

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ana Livia Nogueira

**Desenvolvimento de uma Ferramenta Baseada
na Linguagem *Query by Example* para Criação
e Execução de Consultas SQL**

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ana Livia Nogueira

**Desenvolvimento de uma Ferramenta Baseada na
Linguagem *Query by Example* para Criação e Execução
de Consultas SQL**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Orientador: Prof. Humberto Luiz Razente

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2018

Lista de ilustrações

Figura 1 – Exemplo 1 Consulta em QBE.	9
Figura 2 – Resultado da consulta em QBE para a Figura 1.	10
Figura 3 – Exemplo 2 Consulta em QBE.	10
Figura 4 – Resultado da consulta em QBE para a Figura 3.	11
Figura 5 – Exemplo 3 Consulta em QBE e resultados.	11
Figura 6 – Interface do <i>IDERA's Rapid SQL</i>	13
Figura 7 – Interface do <i>GestureQuery</i>	14
Figura 8 – <i>Oracle SQL Developer</i> : tabelas selecionadas para realização da consulta.	15
Figura 9 – <i>Oracle SQL Developer</i> : janela com a exibição das colunas selecionadas.	15
Figura 10 – <i>Oracle SQL Developer</i> : cláusulas selecionadas e resultado da consulta.	16
Figura 11 – <i>Oracle SQL Developer</i> : SQL gerado na aba Planilha.	16
Figura 12 – <i>Oracle SQL Developer</i> : Consulta SQL escrita na aba Planilha.	17
Figura 13 – <i>Oracle SQL Developer</i> : Consulta gerada pelo <i>Query Builder</i>	17
Figura 14 – PGAdmin: <i>Graphical Query Builder</i>	18
Figura 15 – Arquivos DLLs referenciados no diretório do projeto.	20
Figura 16 – Código SQL: seleção dos nomes das tabelas do banco de dados.	20
Figura 17 – <i>ListBox</i> com os nomes das tabelas.	21
Figura 18 – Código SQL: lista atributos de uma tabela.	21
Figura 19 – Painel com as tabelas escolhidas pelo usuário.	22
Figura 20 – Código SQL: chaves estrangeiras de uma tabela.	23
Figura 21 – Tabela com indicação de chave estrangeira em seu atributo.	23
Figura 22 – Ação de marcar o primeiro atributo para selecionar e adicionar todos.	24
Figura 23 – Ação de marcar um atributo na tabela.	25
Figura 24 – <i>ListBox</i> para armazenar os atributos escolhidos e lista de operadores para as condições.	26
Figura 25 – Painel com os componentes responsáveis pela elaboração e montagem da consulta.	27
Figura 26 – Form com a exibição dos resultados da consulta.	28
Figura 27 – Painel com as tabelas escolhidas e organizadas.	29
Figura 28 – Painel com os atributos escolhido e exibidos em uma lista.	30
Figura 29 – Painel com os atributos escolhido e a cláusula WHERE montada.	30
Figura 30 – Formulário com a consulta SQL montada e seus resultados.	31

Sumário

1	INTRODUÇÃO	5
1.1	Motivação	5
1.2	Objetivos	5
1.3	Contribuições	6
1.4	Organização da Monografia	7
2	FUNDAMENTOS E TRABALHOS CORRELATOS	8
2.1	Fundamentos	8
2.2	Trabalhos Correlatos	12
3	DESENVOLVIMENTO	19
3.1	Apresentação e Delimitações da Pesquisa	19
3.2	Escolhas e Preparação do Ambiente	19
3.3	Primeiros Passos	20
3.4	Construção do Cenário	21
3.5	Montagem e Retorno da Consulta	26
3.6	Estudo de Caso	29
4	CONSIDERAÇÕES FINAIS	32
4.1	Conclusões	32
4.2	Trabalhos Futuros	32
	REFERÊNCIAS	34

Resumo

Esta monografia apresenta os conceitos e um protótipo de uma ferramenta para simplificar a elaboração de consultas SQL para o banco de dados PostgreSQL, bem como a execução das mesmas, por meio da técnica *Query by Example (QBE)*. A técnica permite ao usuário/programador criar consultas complexas por meio de uma interface visual simples e intuitiva. A ferramenta também pode ser utilizada para o ensino da linguagem SQL pois a partir da consulta visual uma consulta na linguagem SQL é apresentada. A ferramenta foi desenvolvida na linguagem Visual Basic.net como uma aplicação para área de trabalho na plataforma Microsoft Windows. Para fazer a conexão entre a aplicação VB.net e o banco de Dados PostgreSQL foi utilizado o provedor de dados Npgsql. A ferramenta é capaz de realizar a conexão com o banco de dados, recuperar as tabelas disponíveis e seus atributos para que o usuário possa elaborar visualmente consultas simples que podem envolver mais de uma tabela. Finalmente, a ferramenta exibe a consulta em SQL e os resultados em forma de tabela. Este estudo servirá de base para que em trabalhos futuros mais funcionalidades possam ser adicionadas a ferramenta, entre elas as funções de agregação.

1 Introdução

1.1 Motivação

Com o advento dos Sistemas de Gerenciamento de Bancos de Dados Relacionais (SGBDR) no final dos anos 70 e sua ampla utilização até atualmente, a *Structured Query Language (SQL)* tornou-se padrão para realização de consultas (EISENBERG et al., 2004). O uso de ferramentas para auxílio aos profissionais da área de Tecnologia de Informação em suas tarefas vem sendo cada vez mais de extrema importância, pois permite aumentar a produtividade de seus trabalhos e diminuir erros humanos. Em especial, para o projeto e desenvolvimento de aplicações de banco de dados há diversas ferramentas para modelagem, manutenção de modelos e engenharia reversa de banco dados.

Em se tratando de escrita de consultas que serão embarcadas nas interfaces dos sistemas, em geral os programadores escrevem manualmente os códigos SQL com base nas informações disponíveis no dicionário de dados e utilizam uma ferramenta/interface para executar as mesmas. Com isso o programador deve ter um conhecimento além do básico (de acordo com sua necessidade) sobre SQL para que ele possa elaborar e estruturar as consultas de forma correta e então o sistema retornar os resultados desejados. Porém nem todos os usuários de bancos de dados são programadores experientes e familiarizados com o assunto. Além disso, para a realização de consultas *ad-hoc* em que se deseja uma resposta rápida, a projeção de múltiplas tabelas e respectiva indicação de todas as condições de junção em conjunto com os outros requisitos da consulta pode ser complicado quando ainda é necessário formular a consulta de maneira estruturada segundo a linguagem SQL. Segundo Elmasri e Navathe (2011) interfaces mais simples vêm sendo desenvolvidas para sistemas de banco de dados, pois há uma necessidade grande na recuperação de dados de uma forma fácil e eficiente.

1.2 Objetivos

O projeto objetivou o desenvolvimento de uma ferramenta (aplicação de área de trabalho) de auxílio à escrita de consultas SQL para o SGBDR PostgreSQL, em que será necessário um conhecimento básico sobre a linguagem para que o usuário construa sua consulta visualmente por meio de operações “apontar e clicar” (*point and click*) de maneira intuitiva e simples. Como saída a ferramenta irá gerar um comando SQL com base nessas indicações do usuário, retornando os resultados em forma de tabela para a respectiva consulta escrita em SQL. Estas indicações serão feitas através de ilustrações de tabelas na qual o usuário poderá marcar seus atributos pela ação de marcação (*check*) e adicionar

condições na cláusula WHERE, para que então o sistema possa gerar automaticamente a consulta SQL e retornar o resultado obtido. Já existem ferramentas no mercado que oferecem estas funcionalidades, porém quase não se encontram artigos atuais sobre esse tipo de método de consulta. Portanto o objetivo do projeto focou nos conceitos fundamentais da técnica *Query By Example* para a construção de uma nova ferramenta e a descrição de todas as etapas necessárias para este desenvolvimento. Dentro desta implementação um dos desafios foi o estudo de recuperação de metadados do PostgreSQL, o uso do provedor de dados NpgSQL e o aprendizado da linguagem Microsoft Visual Basic.net.

1.3 Contribuições

Este trabalho teve como objetivo contribuir para o crescimento e evolução das tecnologias em vários sentidos, principalmente para o meio acadêmico. O estudo e desenvolvimento está focado na criação de um protótipo de uma ferramenta para facilitar consultas em banco de dados. A principal contribuição está em apresentar o estado da arte relacionado à técnica *Query by Example*.

Esta pesquisa teve o intuito de agregar conhecimento para o meio acadêmico da área de Ciência da Computação. A principal referência é o trabalho de Zloof (1975). Além do ferramental teórico, são apresentadas ferramentas comerciais e de código livre que permitem a realização dessas consultas. Assim, as ferramentas existentes juntamente às suas documentações são apresentadas. Contudo os artigos sobre este tema são bem antigos, exemplificando alguns métodos que hoje já estão ultrapassados e as ferramentas do mercado que usam métodos mais avançados não possuem um artigo com todos os detalhes de construção, apenas a documentação de seu funcionamento. Tentando preencher estas lacunas, este trabalho tem o intuito de desenvolver uma ferramenta para consultas SQL, utilizando tecnologias atuais e documentando todo o processo de criação e funcionamento detalhadamente.

O contexto atual da área de Tecnologia da Informação, no qual as ferramentas existentes evoluem constantemente, contribuiu para o desenvolvimento deste trabalho. Com isto decidiu-se desenvolver a ferramenta baseada na linguagem *Query by Example* em uma nova implementação para que seja possível o entendimento completo do funcionamento da mesma, para que então trabalhos futuros possam acrescentar funcionalidades ainda não existentes em outras ferramentas comerciais.

No contexto acadêmico a ferramenta tem o intuito de auxiliar estudantes da graduação durante seu aprendizado com a linguagem SQL, por meio do aprendizado da linguagem baseado na interação visual e respectiva apresentação do texto das consultas. Então a intenção é que a ferramenta sirva também como um primeiro contato com a linguagem de modo intuitivo e didático.

1.4 Organização da Monografia

O restante deste trabalho está organizado da seguinte maneira. O Capítulo 2 está dividido em “Fundamentos” e “Trabalhos Correlatos”, onde os Fundamentos procuram exemplificar em que a pesquisa está sendo baseada juntamente à metodologia usada e nos Trabalhos Correlatos são apresentados ferramentas relacionadas. O Capítulo 3 apresenta o Desenvolvimento do protótipo dividido em 6 subseções “Apresentação e Delimitações da Pesquisa”, “Escolhas e Preparação do Ambiente”, “Primeiros Passos”, “Construção do Cenário”, “Montagem e Retorno da Consulta” e “Estudo de Caso”. O Capítulo 4 apresenta as “Conclusões” e “Trabalhos Futuros”.

2 Fundamentos e Trabalhos Correlatos

A especificação de uma linguagem de consultas baseadas em exemplos foi apresentada pela primeira vez em (ZLOOF, 1975). O trabalho apresentou a linguagem denominada *Query by Example (QBE)*, onde basicamente o usuário formula sua consulta preenchendo as linhas das tabelas apropriadas, com um exemplo de possível resposta.

A base definida pelo trabalho de Zloof permite a criação de ferramentas visuais que visam a especificação de consultas SQL complexas por meio da criação de interações visuais que permitam a seleção de tabelas, projeção de atributos, criação de condições de junções e definição de agrupamentos. No entanto, na época em que foi proposta, os ambientes gráficos não estavam disponíveis, de modo que o trabalho descreve uma implementação em ambiente apenas textual. Nas últimas décadas, com a evolução dos dispositivos, surgiram diversos tipos de interface de desenvolvimento.

2.1 Fundamentos

A base principal para o desenvolvimento da pesquisa será a linguagem de consultas gráficas chamada *Query by Example*, que foi desenvolvida na IBM Research. Para complementar também será utilizado como referência artigos construídos a partir da *QBE* e ferramentas já existentes. A *QBE* (*Query by Example*) se difere da SQL pois tem uma proposta mais simplificada e não usa um estilo linear, o usuário apenas preenche linhas da tabela com elementos constantes e elementos exemplos sem a necessidade de especificar uma sintaxe fixa. Segundo Elmasri e Navathe (2011), a *QBE* foi considerada a primeira linguagem de banco de dados de relação visual amigável ao usuário.

A metodologia de pesquisa é do tipo exploratória, pois está focada no estudo de artigos sobre a linguagem *Query by Example* e o funcionamento de ferramentas baseadas na mesma, o que dá embasamento para a construção de um protótipo de uma ferramenta de criação de consultas SQL. Os resultados são apresentados de maneira qualitativa, descrevendo a complexidade do problema e do desenvolvimento da pesquisa.

Banco de dados é uma parte essencial em quase todos os sistemas de informações utilizados no mundo. Lidar com suas operações pode chegar ao nível alto de complexidade, principalmente para quem não é da área de Tecnologia da Informação. A linguagem *Query by Example* tem o intuito de simplificar estas operações em banco de dados, mas sem perder a eficiência das consultas. Por isto nota-se que o número de sistemas baseados na *QBE* está em crescimento.

Em 1975 Zloof escreveu um artigo sobre esta linguagem. Segundo Zloof (1975), a

Query by Example é uma tentativa de auxiliar profissionais que tem pouca prática com o mundo computacional e o usuário precisa estudar até o nível de complexidade que fará suas consultas, sem a necessidade de aprofundar todo o escopo da linguagem.

Na *QBE* o usuário formula uma consulta preenchendo uma ou mais linhas da tabela com um exemplo de possível resposta. Ainda que seja uma grande classe de consultas “simples” o usuário precisará apenas distinguir dois elementos. O elemento exemplo que deve ser indicado por um sublinhado e será sempre variável, portanto o valor colocado não tem importância. E o elemento constante que não mudará durante o processamento da consulta. Também existe o prefixo “P” que é inserido antes de qualquer dado que será impresso, ou seja, se enquadra no SELECT da linguagem SQL. Com estas propriedades é possível formular consultas de diversos níveis de complexidade. Funções de agregação como AVG, SUM, COUNT, MAX e MIN também podem ser definidas na QBE, porém caso seja necessário que essas funções sejam aplicadas a todos os valores é necessário o uso da função ALL.

Figura 1 – Exemplo 1 Consulta em QBE.

TYPE	ITEM	COLOR	SIZE
	P. <u>PEN</u>	RED	

Fonte: Zloof (1975).

O exemplo apresentado na Figura 1 representa uma consulta simples, onde o objetivo é retornar os elementos de cor *RED* (vermelho). A coluna “ITEM” contém a função “P” que será utilizado para imprimir e “PEN” o elemento exemplo que se encontra sublinhado, portanto, a consulta irá imprimir um elemento que pode ser do tipo “PEN”, contudo é apenas um exemplo de possível resposta já que o elemento exemplo pode ser variável. A coluna “COLOR” contém um elemento que não se encontra sublinhado, portanto, ele é um elemento constante indicando que qualquer que seja a resposta desta consulta, os itens retornados devem ter a cor “RED”, referenciando a linguagem SQL, este elemento é o usado para especificar as cláusulas WHERE. A estrutura da Figura 1, retorna o resultado exibido na Figura 2, retornando os elementos “LIPSTICK” e “PENCIL”, portanto estes são os itens que contém a cor “RED”(vermelho).

Figura 2 – Resultado da consulta em QBE para a Figura 1.

ITEM
LIPSTICK
PENCIL

Fonte: Zloof (1975).

Um outro exemplo para mostrar o funcionamento da *Query by Example* será uma consulta que retornará as cores de um determinado item, como mostrado na Figura 3. Desta vez a função “P” encontra-se na coluna “COLOR”, acompanhada do elemento exemplo “BLACK” indicando que as cores podem variar, já na coluna “ITEM” contém “INK” que é o elemento constante, por tanto a consulta montada retornará as cores do item “INK”.

Figura 3 – Exemplo 2 Consulta em QBE.

TYPE	ITEM	COLOR	SIZE
	INK	P. <u>BLACK</u>	

Fonte: Zloof (1975).

O resultado da consulta é exibido na Figura 4, na qual as cores encontradas para o elemento “INK” foram “GREEN” e “BLUE”. Com isto a *Query by Example* mostra que tanto a função “P” quanto os elementos exemplo e constante, podem alternar de lugar (coluna) de acordo com a necessidade na hora de formular a consulta.

Figura 4 – Resultado da consulta em QBE para a Figura 3.

ANS:

COLOR
GREEN
BLUE

Fonte: Zloof (1975).

A *Query by Example* também permite que consultas envolvam mais de uma tabela, tratando isto de maneira simplista como apresentado na Figura 5, na qual o objetivo da consulta é encontrar os departamentos que vendem um item fornecido pelo fornecedor “PARKER”. A coluna “DEPT” da primeira tabela, contém a função “P” juntamente ao elemento exemplo “TOY” indicando que o departamento pode ser por exemplo “TOY” e na coluna “ITEM” o elemento “ROD” que também é um elemento exemplo. Na segunda tabela, a coluna “ITEM” também contém o elemento exemplo “ROD” para poder filtrar as linhas iguais nas duas tabelas indicando que se o item é vendido por um departamento, este mesmo item deve ser fornecido pela Parker e na coluna “SUPPLIER” o elemento constante “PARKER”. O resultado da consulta é exibido em *ANS* da Figura 5.

Figura 5 – Exemplo 3 Consulta em QBE e resultados.

SALES	DEPT	ITEM
	P. TOY	ROD

SUPPLY	ITEM	SUPPLIER
	ROD	PARKER

ANS:

DEPT
HOUSEHOLD
TOY
STATIONARY
HARDWARE

Fonte: Zloof (1975).

Segundo Elmasri e Navathe (2011), os testes com usuários têm mostrado que a QBE é mais fácil de aprender do que a SQL, principalmente para não especialistas. Isto é possível pois contém um quadro de referência pré-estabelecido através do exemplo adicionado, além de um grau de liberdade maior para elaboração das consultas já que a

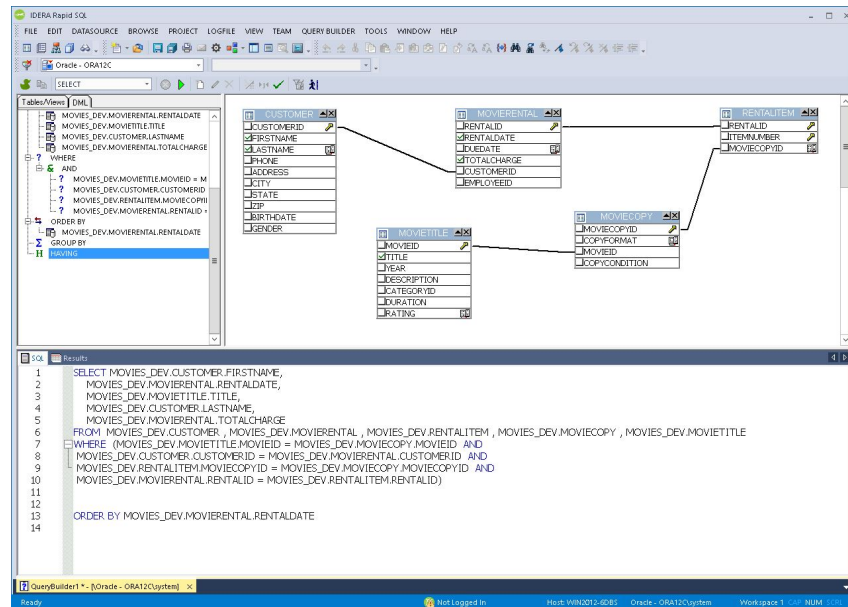
sequência em que são preenchidas (ligadas) as tabelas são irrelevantes, não restringindo o processo de pensamento do usuário/programador.

Para a criação do protótipo da ferramenta utilizou-se o ambiente de desenvolvimento *Visual Studio* ([Microsoft \(2018b\)](#)) juntamente a linguagem de programação *Visual Basic.net*. A ferramenta foi desenvolvida para o SGBDR *PostgreSQL 9.6* ([PostgreSQL \(2018\)](#)), com a versão 3 do seu software gráfico *pgAdmin*. Para realizar a conexão entre o *PostgreSQL* e o *Visual Basic.net* será usado o provedor de dados *Npgsql* ([Npgsql \(2018\)](#)).

O *PostgreSQL* é um sistema de banco de dados objeto-relacional gratuito e de código aberto. *Npgsql* é um provedor de dados para o *PostgreSQL*, que permite programas escritos em C#, *Visual Basic.net*, F# acessem o servidor de banco de dados *PostgreSQL*. É implementado em código 100% C#, é gratuito e de código aberto. E o *Visual Studio* é um ambiente de desenvolvimento integrado da [Microsoft \(2018b\)](#), que contém versões gratuitas e pagas, para o desenvolvimento da ferramenta foi utilizada a versão gratuita *Community*.

2.2 Trabalhos Correlatos

A partir do trabalho de Zloof várias ferramentas de programação para bancos de dados relacionais de diversos fabricantes incorporaram o recurso de *Query by Example*, entre elas o Borland Database Desktop que acompanhava as ferramentas *Borland Delphi* e *C++Builder* ([Borland \(2018\)](#)) e o Microsoft Access ([Microsoft \(2018a\)](#)). Uma ferramenta interessante é a *IDERA's Rapid SQL* ([IDERA's, 2018](#)) que apresenta um ambiente de desenvolvimento para construção de consultas integradas no qual o desenvolvedor pode construir instruções SQL complexas com a facilidade de apontar e clicar usando o *Visual Query Builder*. O ambiente fornece uma interface gráfica que simplifica a criação de consultas SQL, como apresentado na Figura 6.

Figura 6 – Interface do *IDERA's Rapid SQL*.

Fonte: IDERA's (2018).

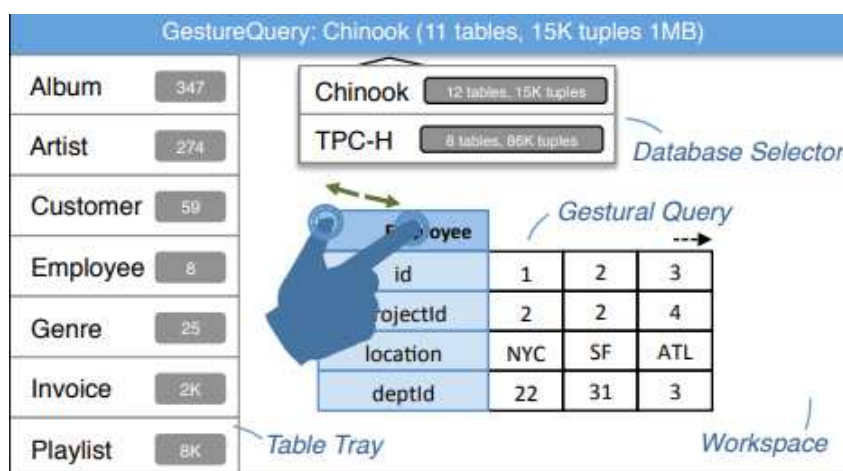
Um outro sistema criado com intuito de facilitar a interação com o usuário final foi o Microsoft Access 2000 para Windows (Microsoft (2018a)), que é um programa para gestão de dados com a intenção de ter uma estrutura mais simples, para que o usuário não lide com a parte complexa de gestão de um SGBDR. Por exemplo, para a criação de tabelas em um banco de dados é necessário apenas um duplo clique na opção “Criar tabela na vista de estrutura” e depois introduzir seus dados em uma janela que dispõe de opções onde o usuário pode escolher o tipo de dados de um atributo e se ele pertence à chave primária da tabela por meio de interações de clique do mouse. Para consultas o processo é similar, também cria-se uma consulta apenas com clique do mouse. Além disso o Microsoft Access contém a opção “lista de consultas” para que possa reutilizá-las quando desejar.

O *GestureQuery* criado por (JIANG; MANDEL; NANDI, 2013) propõem uma técnica de QBE para dispositivos que não têm teclado como entrada principal, entre eles dispositivos de computação móvel e interativa como *smartphones*, *tablets* e dispositivos de realidade virtual, com foco em um sistema de interação com o banco de dados sem teclado. Nesse trabalho são apresentados os desafios no acesso a dados gestuais, desde a criação de uma linguagem intuitiva e completa até o processamento com precisão a intenção da consulta. Porém há grandes desafios no acesso a dados gestuais, desde a criação de uma linguagem intuitiva e completa até o processamento com precisão no momento de elaboração da consulta.

O *GestureQuery* é um frontend que permite aos usuários criarem suas consultas

através de uma interface *multitouch*. Esta interface é dividida em três partes, o cabeçalho, que exibe as informações do banco de dados e dá a opção do usuário de escolher outro banco caso necessário. Há também a bandeja que irá exibir a lista de tabelas disponíveis no banco de dados selecionado. E por último o espaço de trabalho que é usado para formular uma consulta. Para começar a construir uma consulta o usuário arrasta uma tabela da bandeja para o espaço de trabalho, esta tabela é exibida em retângulos um com o seu respectivo nome e os demais com seus atributos então as consultas podem ser formuladas utilizando uma série de gestos multitouch. A interface do *GestureQuery* pode ser observada na Figura 7.

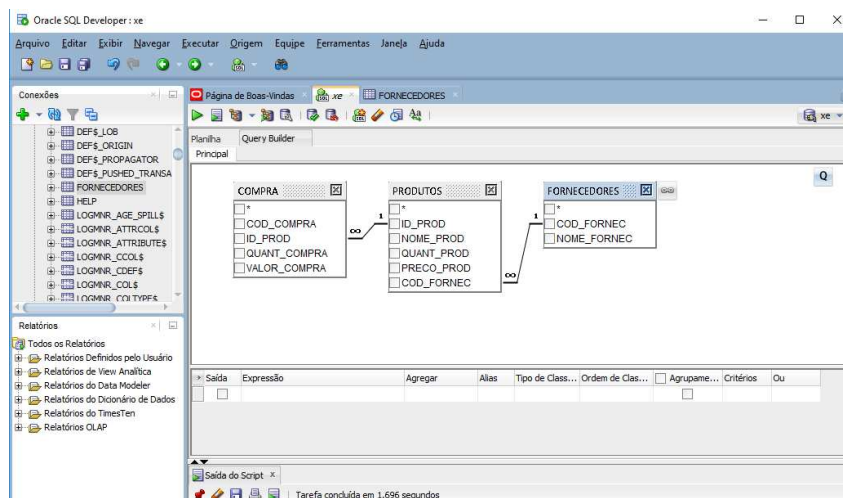
Figura 7 – Interface do *GestureQuery*.



Fonte: Jiang, Mandel e Nandi (2013).

A ferramenta *Oracle SQL Developer* (Oracle SQL (2018)) é uma ferramenta de manipulação para o SGBDR *Oracle Database* (Oracle (2018)). Na qual dispõe de uma interface para QBE (Oracle Query Builder, 2018) e permite lidar com as consultas de maneira interativa, usando o recurso de apontar e clicar, arrastar e soltar. A manipulação das consultas é feita de maneira simples, por meio do recurso de arrastar tabelas localizadas no canto esquerdo e soltar na janela chamada *Query Builder*. A ferramenta automaticamente sugere as junções por igualdade (equijunções) por meio dos relacionamentos entre as tabelas, definidos pelos pares de chaves primárias e estrangeiras, apresentados graficamente como curvas que ligam os atributos de relacionamento, como apresentado na Figura 8.

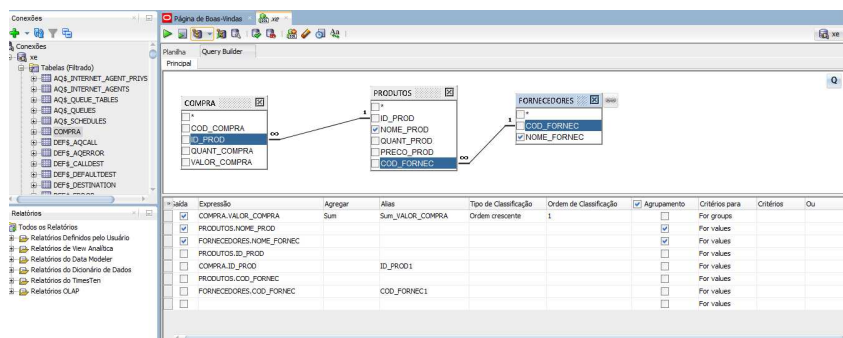
Figura 8 – Oracle SQL Developer: tabelas selecionadas para realização da consulta.



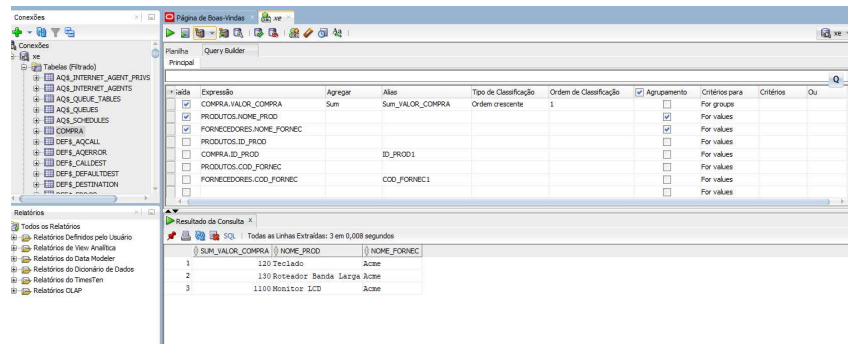
Fonte: Autoria própria (2018).

As tabelas exibem seus atributos (colunas) e há uma opção de marcação, permitindo a escolha dos atributos que serão projetados/exibidos pela consulta. A medida em que os atributos são selecionados, eles são apresentados em uma tabela que permite a definição de parâmetros como a escolha de uma função de agregação (MIN, MAX, SUM, AVG, COUNT, entre outras disponíveis), atribuição de apelido para o atributo, ordenação no resultado e definição de critérios de seleção, bem como sua ordenação na projeção/exibição do resultado da consulta. A Figura 9 apresenta um exemplo de criação de consulta e a Figura 10 o seus resultados.

Figura 9 – Oracle SQL Developer: janela com a exibição das colunas selecionadas.

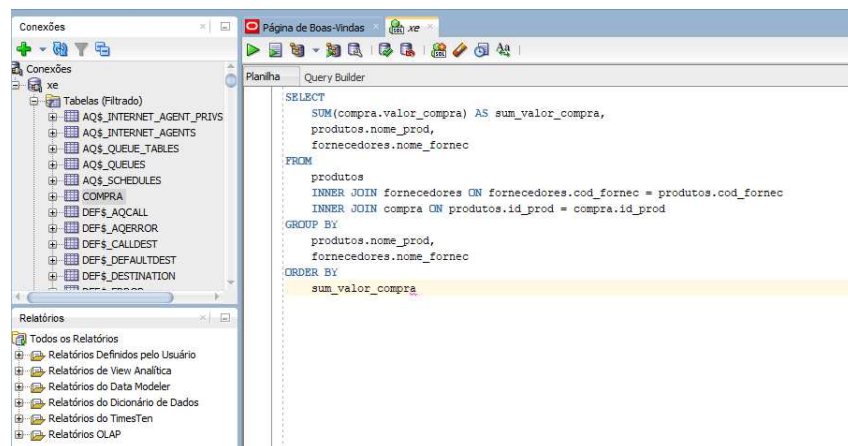


Fonte: Autoria própria (2018).

Figura 10 – *Oracle SQL Developer*: cláusulas selecionadas e resultado da consulta.

Fonte: Autoria própria (2018).

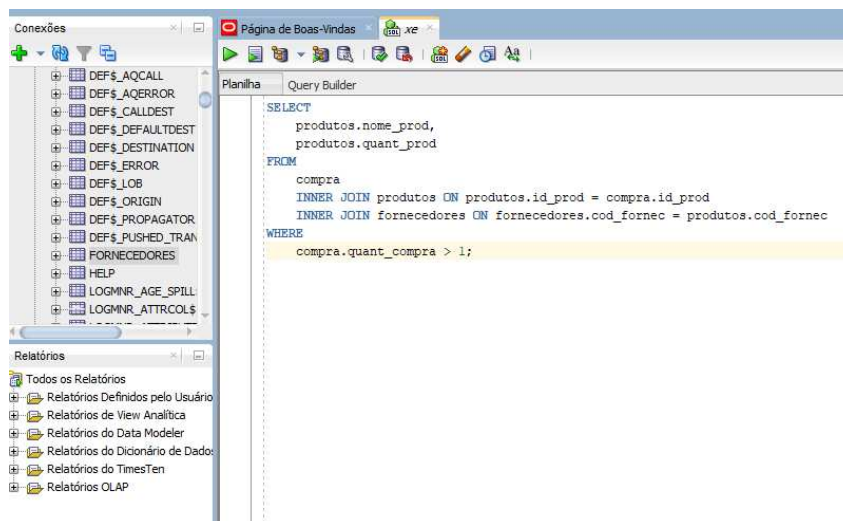
Caso o usuário deseje visualizar o comando SQL resultante da QBE, o botão com o ícone “SQL” gera o código em uma nova janela, como apresentado na Figura 11.

Figura 11 – *Oracle SQL Developer*: SQL gerado na aba Planilha.

Fonte: Autoria própria (2018).

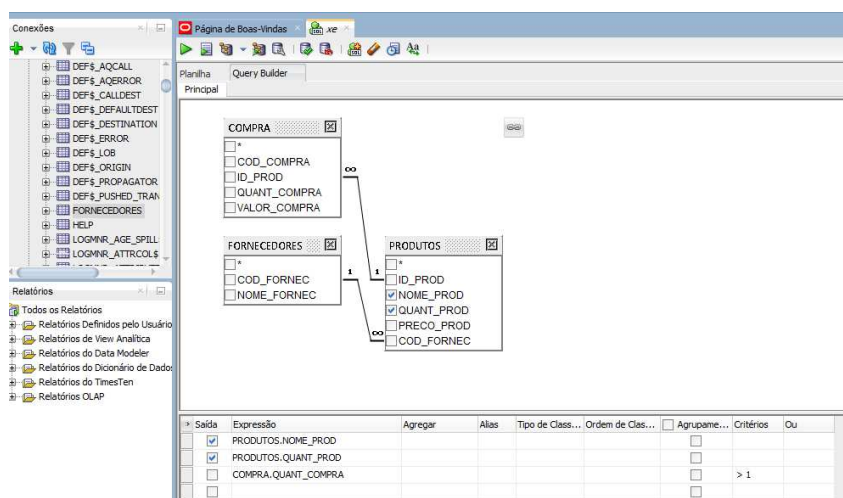
A ferramenta também permite que o processo inverso seja realizado, ou seja, o usuário pode especificar uma consulta SQL de modo textual e então acessar a janela *Query Builder* para visualizar o esboço da consulta em tabelas e suas ligações. A Figura 12 apresenta a especificação textual de uma consulta SQL e a Figura 13 a respectiva consulta em QBE.

Figura 12 – Oracle SQL Developer: Consulta SQL escrita na aba Planilha.



Fonte: Autoria própria (2018).

Figura 13 – Oracle SQL Developer: Consulta gerada pelo Query Builder.



Fonte: Autoria própria (2018).

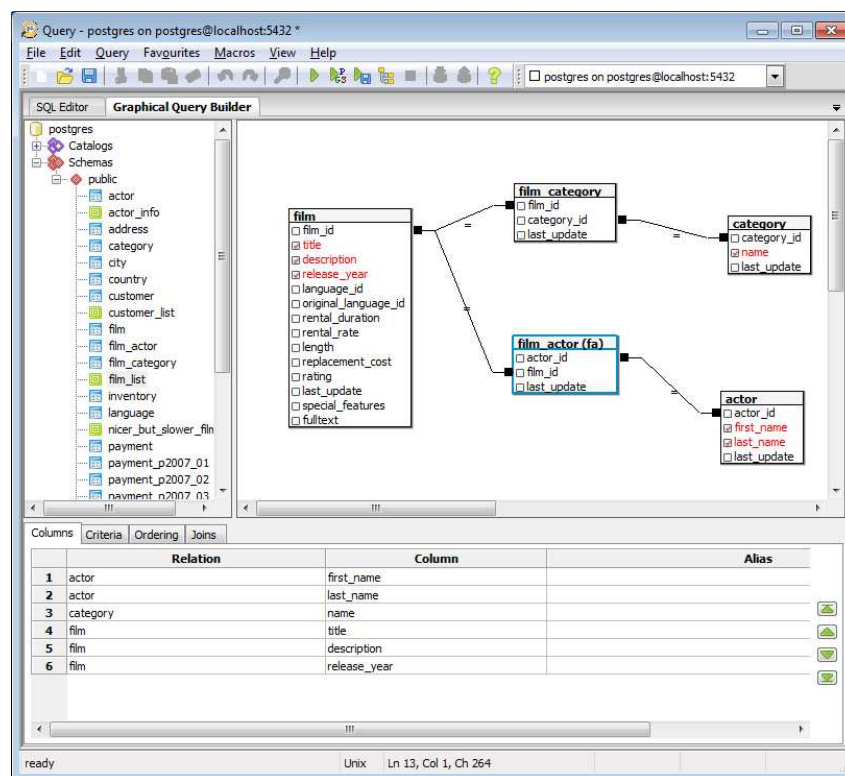
Como a consulta está representada em ambas janelas *Query Builder* e “Planilha”, é possível fazer alterações em uma e verificar a atualização da mesma na outra janela, como colocar um critério para uma coluna no *Query Builder* e vê-lo sendo representado no SQL automaticamente. Essa interação facilita no processo obtendo resultados de maneira mais fácil e intuitiva.

O PgAdmin (2018) desenvolveu *Query Tool* (ferramenta de consulta), para escritas de consultas SQL no PgAdmin, onde uma parte é o *Graphical Query Builder (GQB)* que permite a criação de consultas SQL de maneira visual e mais simples. Existem três partes principais do GQB. O navegador de relações que permite selecionar catálogos,

classes e *views* que podem ser incluídas nas consultas. Um conjunto de guias que permitem especificar critérios de seleção, formato de saída e classificação. E uma tela onde é possível desenhar os relacionamentos entre suas tabelas. Pode-se ajustar o tamanho relativo de cada seção arrastando a junção (ou faixa) entre os painéis.

Para adicionar relações (tabelas) à consulta, é necessário clicar duas vezes no navegador de relações ou arrastá-las do navegador para a tela. Pode-se organizar as relações na tela selecionando o nome da relação com o mouse e movendo a mesma para a posição desejada. Para a criação das junções basta arrastar uma coluna de uma relação para outra e então uma linha será desenhada entre elas, para indicar a união. Para selecionar as colunas que aparecerão nos resultados da consulta, é necessário marcar as colunas desejadas dentro da relação na tela. Cada coluna selecionada será adicionada à lista na guia Colunas, na qual pode-se ajustar a ordem. A Figura 14 mostra a interface do sistema.

Figura 14 – PGAdmin: *Graphical Query Builder*.



Fonte: PgAdmin (2018).

3 Desenvolvimento

3.1 Apresentação e Delimitações da Pesquisa

Esta monografia, intitulada “Desenvolvimento de uma Ferramenta Baseada na Linguagem *Query by Example* para Criação e Execução de Consultas SQL”, apresenta um protótipo de ferramenta para construção de consultas SQL de interface amigável para facilitar o uso por programadores e pessoas com pouca familiaridade em banco de dados, buscando minimizar o tempo gasto em escrita de consultas a SGBDR. Atualmente, a manipulação de dados tornou-se algo rotineiro em quase todos os tipos de estabelecimentos e a falta de conhecimento sobre o assunto vem criando complexidade em tarefas simples nos SGBDR. Existem muitas ferramentas para manipulação de dados em SGBDR no mercado com estas funcionalidades, tanto proprietárias quanto de código aberto. Porém tecnologias existentes devem ser reimplementadas para permitir o avanço tecnológico, pois é este um dos caminhos para que soluções mais otimizadas sejam encontradas. O foco do trabalho é demonstrar o funcionamento da técnica QBE e os desafios para se construir uma ferramenta de programação para manipulação de dados em SGBDR. Adicionalmente, a monografia tem como objetivo contribuir com a documentação detalhada do processo de construção da ferramenta.

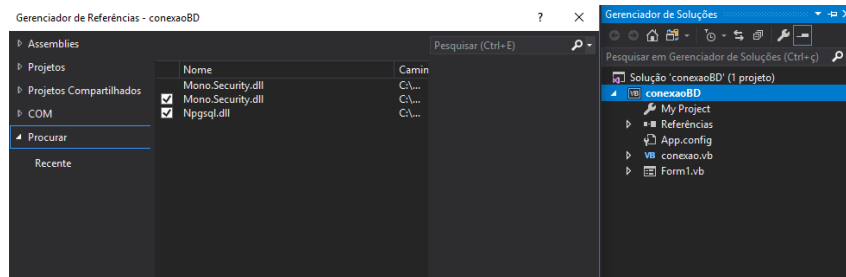
3.2 Escolhas e Preparação do Ambiente

Após o levantamento do estado da arte e das ferramentas existentes foi necessário escolher quais tecnologias seriam usadas para o desenvolvimento. Ao observar a ferramenta *Building Queries Visually* ([Oracle Query Builder, 2018](#)) na qual a interface e as funcionalidades se aproximam do objetivo final deste trabalho, foi possível notar que o processamento de construção da consulta poderia ser em partes baseados em eventos, sendo o primeiro tópico a influenciar a escolha da linguagem de programação criada pela *Microsoft, Visual Basic.net*, que é uma linguagem baseada em paradigma de orientação a objetos e que herdou a característica de RAD (Desenvolvimento Rápido de Aplicativos – *Rapid Application Development*). Assim, o projeto foi desenvolvido com o uso da plataforma de desenvolvimento *Microsoft Visual Studio* e do arcabouço *.net*.

Foi necessário estabelecer a conexão do *Visual Basic.net* com o SGBDR *PostgreSQL*, para isto foi escolhido o provedor de dados *Npgsql* feito para o acesso ao PostgreSQL. Uma outra opção seria a utilização de um driver *ODBC* (*Open Database Connectivity*). Optou-se pelo *Npgsql* para eliminar uma camada de software, ou seja, para que a aplicação pudesse fazer acesso ao SGBDR diretamente. Para tanto, foram

adicionadas as bibliotecas de vínculo dinâmico (DLL) do *NpgSQL* no diretório do projeto na *Solution Explorer* do *Visual Studio*, como apresentado na Figura 15.

Figura 15 – Arquivos DLLs referenciados no diretório do projeto.



Fonte: Autoria própria (2018).

Para a criação da classe de conexão, inicia-se pela importação do *NpgSQL* e do *System.Data*. A *string* de conexão contém nome do banco de dados, endereço de rede, porta, usuário e senha. Assim, a criação do objeto do tipo *NpgsqlConnection* com a devida *string* de conexão efetuará a conexão com o SGBDR, permitindo a execução de sentenças SQL por meio de um *CommandText*.

3.3 Primeiros Passos

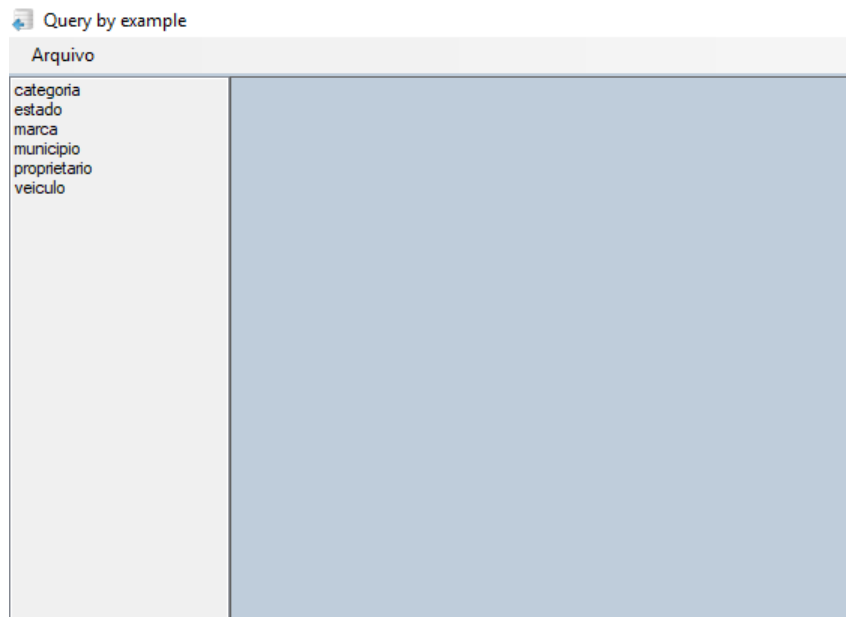
Com o ambiente preparado a etapa seguinte será a recuperação de dados, começando pelas tabelas presentes no banco de dados. Para a recuperação das tabelas, será usado o conceito de metadados que são informações sobre os dados do SGBDR. Toda a consulta será realizada através do catálogo [Documentação PostgreSQL 8.0.0 \(2018\)](#) (O catálogo do sistema é formado por tabelas de sistemas e visões internas, por onde pode ser feito o acesso direto e específico aos metadados). A primeira consulta terá como objetivo a seleção dos nomes das tabelas, conforme apresentado na Figura 16.

Figura 16 – Código SQL: seleção dos nomes das tabelas do banco de dados.

```
SELECT c.relname
FROM pg_class c, pg_namespace n
WHERE c.relnamespace = n.oid
ORDER BY relname
```

Fonte: Autoria própria (2018).

A execução da consulta permite a apresentação das tabelas em um componente do tipo lista presente no formulário que representará a interface da aplicação conforme apresentado na Figura 17. O evento de duplo clique em uma tabela desta lista dispara um procedimento que insere a tabela na consulta QBE.

Figura 17 – *ListBox* com os nomes das tabelas.

Fonte: Autoria própria (2018).

3.4 Construção do Cenário

Dentro do procedimento disparado será feita a verificação do tamanho da lista para garantir que há itens, pois a cada ação de duplo clique o item que sofreu a ação será removido da lista para ser inserido na QBE. O conteúdo do item escolhido será adicionado em uma lista chamada `listaNomeTabelas` e removido da lista que guardava os nomes das tabelas. Esta `listaNomeTabelas` será percorrida e para cada elemento encontrado será executado a consulta apresentada na Figura 18 que retornará os atributos deste item (tabela).

Figura 18 – Código SQL: lista atributos de uma tabela.

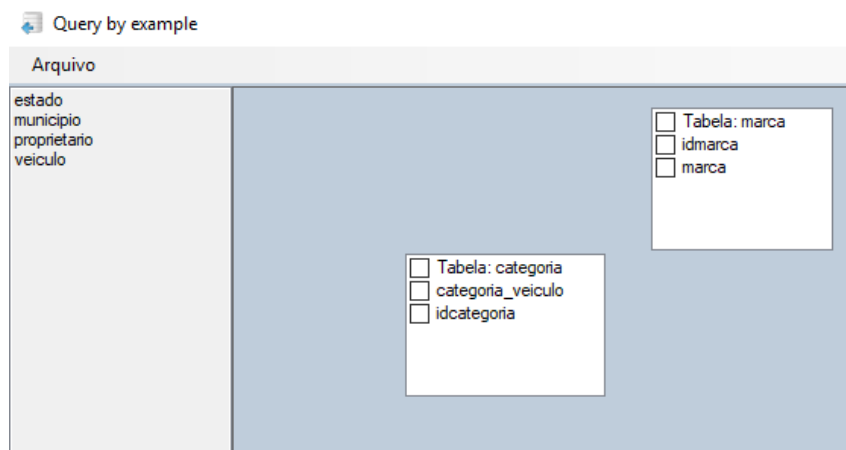
```
SELECT a.attname
FROM pg_class c, pg_attribute a
WHERE a.attrelid = c.oid
      AND c.relname = [nomeTabela]
      AND a.attname NOT IN ( 'cmax', 'cmin', 'ctid', 'tableoid',
                             'xmax', 'xmin' )
ORDER BY attname
```

Fonte: Autoria própria (2018).

Para cada tabela adicionada será criado dinamicamente um componente do tipo Caixa de listagem verificada (*CheckedListBox*) chamado `listaAtributos` que receberá os

resultados da consulta, em que o primeiro item guardará o nome da tabela e os demais seus atributos. Este componente `listaAtributos` será adicionado a uma lista do tipo Caixa de listagem verificada. Estas listas serão adicionadas ao painel principal (espaço onde serão exibidas as tabelas escolhidas pelo usuário). O usuário pode interagir com a tabela adicionada no QBE por meio da interação clicar e arrastar, de modo a organizar visualmente os componentes da consulta. A Figura 19 apresenta duas tabelas dispostas no painel do QBE.

Figura 19 – Painel com as tabelas escolhidas pelo usuário.



Fonte: Autoria própria (2018).

Após estes procedimentos as tabelas e seus atributos estarão dispostos no painel, porém não sabe-se qual atributo é uma chave estrangeira. Um dos pontos levantados para esta pesquisa foi a identificação de chaves estrangeiras para facilitar a montagem das consultas e sugerir condição de junção entre as tabelas. Esta marcação será feita por meio da concatenação de uma *String* no atributo para identificação da chave estrangeira. A consulta apresentada na Figura 20 retorna o nome da coluna que é uma chave estrangeira, a tabela referenciada e a coluna referenciada, em que o atributo “tabelaFK” é o nome da tabela para qual a consulta está sendo realizada.

Figura 20 – Código SQL: chaves estrangeiras de uma tabela.

```

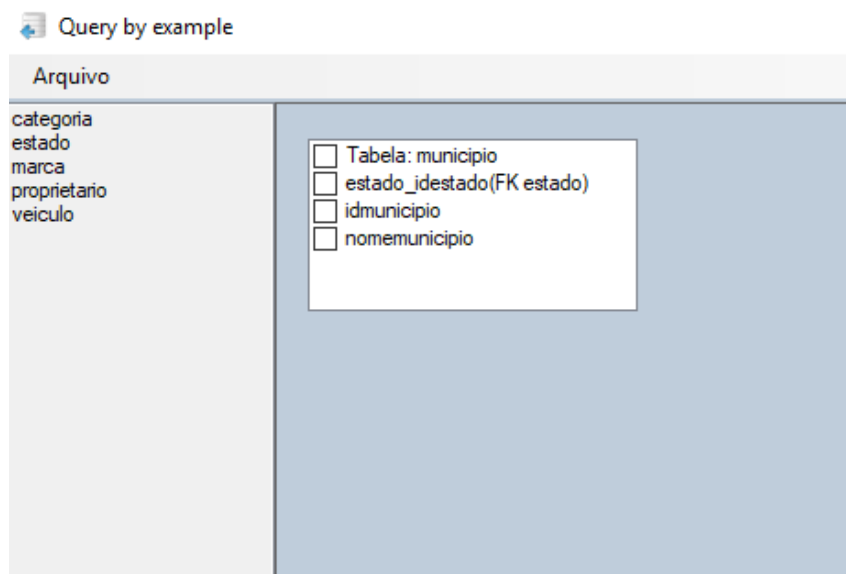
SELECT a.attname AS coluna , clf.relname As tabela_ref ,
       af.attname AS coluna_ref
FROM pg_catalog.pg_attribute a
JOIN pg_catalog.pg_class cl
      ON (a.attrelid = cl.oid AND cl.relkind = 'r'
          AND cl.relname = [tabelaFK]
JOIN pg_catalog.pg_namespace n
      ON (n.oid = cl.relnamespace)
JOIN pg_catalog.pg_constraint ct
      ON (a.attrelid = ct.conrelid AND ct.confrelid!= 0
          AND ct.conkey[1] = a.attnum)
JOIN pg_catalog.pg_class clf
      ON (ct.confrelid = clf.oid AND clf.relkind = 'r')
JOIN pg_catalog.pg_namespace nf
      ON (nf.oid = clf.relnamespace)
JOIN pg_catalog.pg_attribute af
      ON (af.attrelid = ct.confrelid AND
          af.attnum = ct.confkey[1])

```

Fonte: Autoria própria (2018).

Para o atributo retornado será concatenado as iniciais “FK” (*Foreign key*) e o nome da tabela referenciada. A Figura 21, mostra o atributo “estado_idestado” onde o mesmo é uma chave estrangeira da tabela “estado”.

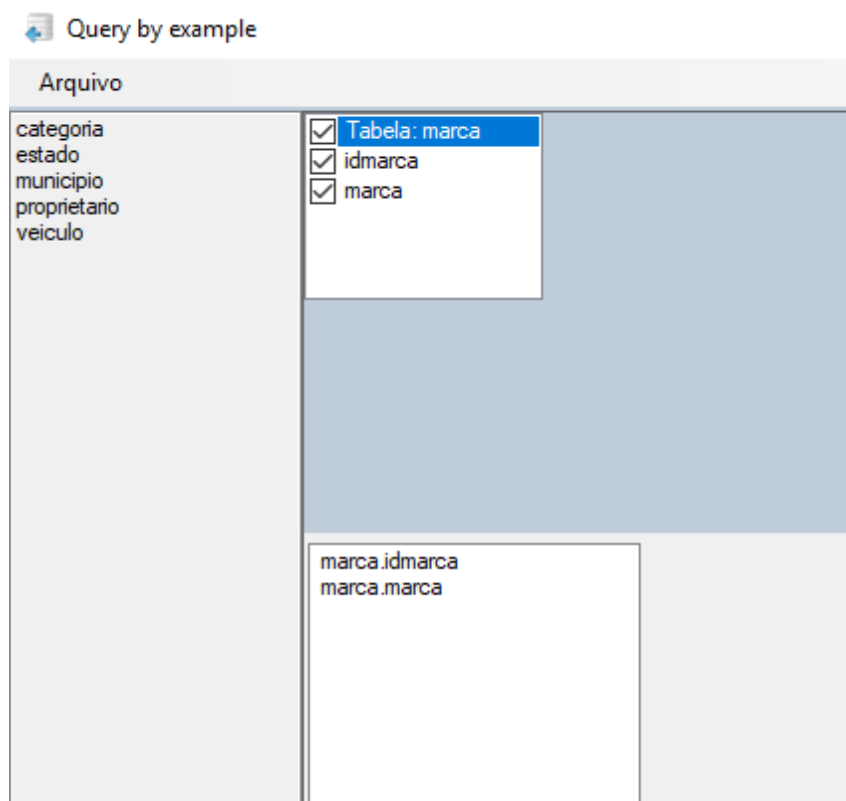
Figura 21 – Tabela com indicação de chave estrangeira em seu atributo.



Fonte: Autoria própria (2018).

Os atributos serão escolhidos por meio de marcações nas Caixas de listagem verificada. Estas marcações serão monitoradas pelo evento de clique em que será chamado um procedimento. Quando houver uma ação de marcação (*check*) e o procedimento for chamado será validado se em alguma das tabelas do painel existem itens marcados. Caso haja itens será necessário verificar se é o primeiro item da tabela (o nome da tabela), pois marcando o nome da tabela, indicará que todos os atributos dela estarão na consulta. Um outro painel é criado para manipulação e formação da consulta, denominado painelConsulta. O painelConsulta contém uma lista para armazenar os atributos escolhidos pelo usuário, esta lista fica posicionada no canto esquerdo do painelConsulta. Portanto, caso o primeiro atributo da tabela (nome da tabela) seja marcado, será necessário adicionar todos os atributos desta tabela, na lista que armazena os atributos escolhidos. Antes de adicioná-los é verificado se eles já estão na lista de atributos. Como foi marcado (*checked*) apenas o primeiro atributo (nome da tabela) o estado dos demais deverá ser alterado de desmarcado (*unchecked*) para marcado (*checked*). A Figura 22 mostra o nome da tabela “marca” selecionada e por consequência seus atributos também. No painelConsulta é apresentada a lista de atributos escolhidos com os itens da tabela “marca”.

Figura 22 – Ação de marcar o primeiro atributo para selecionar e adicionar todos.

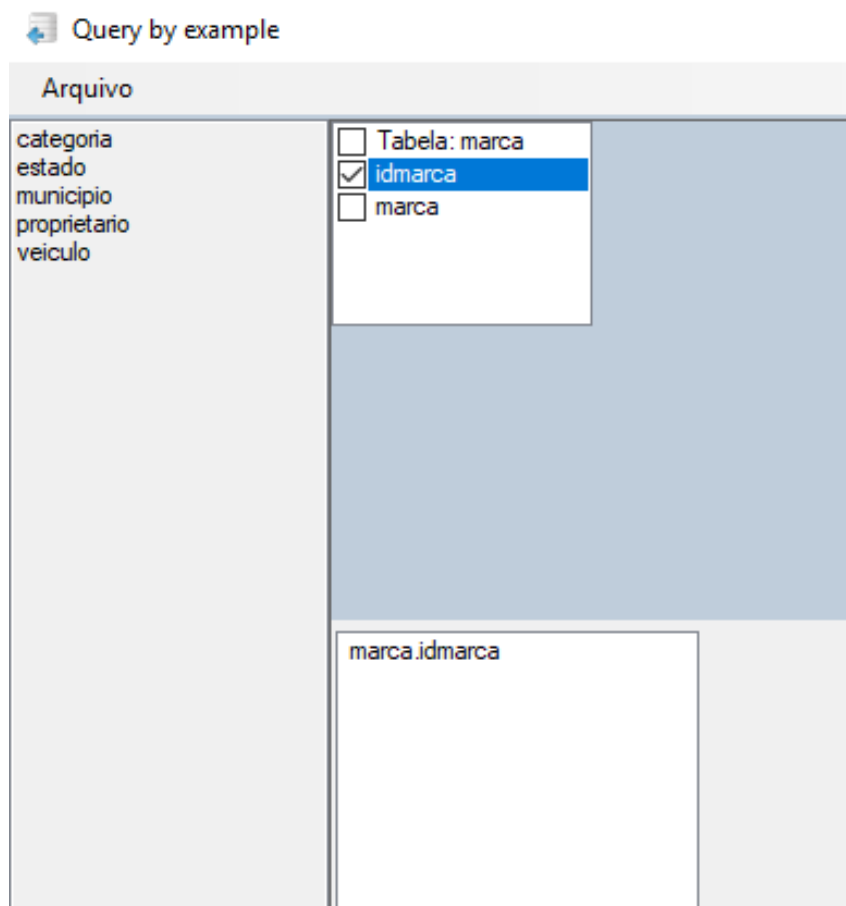


Fonte: Autoria própria (2018).

Caso o atributo marcado não seja o primeiro (o nome da tabela), basta verificar se

o mesmo já está adicionado na lista de atributos escolhidos, caso não, adicioná-lo. Neste caso não será necessário tratar a alteração de estado do *checked*, pois ao clicar isto é feito automaticamente. Na Figura 23 será mostrado o atributo “idmarca” da tabela “marca”, sendo marcado e por consequência adicionado na lista de atributos.

Figura 23 – Ação de marcar um atributo na tabela.



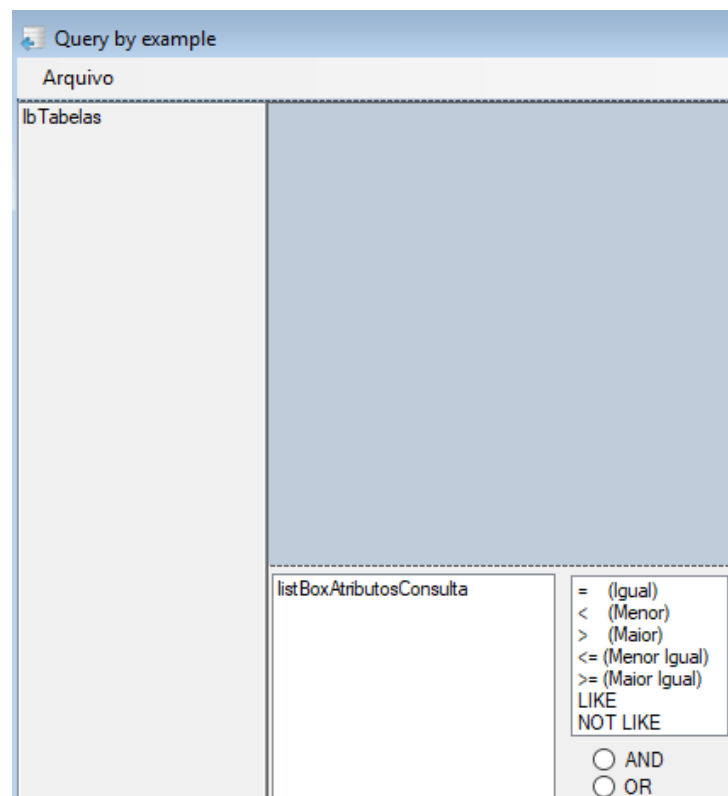
Fonte: Autoria própria (2018).

Um item também poderá ser desmarcado, esta ação será monitorada pelo evento *ItemCheck* que será tratado por um procedimento. Faz-se necessário a verificação do item desmarcado, se é o primeiro item da lista (o nome da lista), neste caso todos os itens da tabela devem ser desmarcados e removidos da lista de atributos. O item que sofreu a ação será comparado com as tabelas escolhidas, ao encontrar a tabela que sofreu a ação, seus atributos serão comparados com os da lista de atributos escolhidos e removidos da mesma. Os atributos da tabela deverão ter seu estado alterado para desmarcado (*unchecked*). Caso o item desmarcado não seja o primeiro da lista (o nome da tabela) a lista de atributos escolhidos será percorrida para encontrá-lo e removê-lo.

3.5 Montagem e Retorno da Consulta

A ferramenta terá novos componentes no painelConsulta que ajudarão na elaboração da consulta. A lista de atributos escolhidos corresponde aos atributos projetados na cláusula SELECT. Uma lista com operadores lógicos de igualdade e diferença, e operadores relacionais (maior, menor, maior igual e menor igual) são disponibilizados para elaboração das condições da cláusula WHERE. Os operadores AND e OR, disponibilizados como botões de rádio (*radioButton*) também serão usados na condição WHERE, porém eles permanecerão desabilitados até que a condição WHERE não esteja vazia. A Figura 24 apresenta estes componentes na interface.

Figura 24 – *ListBox* para armazenar os atributos escolhidos e lista de operadores para as condições.



Fonte: Autoria própria (2018).

Os itens da lista de atributos escolhidos podem ter alguma condição, para isto será necessário clicar em cima do atributo na lista para que ele fique selecionado e repetir o processo na lista de operadores para selecionar o operador desejado. Serão disponibilizados dois campos (“Valor” e “Campo”) para que seja possível adicionar o valor desejado para condição. Caso a condição seja um valor constante, o campo “Valor” deve ser marcado e seu valor atribuído. Porém caso deseje-se que a comparação seja feita entre dois atributos o campo “Campo” deverá ser marcado e também digitado o campo desejado, no mesmo

formato que se encontram os atributos da lista de atributos escolhidos. Em seguida o usuário deverá clicar no botão “– >” que será responsável pela montagem da condição.

Para o procedimento deste botão será verificado qual campo foi marcado e o valor será armazenado em uma variável. Caso o valor da entrada seja do campo "Campo" também será adicionado o nome da tabela que o atributo pertence, a lista chamada tabelasFrom que será usada para o FROM da consulta SQL.

Como pode-se ter atributos que são chaves estrangeiras, eles terão uma String concatenada e será necessário removê-la para que não ocorra erros na hora da execução da consulta. Portanto o atributo selecionado para montar a condição WHERE passará pela função “*Split*” e será recuperado apenas a parte sem a String de marcação da chave estrangeira.

Será criado uma variável denominada “atributoWhere” onde a mesma receberá o retorno da função “*Split*” concatenado com o operador selecionado e o valor do campo. Então esta variável “atributoWhere” será adicionada a uma outra lista para armazenar as condições do WHERE. Quando o tamanho desta lista for maior que zero, os botões de rádio AND e OR serão habilitados para caso seja necessário possa concatená-los na hora de montar uma próxima condição WHERE. A Figura 25 apresenta na lista de atributos escolhidos o atributo “municipio.idestado” que é uma chave estrangeira, selecionada, o operador de igualdade, o campo “valor” com valor desejado para comparação e por último a lista de condições WHERE com a condição montada e sem a String de referência para chave estrangeira.

Figura 25 – Painel com os componentes responsáveis pela elaboração e montagem da consulta.

Fonte: Autoria própria (2018).

No painelConsulta terá um botão denominado “Executar”, que executa o procedimento para montagem da consulta. A lista de atributos escolhidos será percorrida passando cada elemento da lista pela função “*Split*”, para remoção da string de marcação no atributos que são chave estrangeira. Todos os atributos da lista serão concatenados em uma variável chamada “selectAtributo”. Os nomes das tabelas escolhidas serão concatenados em uma outra variável chamada “fromTabela” e por último a lista de condições será percorrida e seus itens concatenados em uma variável chamada “whereCondicao”.

Com todas as variáveis preenchidas basta organizá-las em uma consulta. Será feito uma validação para verificar se a variável “whereCondicao” não está vazia, pois pode haver uma consulta sem WHERE, caso a variável não esteja vazia será concatenada na consulta como as outras:

```
SELECT [selectAtributo]
FROM [fromTabela]
WHERE [whereCondicao]
```

A consulta resultante da QBE será colocada em uma variável do tipo NpgsqlCommand e executada, seu resultado será atribuído a um DataSet (um componente que pode ser usado para armazenar dados de um banco de dados). Para que os dados possam ser exibidos de maneira organizada e em forma de tabela, uma nova janela que contém um DataGridView que irá receber este DataSet. Para mostrar a tabela será criado um novo formulário, que irá conter a consulta SQL construída e seu resultado em forma de tabela.

Para exibição da consulta em SQL será usado uma lista, que irá armazenar e exibir, linha a linha o resultado. A lista irá receber as variáveis que contém as partes da consulta, em que cada variável será adicionada em uma linha. E por último será adicionado os componentes DataGridView e o ListBox no Form, o mesmo será exibido.

Figura 26 – Form com a exibição dos resultados da consulta.

Fonte: Autoria própria (2018).

A Figura 26 contém uma consulta com a tabela “marca”, onde foram marcados os atributos “marca.idmarca” e “marca.marca”, com a condição de que o atributo “marca.idmarca” deverá ter o valor “2”. No formulário que exibe o resultado, terá a princípio a consulta em SQL e seu retorno em forma de tabela.

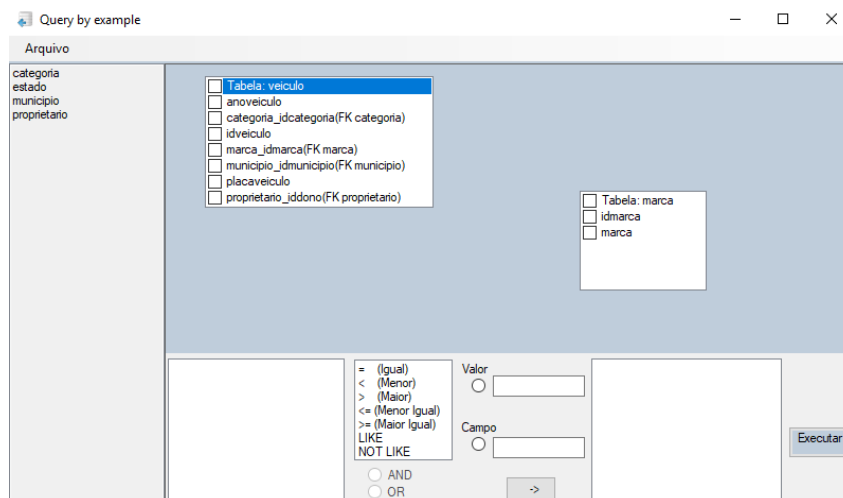
3.6 Estudo de Caso

Esta seção irá apresentar um exemplo de consulta QBE para que seja apresentado todo o funcionamento da interface da ferramenta. Para a construção da consulta, um banco de dados pertencente a uma garagem/loja de veículos usados, composto das seguintes tabelas: “categoria”, “estado”, “marca”, “municipio”, “proprietario” e “veiculo”.

Ao iniciar a ferramenta o usuário deve realizar a conexão com o banco de dados. Então o primeiro passo será clicar no botão “Arquivo” que fica localizado no canto esquerdo superior da janela, ele abrirá a opção “Conexão” e então clicar nesta.

Após a conexão ser aberta será carregado na tela os nomes das tabelas presentes no banco. Para a escolha das tabelas que farão parte da consulta, basta realizar um duplo clique sobre o nome da mesma e automaticamente ela será carregada para o painel central, juntamente aos seus atributos. Caso seja escolhida mais de uma tabela, elas serão carregadas no painel uma sobre a outra, então será necessário clicar e arrastar para que elas se desloquem e possam ser organizadas pelo usuário no painel.

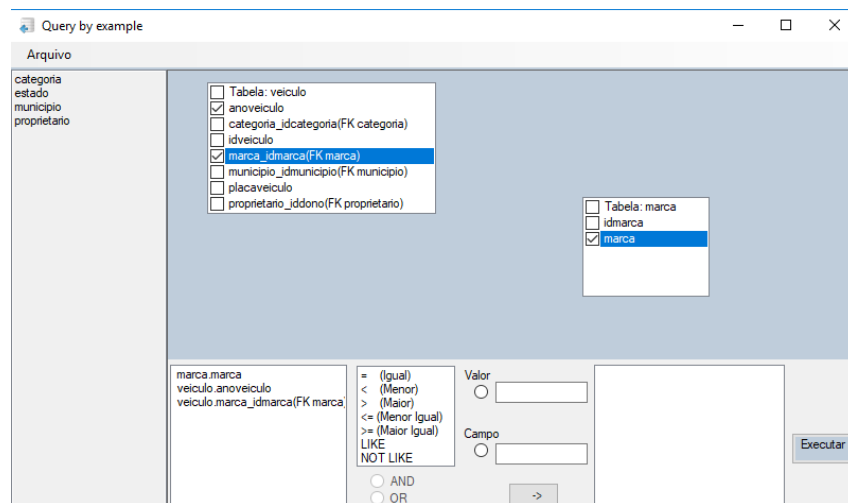
Figura 27 – Painel com as tabelas escolhidas e organizadas.



Fonte: Autoria própria (2018).

A Figura 27 apresenta o painel que contém as duas tabelas escolhidas para realizar a consulta. Pode-se notar que os nomes das tabelas escolhidas foram removidos da lista de tabelas disponíveis, para indicar que uma tabela não pode ser escolhida duas vezes. Com as tabelas no painel será necessário escolher os atributos que serão utilizados na consulta, para isto basta marcar (*checked*) os atributos escolhidos com um clique, que automaticamente eles aparecerão em uma lista localizada no canto esquerdo do painel Consulta.

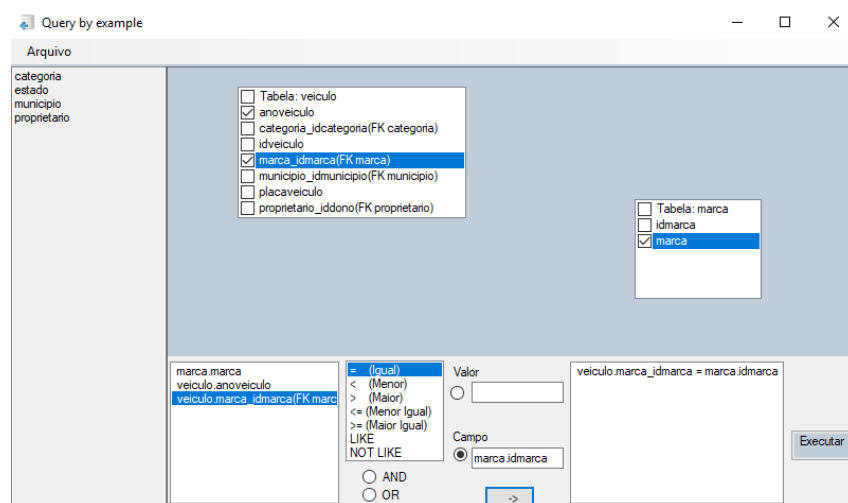
Figura 28 – Pannel com os atributos escolhido e exibidos em uma lista.



Fonte: Autoria própria (2018).

A Figura 28 exibe os atributos marcados “anoveiculo”, “marca_idmarca” da tabela “veiculo” e “marca” da tabela “marca”. O atributo “marca_idmarca” é uma chave estrangeira da tabela “marca”. Estes aparecem na lista localizada no painel Consulta onde acontecerá o restante da montagem da consulta. Como apresentado anteriormente, pode-se adicionar condições para a cláusula WHERE, é necessário selecionar o atributo que irá ter a condição, selecionar o operador e adicionar o valor no campo de input, deve-se escolher se o valor é do tipo “Valor” ou “Campo” e então clicar no botão “— >” que montará a cláusula e a colocará em uma outra lista localizada no canto direito do painel.

Figura 29 – Pannel com os atributos escolhido e a cláusula WHERE montada.

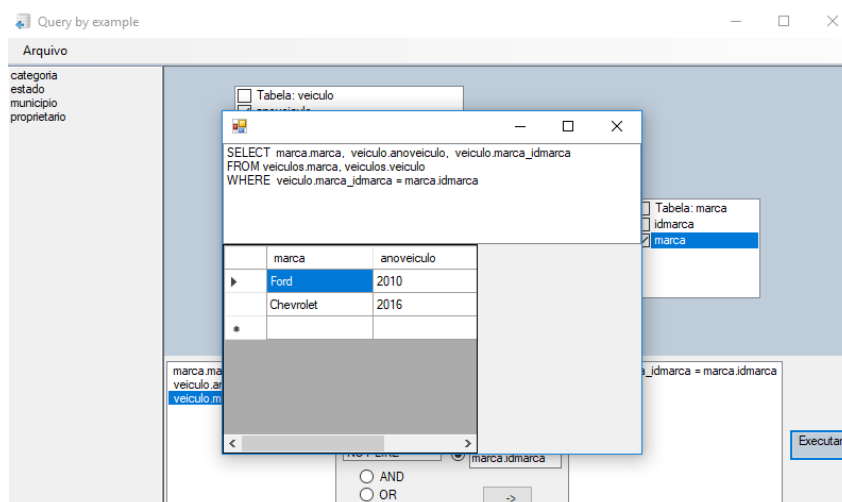


Fonte: Autoria própria (2018).

A Figura 29 contém uma lista no canto direito do painel com a cláusula mon-

tada, em que a chave estrangeira “veiculo.marca_idmarca” deve ser igual ao atributo “marca.idmarca” da tabela “marca”. Para execução da consulta, basta clicar no botão “executar”, que mostrará uma nova janela com os resultados da consulta. A Figura 30 exibe o janela com a consulta SQL gerada e o seu resultado em forma de tabela.

Figura 30 – Formulário com a consulta SQL montada e seus resultados.



Fonte: Autoria própria (2018).

4 Considerações Finais

4.1 Conclusões

A presente monografia teve como objetivo construir um protótipo de uma ferramenta que auxilia as construções de consultas SQL de maneira mais simplificada, pois o conceito de banco de dados não está limitado para a área de Tecnologia da Informação, hoje em dia quase todos os comércios que utilizam algum tipo de sistema necessitam trabalhar com banco de dados. Porém esta monografia não se limitou a parte comercial, pois hoje já existem softwares livres que atendem a esta necessidade. Entre os desafios deste desenvolvimento os principais foram o aprendizado de um novo ambiente de desenvolvimento *Microsoft Visual Studio* e da linguagem de programação *Visual Basic.net*, na utilização do provedor de dados *Npgsql* usado para realizar a conexão entre O *Visual Basic.net* e o *PostgreSQL*, que limitou a implementação da ferramenta até as funcionalidades mais básicas da linguagem SQL. Além disso, precisou-se fazer um estudo detalhado dos metadados armazenados no *PostgreSQL*, pois é um conceito ainda não muito abordado dentro das disciplinas de Banco de Dados. Contudo o conhecimento e crescimento adquirido durante o desenvolvimento deste trabalho foi muito grande.

O objetivo de criar um protótipo de uma ferramenta para consultas SQL foi atingido, contudo diante das dificuldades encontradas algumas funcionalidades ficaram para trabalhos futuros. A ferramenta desenvolvida é capaz de realizar a conexão com o banco de dados PostgreSQL utilizando o provedor de dados Npgsql, recuperar as tabelas criadas no banco, disponibilizar estas tabelas e seus atributos para o usuário, auxiliá-lo na elaboração da consultas simples por meio de componentes que facilitam a elaboração da mesma, enviar a consultas para o banco, recuperar os resultados e por fim exibir a consulta montada e seus resultados em forma de tabela. As consultas que podem ser construídas pela ferramenta suportam mais de uma tabela, porém sem funções de agregação e junções diferentes da equijunção.

4.2 Trabalhos Futuros

Como recomendações para trabalhos futuros deve-se focar em aumentar as funcionalidades da ferramenta para que facilite cada vez mais a utilização do usuário. Dentre essas funcionalidades está o suporte para funções de agregações e junções, indicações de chaves estrangeiras por meio de ligações entre a chave e o elemento referenciado, disponibilização de um recurso que possibilite após a exibição dos resultados retornar ao painel principal para montagem de uma nova consulta, botões para exclusão e limpeza dos cam-

pos durante a elaboração da consulta e uma lista com os atributos do campo “campo” aonde o valor é inserido, para que não seja mais necessário digitá-lo, e o tratamento das cláusulas ORDER BY e HAVING.

Referências

- Borland. *Embarcadero*. 2018. Disponível em: <<https://www.embarcadero.com/products/delphi>>. Acesso em: junho 2018. Citado na página 12.
- Documentação PostgreSQL 8.0.0. *Catálogo do Sistema PostgreSQL*. 2018. Disponível em: <<http://pgdocptbr.sourceforge.net/pg80/catalogs.html>>. Acesso em: 23 março 2018. Citado na página 20.
- EISENBERG, A. et al. Sql:2003 has been published. *SIGMOD Record*, ACM, v. 33, n. 1, p. 119–126, mar. 2004. ISSN 0163-5808. Citado na página 5.
- ELMASRI, R.; NAVATHE, S. *Database systems*. [S.l.]: Pearson Education Boston, MA, 2011. v. 9. Citado 3 vezes nas páginas 5, 8 e 11.
- IDERA's. *IDERA Rapid SQL*. 2018. Disponível em: <<https://www.idera.com/rapid-sql-ide/overview>>. Acesso em: 30 junho 2018. Citado 2 vezes nas páginas 12 e 13.
- JIANG, L.; MANDEL, M.; NANDI, A. Gesturequery: A multitouch database query interface. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 6, n. 12, p. 1342–1345, 2013. Citado 2 vezes nas páginas 13 e 14.
- Microsoft. *Microsoft Access*. 2018. Disponível em: <<https://products.office.com/en/access>>. Acesso em: janeiro 2018. Citado 2 vezes nas páginas 12 e 13.
- Microsoft. *Visual Studio*. 2018. Disponível em: <<https://visualstudio.microsoft.com/pt-br/>>. Acesso em: janeiro 2018. Citado na página 12.
- NpgSQL. *NpgSQL*. 2018. Disponível em: <<http://www.npgsql.org/>>. Acesso em: 20 janeiro 2018. Citado na página 12.
- Oracle. *Oracle*. 2018. Disponível em: <<https://www.oracle.com/database/index.html>>. Acesso em: junho 2018. Citado na página 14.
- Oracle Query Builder. *Building Queries Visually*. 2018. Disponível em: <<https://blogs.oracle.com/oraclemagazine/building-queries-visually>>. Acesso em: 14 abril 2018. Citado 2 vezes nas páginas 14 e 19.
- Oracle SQL. *Oracle*. 2018. Disponível em: <<https://www.oracle.com/database/technologies/appdev/sql-developer.html>>. Acesso em: junho 2018. Citado na página 14.
- PgAdmin. *Graphical Query builder*. 2018. Disponível em: <<https://www.pgadmin.org/docs/pgadmin3/1.22/gqb.html>>. Acesso em: julho 2018. Citado 2 vezes nas páginas 17 e 18.
- PostgreSQL. *PostgreSQL*. 2018. Disponível em: <<https://www.postgresql.org/>>. Acesso em: 20 janeiro 2018. Citado na página 12.
- ZLOOF, M. M. Query by example. In: *Proceedings of the National Computer Conference and Exposition (AFIPS)*. Anaheim, California: ACM, 1975. p. 431–438. Citado 5 vezes nas páginas 6, 8, 9, 10 e 11.