foi projetado com foco na integridade, auditoria e facilitação da consulta de dados relacionados a crimes e sua distribuição geográfica. A utilização de **triggers**, **procedures**, **delete cascade** e **views** proporciona robustez e eficiência ao sistema, garantindo a rastreabilidade das alterações e a automação de processos.

Este banco de dados é adequado para ser utilizado em sistemas de monitoramento de crimes, oferecendo uma solução prática e eficiente para armazenar e consultar informações sobre ocorrências de crimes e suas características.