DESENVOLVIMENTO DE SISTEMA DESKTOP OPEN SOURCE PARA MANIPULAÇÃO DE DADOS CUM SUPORTE A MULTIPLOS BANCOS

Eduardo Provenzi Karoly*

*eduardokaroly@gmail.com

Resumo: Este artigo descreve o desenvolvimento de um sistema de apoio para sistemas de gerenciamento de banco de dados. Agilizando processos simples de manutenção de registros, manutenção da estrutura dos bancos de dados e importação e exportação de dados. A escolha do sistema se deve à falta de agilidade encontrada no gerenciamento de múltiplos bancos de dados. O objetivo é informar quais as medidas que foram tomadas para o desenvolvimento do sistema *desktop*, desde a sua concepção até a implementação, como foi possível a disponibilização *open source,* bem como os resultados obtidos. A metodologia empregada foi apresentar os ganhos da praticidade de utilização de um sistema genérico com suporta a múltiplos bancos de dados. Os resultados obtidos com o desenvolvimento do sistema apresentaram boa performance, atendendo aos objetivos definidos perante o problema. Porém, encontraram-se limitações durante o desenvolvimento. Portanto, não se pode conceber que o projeto proposto finde-se em si mesmo, e sim seja a propulsão para novas propostas e investigações.

Palavras chave: Banco de Dados. Manipulação de Dados. Multiplataforma.

Abstract: This article describes the development of a support system for database management systems. Speeding up simple processes of record keeping, database structure maintenance, and data importation and exportation. The choice of system is due to the lack of agility found in managing multiple databases. The aim is to inform the measures that have been taken to develop the desktop system, from its conception to implementation, how open source availability was possible, as well as the results obtained. The methodology used was to present the gains of the practicality of using a generic system with support to multiple databases. The results obtained with the development of the system presented good performance, attending to the objectives defined before the problem. However, limitations were encountered during development. Therefore, one can not conceive that the proposed project ends in itself, but is the propulsion for new proposals and investigations.

Keywords: Data Base. Data Management. Cross-Platform.

INTRODUÇÃO

Com o avanço das tecnologias digitais, a expansão do uso de sites e aplicações para gerenciamento de negócios cresce de forma acentuada. Consequência disso é a demanda de produção e manutenção dos desenvolvedores de sistemas, os quais dedicam tempo para a criação das aplicações, com o apoio de ferramentas disponíveis no mercado para auxiliar em seu trabalho. Porém, apesar de existirem ferramentas disponíveis, há a dificuldade de encontrar alguma que ofereça uma solução que permita a manutenção e migração de dados de forma simples e prática.

Diante desta contextualização apresenta-se o problema de pesquisa que é como criar um sistema com suporte a múltiplos tipos de bancos de dados, simples e de prática utilização, visando facilitar a manipulação de dados e estruturas de bancos.

Com o uso de banco de dados, há a necessidade de criar e alterar a estrutura dos dados frequentemente, bem como seus registros. Porém torna-se uma tarefa trabalhosa se não houver um Sistema Gerenciador De Banco De Dados (SGDB), e mesmo assim, não se deve perder de vista a praticidade de utilização, que pode tornar algumas tarefas repetitivas e cansativas. Há dificuldades em se trabalhar com mais de um tipo de banco de dados ao mesmo tempo, às vezes, é necessário a utilização de um programa para gerenciamento individual de cada um deles.

Diante disso o objetivo geral do projeto é desenvolver um sistema *desktop open source* para manipulação de dados com suporte a múltiplos bancos de dados. Tendo como objetivos específicos apresentar simplicidade e praticidade de utilização, suportar múltiplos bancos de dados, permitir importar e exportar dados e estrutura.

O artigo está estruturado em tópicos, o próximo aborda as ferramentas utilizadas para o desenvolvimento do sistema, e na sequência será visto a metodologia utilizada, bem como o desenvolvimento prático do sistema e as considerações finais.

FERRAMENTAS UTILIZADAS PARA O DESENVOLVIMENTO DO SISTEMA

Atualmente, há significativo número de bancos de dados disponíveis para uso, de acordo com o site *db-enginers (2016)* são mais de 300. Geralmente, cada um possuindo suas peculiaridades e um sistema de gerenciamento de banco de dados próprio. Essa diversidade dificulta o trabalho de desenvolvedores que utilizam mais de uma linguagem, visto que terão que utilizar mais de um sistema de gerenciamento e ter uma curva de aprendizado maior para compreender as linguagens de cada banco. Em linhas gerais, os sistemas de gerenciamento têm em vista a disponibilização de uma solução que contempla tarefas e configurações complexas, possuindo validações e limitações no seu uso, assim, tendo como consequência a perda de praticidade em suas utilizações.

Outro fator impactado é a replicação e migração de dados, segundo Morris (2016), migração de dados é a seleção, extração, transformação e movimento permanente dos dados apropriados com qualidade para o lugar certo, no momento certo e a desativação de bancos de dados legados. Portanto, migração de dados envolve processos complexos e estudos sobre cada situação. E apesar de existirem sistemas que oferecem uma solução completa de Extract, Transform and Load (ETL), em casos simples de migrações como uma base pequena, somente de uma tabela ou replicação de alguns dados, passa a ter expressivas configurações que se tornam desnecessárias. A seguir falaremos sobre a definição de bancos de dados e de sistemas de gerenciamento de banco de dados (SGBD).

Segundo Korth (2016), um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados.

Já um sistema de gerenciamento de banco de dados (SGBD) é um software que possui recursos capazes de manipular as informações do banco de dados e interagir com o usuário. Exemplos de SGBDs são: Oracle, SQL Server, DB2, PostgreSQL, MySQL, Access, Firebird, entre outros. Conceitua-se um sistema de banco de dados como o conjunto de quatro componentes básicos: dados, hardware, software e usuários. Date (2016) conceituou que “sistema de bancos de dados pode

ser considerado como uma sala de arquivos eletrônica". Os objetivos de um sistema de banco de dados são o de isolar o usuário dos detalhes internos do banco de dados (promover a abstração de dados) e promover a independência dos dados em relação às aplicações, ou seja, tornar independente da aplicação, a estratégia de acesso e a forma de armazenamento.

Após apresentado os conceitos básicos da base do problema, aborda-se as ferramentas utilizadas no desenvolvimento da solução, os padrões de projeto utilizados e ferramentas de controle de versão.

Para o desenvolvimento do software, foi utilizado Lazarus 1.6.0, pois conforme o site lazarus-ide.org (2016), é um Ambiente de Desenvolvimento Integrado (do inglês *Integrated Development Environment, IDE*) baseada em Pascal que utiliza o compilador Free Pascal, é open source e multi-plataforma. Suporta a linguagem de programação Object Pascal, com o framework padrão Lazarus Component Library (LCL) possuindo uma grande variedade de componentes prontos para uso, proporcionando um rápido desenvolvimento de aplicações.

Em conjunto com a IDE Lazarus foi utilizado a extensão FpSpreadSheet, que adiciona componentes multi-plataforama para importação e exportação de arquivos nos formatos Excel, HTML, XML, CSV, ODS entre outros. Conforme FreePascal Wiki (2016):

> A biblioteca fpSpreadsheet oferece uma maneira prática de criar e ler planilhas em vários formatos. A biblioteca foi escrita de uma maneira muito flexível, capaz de ser estendida para suportar qualquer formato facilmente.

Com o uso dessa extensão foi poupado tempo de desenvolvimento, utilizando uma solução robusta e sanando qualquer necessidade da aplicação em relação a escrita e leitura de arquivos.

Após a definição das ferramentas e linguagem utilizadas, foram definidos padrões de projeto para o desenvolvimento eficiente do sistema, pois segundo o site Macoratti.net (2016), são soluções eficientes já comprovadas para a resolução de problemas em projetos de software. Segue abaixo os padrões utilizados:

- Factory Method, mantem as instâncias das classes isoladas das que as solicitam;

- DAO, permite separar as regras de negócio das regras de acesso ao banco de dados. Implementado junto com o Factory Method, vai permitir separar as regras de acesso de cada banco de dados em uma única classe, tornando a manutenção e implementação mais simples;
- Singleton, deve garantir que exista somente uma única instância de uma classe, mantendo um ponto de acesso global ao seu objeto.

Seguindo os padrões será fácil para outra pessoa compreender o funcionamento do sistema e realizar manutenção ou melhorias. Para finalizar esse tópico veremos abaixo o sistema de controle de versão e repositório dos dados.

Para gerenciar o repositório de dados foi utilizado o TortoiseGit versão 2.2.0.0 (Git versão 2.9.2). O software fornece uma interface agradável e fácil de configurar. Além disso, é possível submeter, reverter, atualizar, remover e fazer o merge dos arquivos, mostrando as alterações dos mesmos.

O projeto será hospedado em um repositório do GitHub, e será clonado pelo TortoiseGit no computador local. Permitindo um fácil gerenciamento do projeto e com mais segurança para evitar perdas. No próximo tópico será apresentada a metodologia utilizada no projeto.

METODOLOGIA

Após a identificação do problema, foram feitas pesquisas para a averiguação do mesmo. Com os resultados destas pesquisas, foi definido as ferramentas que serão utilizadas, maneiras de desenvolver o sistema e de como desenvolver o sistema da melhor maneira possível.

Para melhorar e expandir a produtividade de desenvolvedores de sistemas, foi desenvolvido um software, no qual os usuários podem realizar alterações em múltiplos bancos de dados, a fim agilizar as tarefas da melhor maneira possível.

As ferramentas utilizadas para o desenvolvimento do projeto foram escolhidas de acordo com pesquisas feitas sobre softwares para desenvolvimento desktop que possuem agilidade, segurança, performance, multi-plataforma. Visando a disponibilização do sistema como open source, serão utilizadas somente

ferramentas gratuitas, que não possuem qualquer tipo de custo para sua utilização. As ferramentas são:

A principal ferramenta utilizada é a IDE Lazarus versão 1.6.0, conforme explicado na secção 3.2, é uma IDE *open source* e que dispõem de uma solução completa para o objetivo do projeto. Em conjunto da IDE será utilizado o componente de terceiro FpSpreadSheet versão 1.6.2, o qual também é open source e livre de custos.

Quanto ao repositório de dados, citado na secção 3.5, o GitHub (versão 2.2.0.0) apresenta uma solução livre porém pública para outras pessoas acessarem, ou para ser privado é necessário pagar pelo serviço. Como a proposta é disponibilizar o sistema como open source não há a necessidade de pagar por um serviço privado, então será utilizado o serviço público e sem custos. Quanto ao TortoiseGit (Git versão 2.9.2) foi utilizado na máquina local também é uma ferramenta livre e open source, dispensando qualquer custo adicional.

Com isso não ocorreu nenhum custo com as ferramentas utilizadas. Tendo somente conhecimento e tempo dedicados a pesquisa e desenvolvimento do projeto, tornando totalmente viável a disponibilização do sistema como open source e de livre utilização para qualquer pessoa.

Para o início do desenvolvimento, foram elencados os requisitos, ou seja, foi realizada a análise detalhada das regras de negócio e das necessidades do sistema, verificando o que é esperado que o sistema faça para garantir que o desenvolvimento ocorra da forma correta e de acordo com o que foi descrito nos requisitos. Conforme Fabio Gomes Rocha (2013, p.1), “parte das falhas de projetos é devida a falhas no levantamento de requisitos. ”.

Após os requisitos verificados, o desenvolvimento foi realizado de acordo com os padrões de projeto Singleton, Factory e DAO, garantindo que o sistema seja feito de forma organizada e clara.

Segundo Rissino (2016), a migração de dados é um processo de alto grau de complexidade e risco, já que modifica um dos mais importantes ativos da instituição — os dados. Para se mitigar possíveis danos e inconsistências na base de dados, é necessário o desenvolvimento de um projeto que contemple, de forma clara e

objetiva, todas as ações e etapas da migração de dados. Muitas das ações e etapas necessárias podem ocorrer antes da migração e após o referido processo.

Portanto o projeto trata a migração de dados de uma forma simplificada, suportando apenas operações básicas de importação e exportação, partindo do princípio de que o usuário possui pleno conhecimento sobre suas ações. Assim, faz-se desnecessárias validações complexas e específicas para cada banco de dados, além de tornar a utilização do sistema mais dinâmica. Desta forma, no tópico seguinte trata-se do desenvolvimento prático do sistema.

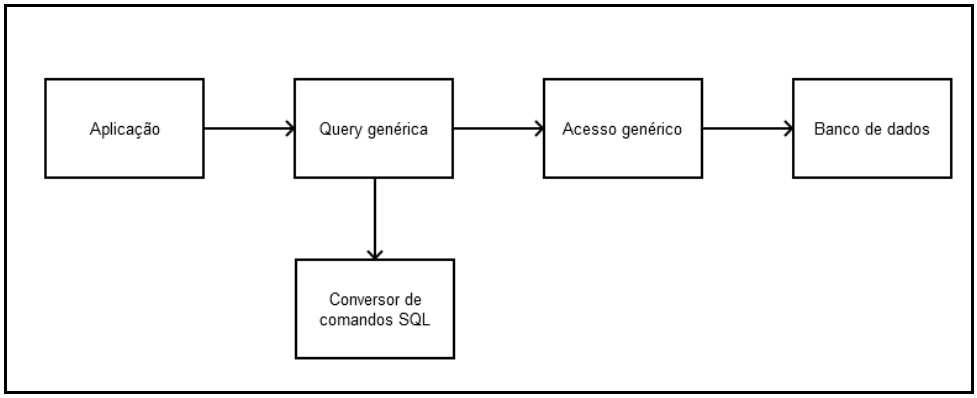
DESENVOLVIMENTO PRÁTICO DO SISTEMA

O sistema SQLManager permite acessar diversos tipos de banco de dados relacionais nativamente, sendo necessário apenas configurar a conexão para poder realizar operações na base conectada. O foco do sistema é permitir realizar operações básicas de forma prática, em qualquer banco de dados conectado. Em conjunto com uma interface simples, que vai facilitar o seu uso. A seguir são detalhados os objetivos específicos do sistema.

a) Para atender o **objetivo de simplicidade e praticidade** do sistema foi decidido utilizar uma única tela, que por meio de herança vai implementar as camadas de negócios referentes à manipulação de registros e outra a manipulação de estrutura do banco de dados. Desta forma, segue o mesmo padrão de utilização para todas as funcionalidades oferecidas. As alterações no banco são feitas pela própria *grid* de apresentação dos dados, na tela principal de cada funcionalidade, sendo necessário somente salvar as modificações após finalizadas, descartando a utilização de qualquer tela adicional ou demais validações. Outra forma de realizar as alterações pé por meio da importação de um arquivo, que vai adicionar, alterar ou excluir registros conforme o especificado.

b) Quanto ao **objetivo de suporte a múltiplos bancos de dados**, o sistema foi dividido em duas camadas distintas, que vão se complementar para atender a este requisito: Uma camada para acesso genérico e outra para adaptar os comandos SQL para cada tipo de bancos de dados. A Figura 1 representa o fluxo do sistema para se comunicar com o banco de dados.

Figura 1 - Fluxo de conexão



Conforme a figura apresentada, a aplicação encaminha o SQL para a query genérica, que automaticamente converte para a linguagem certa do banco de dados conectado. Por fim, quando a aplicação mandar executar o comando SQL, a query utiliza a camada de conexão genérica para executar o comando no banco.

b.1) A camada de acesso genérico, o acesso é feito por meio de componentes integrados ao framework padrão do Lazarus. Foi possível utilizar uma conexão padrão para os bancos mais conhecidos ou uma conexão Open Database Connectivity (ODBC), em que o usuário pode se conectar com qualquer banco de dados que possua suporte ao ODBC.

b.2) Já sobre a camada de conversão de SQL, para manter a compatibilidade dos comandos de um banco para outro foi criado uma camada para converter o script para o banco necessário. Conforme o sistema vai criando os comandos para serem executados, o mesmo vai chamar uma classe para converter os comandos para o banco desejado para depois armazenar o comando e então executar por meio do acesso genérico.

c) A **exportação de dados funciona de forma simples**, seguindo o **objetivo de praticidade, vai exportar** exatamente o que estiver sendo exibido na *grid* principal de cada tela. Desta forma funciona tanto para exportação de dados como de estrutura. Após selecionar a opção para exportar será solicitado para escolher o local de destino, nome e tipo do arquivo para então exportar os dados conforme as configurações informadas. O processo de exportação vai ocorrer por meio de *threads* para permitir que o sistema continue sendo executado. Quando a

thread terminar a exportação, envia uma mensagens customizada para o sistema principal que vai exibir um pop-up na tela informando que os dados foram exportados com sucesso.

d) O padrão definido para a o **objetivo de importação** é que cada arquivo deve ser referente a somente uma tabela, e será aceito somente arquivos de texto(.txt) ou excel (.xlsx). A primeira linha do arquivo deve conter os nomes dos campos, as demais linhas devem possuir os valores à serem importados, seguindo exatamente a mesma ordem dos campos. Todas as importações também vão salvar logs de execução, em um arquivo pré-definido pelo sistema no local de execução da aplicação principal. Porém, a importação ocorre de uma forma para dados e outra para estrutura.

Quanto a importação de dados, esta ocorre somente com um arquivo de cada vez, referente a uma tabela. A primeira etapa da importação verifica se todos os campos definidos na primeira linha estão coerentes com a estrutura atual da base. Caso haja divergências será apontado o erro no log de importação e o processo será encerrado. Se os campos informados estiverem corretos, o sistema prossegue com a importação, que a partir desse ponto vai percorrer cada linha do arquivo identificando os valores de cada coluna e montando o script de insert, e aplicando o script na base de dados.

Já em relação a importação de estruturas vai possuir um layout padrão para os arquivos, conforme é demonstrado no Quadro 1:

Quadro 1 - Layout padrão para os arquivos

Coluna	**Descrição**
Nome	Nome do campo
Tipo	Determina o tipo do campo
Collation	Determina o collation do campo
Nullable	Determina se o campo vai aceitar valores nulos
PK	Determina se o campo é chave primária
Default	Determina o valor padrão do campo

Com base nesse layout o sistema vai percorrer cada linha do arquivo, identificar todos os valores e gerar o comando adequado conforme a base atual. Se

o campo já existir é criado o comando de *update*, senão cria o comando para adicionar o campo e se tiver somente o nome informado e as demais colunas vazias o campo será deletado.

Em todas as importações serão adicionados logs em cada etapa do processo, e para cada linha, informando-se que está sendo analisada, os valores identificados, o script montado e caso ocorra erro na execução salva no log o erro apresentado.

Com todos os objetivos específicos detalhados, agora aborda-se sobre a arquitetura do sistema. A qual será subdividida em quatro camadas, conforme está demonstrado na figura a seguir:

Figura 2 - Arquitetura do sistema

Com base nesta figura, a seguir são descritas as quatro camadas desenvolvidas na aplicação:

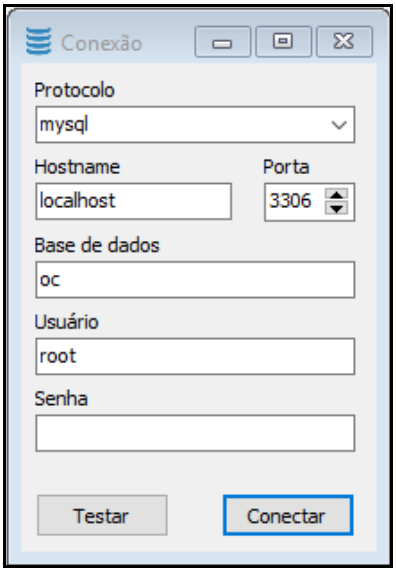
- A camada de apresentação é responsável pela iteração com o usuário, em que os dados serão apresentados e vai receber as ações desejadas;
- A camada de negócios é responsável por interpretar as ações recebidas, ajustar configurações necessárias para atender às

solicitações, exportar dados e encaminhar os scripts necessários para a camada de adaptação de SQL;

- A camada de adaptação de SQL vai possuir duas *query's* genéricas, uma para consulta de dados e outra que será usada para execução dos demais scripts. A *query* de consulta vai ser usada para exibir a listagem principal de cada tela, de forma que não sofrerá interferência de outras execuções realizadas pelo sistema. Enquanto a outra *query* servirá para executar scripts como inserção, update e delete de registros e alterações da estrutura do banco;
- A camada de acesso genérico servirá para centralizar o acesso à base de dados em um único lugar, permitindo conectar com qualquer banco utilizando o mesmo mecanismo.

Tendo as especificações técnicas do sistema finalizadas, a seguir é descrito o funcionamento do sistema em alto nível. O sistema funciona de forma simples, pois, ao iniciar será solicitado para conectar com uma base de dados, conforme a Figura 3. Após estar corretamente conectado o usuário pode utilizar a tela de manipulação de dados e outra de manipulação de estrutura.

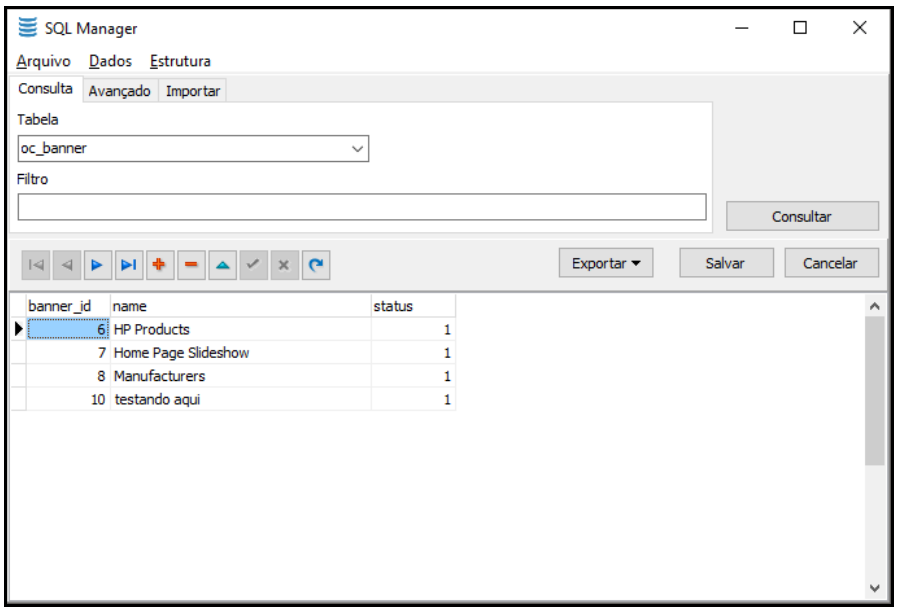
Figura 3 - Tela de conexão



Na figura a seguir, verifica-se a tela de manipulação de dados, que vai oferecer suporte somente à manipulação de registros do banco de dados. A

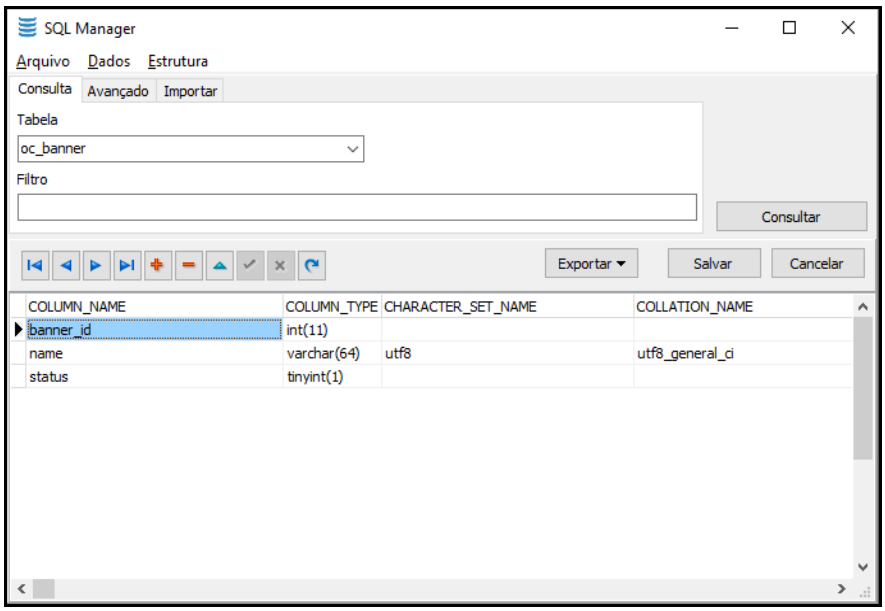
apresentação da tela é intuitiva, tendo um painel superior para as consultas ou importação, e uma grid que ocupa a maior parte da ela para a apresentação dos dados. As alterações dos registros podem ser feitas na própria grid, bastando alterar o que desejar e salvar ou cancelar as alterações, ou as alterações podem ser importadas de um arquivo com registros para a base, que vai criar os registros ou alterar caso a chave primária já exista. A exportação de registros também ocorre por meio da grid de resultados, pois vai exportar exatamente o que estiver sendo exibido para o local e tipo de arquivo selecionado.

Figura 4 - Tela de manipulação de dados



Já na figura seguinte, apresenta-se outra tela de manipulação de estrutura, que funciona de uma forma bastante semelhante a tela de manipulação de dados, figura anterior. Os componentes das duas telas propostas são exatamente iguais, seguindo o padrão de utilização, porém os comandos internos do sistema são diferentes para suportar somente a estrutura do banco de dados.

Figura 5 - Tela de manipulação de estrutura



Desta forma, pode-se ver que o sistema não possui complexidade, dispõe somente de funções simples e não realiza operações avançadas nos bancos de dados. Porém é possível realizar alterações simples de forma prática e intuitiva, bastando alterar o que desejar na *grid* principal de cada tela. Outra opção também é exportar os dados para um arquivo, alterar as informações no próprio arquivo e depois importar o mesmo para aplicar as alterações desejadas no banco. Desta forma, no próximo tópico apresenta-se as considerações finais do projeto.

CONSIDERAÇÕES FINAIS

Após realizar pesquisas sobre o problema identificado, e como seria possível desenvolver uma aplicação para gerenciamento de bancos de dados, simples, prática e com suporte a múltiplos bancos de dados. Foram verificados os objetivos específicos do projeto para criar uma solução que seja simples, com prática utilização, com suporte a múltiplos bancos e permitir exportar e importar dados e estruturas de tabelas no banco de dados desejado.

Verificou-se que a solução atendeu aos resultados esperados com boa performance, porém foram encontrados limitações na execução de tarefas complexas nos bancos de dados, como: suporte para *procedures*, *functions*, *triggers*, gerenciamento de permissões, entre outros. Para suportar as limitações

encontradas serão necessários estudos aprofundados em cada linguagem de banco de dados e uma implementação avançada para cada solução. Desta forma, o SQLManager não substitui os SGDBs próprios para cada linguagem, sendo útil como uma ferramenta de apoio para tarefas simples e corriqueiras.

Por fim, permanece como futuras implementações as limitações encontradas, para tornar-se uma solução completa e, se possível, descartar o uso de outros SGDBs. Haja vista que não se pode conceber que o projeto proposto finde-se em si mesmo, e sim seja a propulsão para novas propostas e investigações.

REFERÊNCIAS BIBLIOGRÁFICAS

Livro ***Database System Concepts***, SILBERSCHATZ, Abraham, KORTH, Henry, S. Sudarshan, sexta edição. Disponível em:< http://www.jordomseiling.org/Database%20System%20Concepts%206th%20edition.pdf>. Acesso em novembro. 2016.

Livro ***Practical Data Migration***, MORRIS, Johny, segunda edição. Disponível em: <goo.gl/kw0aZl>. Acesso em novembro. 2016.

Componente de terceiro FPSpreadsheet. Disponível em:< http://wiki.freepascal.org/FPSpreadsheet>. Acesso em Setembro. 2016.

Conceitos Fundamentais de Banco de Dados. Disponível em:< http://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649>. Acesso em Setembro. 2016.

Lazarus e Free Pascal wiki. Disponível em: <http://wiki.freepascal.org/>. Acesso em Setembro. 2016.

Lazarus IDE. Disponível em: <https://en.wikipedia.org/wiki/Lazarus_(IDE)>. Acesso em Setembro. 2016.

Migração de banco de dados, RISSINO, Silvia, COÊLHO, Darlene, OLIVEIRA, David. Disponível em: < http://www.devmedia.com.br/migracao-de-banco-de-dados/37428>. Acesso em novembro. 2016.

Padrão de projeto Singleton. Disponível em:< https://pt.wikipedia.org/wiki/Singleton>. Acesso em Setembro. 2016.

Padrões de Projeto. Disponível em:<http://www.devmedia.com.br/conheca-os-padroes-de-projeto/957>. Acesso em Setembro. 2016.