MultiCategoria: o processamento de consulta multimodelo atende à categoria Teoria e Programação Funcional

Valter Uotila
Jiaheng Lu
Universidade de Helsinque first.last@helsinki.fi

Dieter Gawlick
Zhen Hua Liu
Souripriya Das
Oracle Corporation
first.last@oracle.com

Gregory Pogossiants
SATS Technologies
gregp_21@yahoo.com

ABSTRATO

A variedade de dados é uma das questões importantes na era do Big Data. Os dados são naturalmente organizados em diferentes formatos e modelos , incluindo dados estruturados, dados semiestruturados e dados não estruturados. Pesquisas anteriores previram uma abordagem para abstrair dados multimodelos com uma categoria de esquema e uma categoria de instância usando a teoria da categoria. Neste artigo, demonstramos um sistema, chamado MultiCategory, que processa consultas multi-modelo baseadas na teoria de categorias e programação funcional. Esta demonstração é centrada em quatro cenários principais para mostrar um sistema tangível.

Primeiro, mostramos como criar uma categoria de esquema e uma categoria de instância carregando diferentes modelos de dados, incluindo dados relacionais, XML, valor-chave e gráficos. Em segundo lugar, mostramos alguns exemplos de processamento de consultas usando a linguagem de programação funcional Haskell. Em terceiro lugar, demonstramos as saídas flexíveis com diferentes modelos de dados para a mesma consulta de entrada. Quarto, para entender melhor a estrutura teórica da categoria por trás das consultas, oferecemos uma variedade de ganchos gráficos para explorar e visualizar consultas como gráficos com relação à categoria do esquema, bem como o procedimento de processamento de consultas com Haskell.

1. INTRODUÇÃO

A variedade de dados é uma das questões mais importantes nos sistemas de gerenciamento de dados modernos para lidar com o desafio do Big Data. Em muitos aplicativos, as fontes de dados são naturalmente organizadas em diferentes formatos e modelos, incluindo dados estruturados, dados semiestruturados e dados não estruturados. Para enfrentar o desafio da variedade, os bancos de dados multimodelos começaram a surgir com uma única plataforma de banco de dados para gerenciar dados multimodelos juntos, com um back-end totalmente integrado para lidar com as demandas de desempenho e escalabilidade [3] .

Vamos considerar um exemplo de um ambiente de dados multimodelo. A Figura 1 ilustra uma aplicação de E-commerce, que contém clientes, uma rede social e informações de pedidos com quatro modelos de dados distintos. Os dados do gráfico de propriedade trazem informações sobre relacionamentos mútuos entre os clientes, ou seja, quem sabe quem, e algumas propriedades do cliente, como nome e limite de crédito.

A localização geográfica dos clientes é armazenada em uma tabela relacional. Nos documentos XML, cada pedido possui um ID e uma sequência de produtos solicitados, cada um dos quais inclui número, nome e preço do produto.

O quarto tipo de dados, pares chave/valor, contém as relações entre diferentes conjuntos de dados. Em um aplicativo típico como a visão 360 do cliente, os usuários de bancos de dados exigem analisar as informações dessas quatro fontes de dados diferentes juntas para permitir uma análise holística dos comportamentos do cliente.

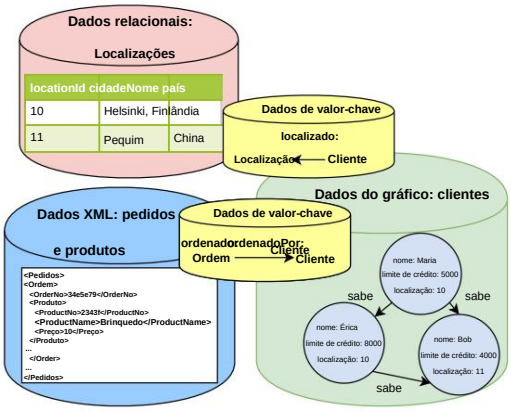


Figura 1: um ambiente de dados multimodelo

A teoria das categorias foi desenvolvida por matemáticos na década de 1940 e tem sido aplicada com sucesso em muitas áreas da ciência, incluindo a ciência da computação. Iniciativas de pesquisa recentes aplicaram a teoria das categorias para a área de banco de dados. Em particular, Spivak [6, 7] usou uma categoria de esquema e um functor de instância para modelar bancos de dados relacionais. Liu et ai. [2] promoveu a teoria das categorias para desempenhar o papel da nova base matemática para raciocinar sobre construções declarativas e transformações entre vários modelos de dad

Embora os trabalhos anteriores tenham vislumbrado o significado teórico de modelar e gerenciar dados com a teoria da categoria, esta demonstração mostra nossa iniciativa de mostrar uma implementação de prova de conceito do MultiCategory, um sistema para suportar o processamento de consultas multimodelo com base na categoria teoria. As partes principais do sistema foram codificadas com a linguagem de programação funcional Haskell, que é amplamente reconhecida por ter uma forte conexão com a teoria das categorias. A estrutura de armazenamento de dados do MultiCategory é estabelecida nos conceitos de categorias de esquema e instância [6], e a estrutura de processamento de consultas é baseada em catamorfismo e estruturas de dados dobráveis [1]. Com essas propriedades-chave, podemos criar um sistema que tenha uma integração consistente com modelos de dados relacionais, hierárquicos e gráficos e mostramos como a teoria da categoria pode ser usada para obter perspectivas valiosas para representação e processamento de consultas de vários modelos.

Em resumo, a demonstração do MultiCategory oferece o seguinte ao público: • métodos

orientados à programação funcional e teórica de categoria para consulta e acesso a dados multimodelo com um esquema unificado;

- uma linguagem de consulta unificada dotada de expressões lambda de Haskell, permitindo que os usuários enviem uma consulta para acessar diferentes modelos de dados sem problemas;
- a flexibilidade de produzir o mesmo resultado com diferentes modelos, o que oferece aos usuários a oportunidade de explorar os mesmos dados com diferentes representações;
- entender melhor a estrutura teórica por trás do consultas, esta demonstração também fornece um visualizador interativo para entender o esquema e as categorias de instância, bem como o procedimento de processamento da consulta.

Em nossa demonstração, os participantes podem compor suas consultas que seguem a sintaxe de nossa linguagem de consulta para pesquisar conjuntos de dados multimodelos . O código-fonte deste sistema está disponível no GitHub [8] e o vídeo de demonstração pode ser assistido online no YouTube [9].

2 PRELIMINARES

Nesta seção, revisamos primeiro a definição matemática de uma categoria [4], seguida pelas descrições do esquema e das categorias de instância que são influenciadas por [6, 7].

Definição 2.1. Uma categoria C consiste em uma coleção de objetos denotados por (C) e uma coleção de morfismos denotados por (C). Para cada morfismo ÿ (C) existe um objeto (C) que é domínio de e um objeto ÿ (C) que é alvo ÿ de . Neste caso denotamos : ÿ . Exigimos que todas as composições definidas de morfismos sejam incluídas em C: se : ÿ ÿ (C) e : ÿ ÿ (C), então (C). Assumimos que a operação de composição é ÿ : ÿ ÿ associativa e que para todo objeto ÿ (C) existe um morfismo de identidade id : ÿ de modo que ÿ id = e id ÿ = sempre que a composição for definida.

Informalmente, podemos entender uma categoria como um grafo dotado da regra de composição.

A Figura 2 constrói uma categoria de esquema unificada, que representa as informações de esquema de um ambiente de dados multimodelo na Figura 1. Conceitualmente, um objeto em uma categoria de esquema inclui dois tipos de tipos de dados: (1) a primeira coleção de tipos de dados consiste em uma string, inteiro, racional, booleano, etc., chamados de tipos de dados predefinidos; e (2) a segunda coleção de tipos de dados inclui entidades, como clientes e produtos. Os morfismos são definidos como as funções digitadas entre os tipos de dados, como um cliente localizado em um determinado local e um pedido feito por um cliente. Além disso, é importante observar que uma categoria de esquema apresenta uma única visão unificada para diferentes modelos de dados. Com base nessa visão, desenvolvemos um mecanismo de consulta unificado para processar diferentes modelos de dados de forma integrada.

Uma categoria de instância modela como as instâncias de dados concretas são armazenadas. Cada objeto da categoria de esquema é mapeado para a estrutura de dados Haskell digitada correspondente na categoria de instância (consulte a Figura 2). Cada morfismo na categoria de esquema é mapeado para uma função Haskell concreta na categoria de instância. O mapeamento entre essas categorias é chamado de functor de instância que é definido em objetos por functores construtores de coleção [1]. Conforme mostrado em nossa demonstração, as consultas são formuladas com base na categoria do esquema e as respostas são recuperadas da categoria da instância com base no functor da instância e nos functores do construtor de coleção.

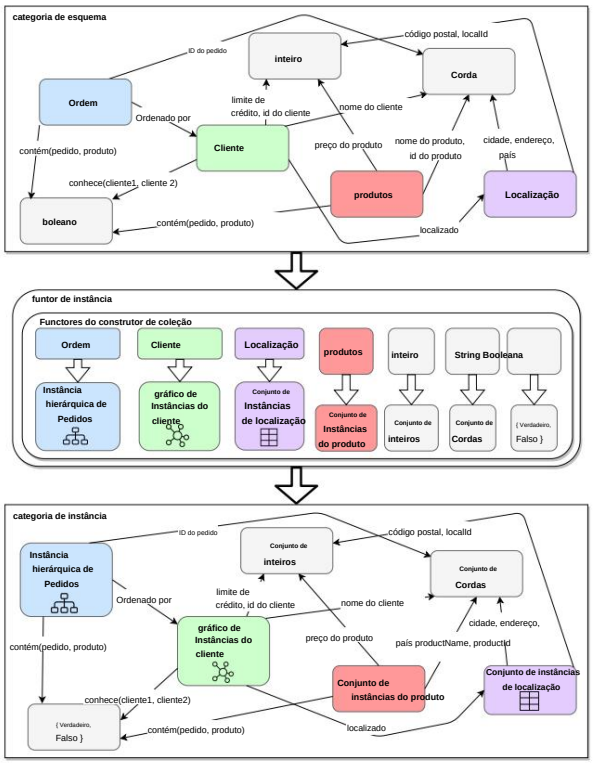


Figura 2: Exemplo de construção teórica de uma categoria

Nas categorias de esquema e instância, podemos seguir qualquer caminho para formar uma função bem definida entre o nó inicial e o nó final do caminho, graças à regra de composição da Definição 2.1.
Por exemplo, existe um morfismo (borda) na instância categoria que nos dá que o cliente faz o pedido e outro morfismo que nos dá que o pedido inclui o produto.
Com base na regra de composição desses morfismos existe um morfismo bem definido que nos dá que o cliente compra o produto . Essa propriedade de composicionalidade é importante para garantir a correção de programas para percorrer vários modelos de dados.

3 VISÃO GERAL DO SISTEMA

Nesta seção, fornecemos uma visão geral da arquitetura , linguagem de consulta e mecanismo de processamento de consulta do MultiCategory . Para obter mais detalhes sobre soluções técnicas, um tutorial e um guia de instalação, você pode encontrar na documentação do MultiCategory e no Github [8].

A Figura 3 representa a arquitetura do MultiCategory que consiste no front-end e no back-end. Em particular, o frontend cria uma interface web e visualizações de dados para dados relacionais, documentos hierárquicos e gráficos. O back-end é responsável pelo processamento de consultas e pela implementação de construções teóricas de categorias.

3.1 Linguagem de consulta multimodelo

Desenvolvemos uma linguagem de consulta multimodelo que encapsula funções e expressões Haskell. Por exemplo, a seguinte consulta

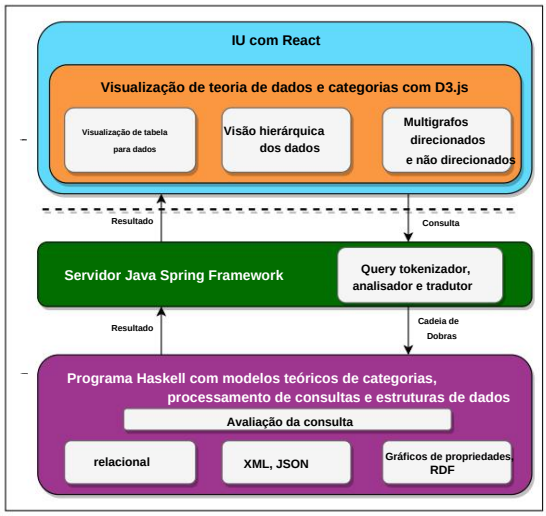


Figura 3: A arquitetura do MultiCategory

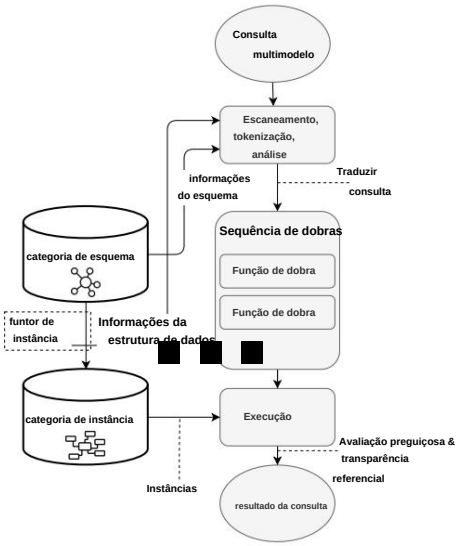


Figura 4: o diagrama de fluxo de trabalho do processamento de consultas

localiza os clientes cujo limite de crédito é maior que um limite, digamos 3.000.

Exemplo 3.1. QUERY (\x -> se creditLimit x > 3000
então **contras** x senão **nulo)**
DE clientes
PARA gráfico/xml/relacional

Cada bloco de consulta começa com a palavra-chave **QUERY ,** que é seguida por uma função definida com a notação lambda de Haskell. No Exemplo 3.1, x é uma variável que representa um cliente no gráfico, e a consulta retorna o gráfico dos clientes com limite de crédito > 3000. A palavra-chave **FROM** especifica a coleção de origem dos dados. A palavra-chave **TO** configura o modelo de dados do resultado.
Observe que no exemplo acima o modelo retornado pode ser gráfico, relacional ou XML. A Figura 5 demonstra os três diferentes modelos de representação para as respostas da consulta do Exemplo 3.1.

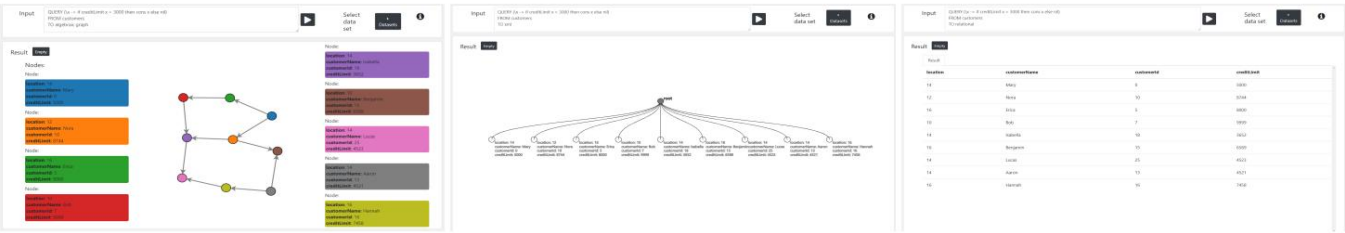
Exemplo 3.2. LET t BE
QUERY (\x xs -> if elem "Book" (map productName (orderProducts
x)) then **cons** x xs else xs)
DE pedidos PARA relacional IN
QUERY (\x -> se houver (\y -> ordersBy y clientes == x) t then **cons**
(customerName x, countryName(localizado em x locais))
else **nil)**
DE clientes PARA gráfico algébrico/relacional/xml

A consulta no Exemplo 3.2 retorna um gráfico que contém nomes e localizações dos clientes que pediram um livro, que envolve relação, XML e tipos de dados chave/valor. A Figura 6 mostra o resultado dessa consulta. Existem duas cláusulas QUERY nesta consulta que correspondem às duas funções dobradas. As cláusulas estão conectadas com a estrutura **LET BE IN** que funciona da mesma forma que o mecanismo correspondente em Haskell. A palavra-chave **LET** introduz uma variável (isto é,) que conecta as funções fold. Em particular, a primeira cláusula **QUERY** localiza qualquer pedido que contenha um livro. A segunda cláusula **QUERY** encontra clientes que fizeram tais pedidos. Os resultados contêm o nome do cliente e as informações de localização (ou seja, countryName).

3.2 Mecanismo de processamento de consultas A Figura 4 descreve o fluxo de trabalho principal do processamento de consultas em Multi Categoria. Quando um usuário insere uma consulta no sistema, ela é analisada em uma sequência de funções dobradas em relação às informações do esquema da categoria do esquema. A sequência de dobras é enviada para o programa Haskell em execução no back-end. O programa Haskell acessa a categoria de instância e executa a sequência de funções de dobra que é apenas código Haskell puro. Observe que não exigimos que todas as estruturas de dados sejam instâncias da classe de tipo dobrável de Haskell, pois podemos usar generalizações de dobras para consultar tipos de dados algébricos mais complexos. Por exemplo, usamos a função foldg para consultar os gráficos algébricos do pacote Algebra.Graph [5]. Por fim, o resultado é retornado ao frontend, onde é visualizado dependendo do modelo que o usuário definiu na consulta.

4 CENÁRIOS DE DEMONSTRAÇÃO

4.1 Usando categorias de esquema e instância Primeiro apresentamos nosso sistema convidando os participantes a visualizar o esquema e as categorias de instância de vários conjuntos de dados por meio da interface gráfica. Esta demonstração usará seis conjuntos de dados sintéticos e reais diferentes. Cada conjunto de dados inclui modelos diferentes, como dados relacionais , XML, JSON, RDF e gráficos de propriedades. Os participantes podem selecionar um conjunto de dados, visualizar o esquema relacionado e as categorias de instância e examinar os nós e arestas de qualquer gráfico para obter mais informações. Por exemplo, pode-se descobrir que o tipo de dados do cliente tem um atributo customerName que é conside Clientes para OString.



(um gráfico (b) Árvore (c) Relação

Figura 5: Três representações diferentes do mesmo resultado da consulta do Exemplo 3.1

Figura 6: O resultado em gráficos e modelos relacionais para a consulta no Exemplo 3.2

4.2 Consultando dados de vários modelos Em MultiCategory , criamos uma coleção de consultas de exemplo que podem consultar diferentes modelos de dados juntos. Por exemplo, um conjunto de dados de comércio eletrônico inclui todas as sete combinações possíveis de consultas de vários modelos combinando modelos relacionais, XML e gráficos. Esta demonstração permite que os participantes formulem suas consultas com orientação e observem os resultados em diferentes modelos de saída. Atualmente, o sistema não suporta processamento de dados em grande escala devido a razões de implementação, mas geralmente a estrutura baseada em teoria da categoria funciona bem.

4.3 Visualização de consultas multimodelo Para obter um melhor entendimento da estrutura teórica por trás das consultas, o MultiCategory fornece um visualizador automático para as consultas baseadas em função de dobra. Esse recurso visualiza as consultas como gráficos em relação à categoria da instância. Em particular, após a execução de uma consulta, a consulta é visualizada como um gráfico que exibe uma composição de funções de dobra Haskell. Cada função de dobra possui pelo menos uma expressão lambda na qual o usuário pode clicar para ver as informações detalhadas na interface gráfica.

5 COMPARAÇÃO COM OS SISTEMAS EXISTENTES

Os bancos de dados multimodelos existentes, por exemplo, ArangoDB e OrientDB, são implementados com base em um único modelo dominante, de modo que não podem ser chamados de “nativos” em relação a todos os modelos que suportam . Em contraste, desenvolvemos o MultiCategory de forma que cada modelo de dados seja igual e cada instância seja armazenada em sua estrutura de dados nativa. Nenhuma operação de transformação específica é necessária quando os dados são carregados. Uma visão unificada é gerada para acomodar diferentes modelos de dados juntos. Além disso, existem poucos sistemas que suportariam tantos modelos (relacionais , XML, JSON, RDF e gráficos de propriedade) quanto o MultiCategory suporta.

6 CONCLUSÃO E TRABALHOS FUTUROS

MultiCategory é um sistema tangível que aplica teoria de categoria para modelar e consultar dados multimodelos. Implementamos uma linguagem de consulta com uma linguagem de programação funcional. Visualizamos as construções teóricas da categoria, ou seja, esquemas e categorias de instâncias, e processamento de consultas para mostrar as conexões entre a teoria e as aplicações. Observe que o MultiCategory ainda não é um sistema de banco de dados completo e, devido à sua implementação, também não é um sistema de big data.

A categoria do esquema e a categoria da instância são fixas e predefinidas manualmente. No futuro, consideramos gerar automaticamente essa exibição de categoria unificada com base em conjuntos de dados de entrada. Desenvolvemos a estrutura teórica para que ela possa definir junções de vários modelos e transformações de dados, esquemas e consultas.

REFERÊNCIAS

[1] Torsten Grust. 1999. Compreender as consultas. doutorado Dissertação. Universitaet Constança, Constança.

[2] Zhen Hua Liu, Jiaheng Lu, Dieter Gawlick, Heli Helskyaho, Gregory Pogossiants e Zhe Wu. 2018. Sistemas de gerenciamento de banco de dados multimodelo - um olhar para frente. Nos Workshops Polystores VLDB 2018. 16–29.

[3] Jiaheng Lu e Irena Holubová. 2019. Bancos de dados multimodelos: uma nova jornada para lidar com a variedade de dados. Computação ACM. Sobreviver 52, 3 (2019), 55:1–55:38.

[4] Saunders MacLane. 1971. Categorias para o matemático de trabalho. Springer, Nova York, NY. https://doi.org/10.1007/978-1-4612-9839-7 [5] Andrey Mokhov.

2017. Gráficos Algébricos com Classe (Pérola Funcional). In Proceedings of the 10th ACM SIGPLAN International Symposium on Haskell (Oxford, Reino Unido) (Haskell 2017). Nova York, NY, EUA, 2–13. https://doi.org/10.1145/3122955.3122956

[6] David Spivak. 2014. Teoria das categorias para as ciências. (2014).

[7] David I. Spivak. 2010. Migração de Dados Funcionais. CoRR abs/1009.1166 (2010). arXiv:1009.1166 http://arxiv.org/abs/1009.1166

[8] Valter Uotila. 2021. Documentação multicategoria e códigos de sistema. https: // multicategory.github.io/, https://git.io/JvPqM. Acessado em 19 de julho de 2021.

[9] Valter Uotila e Jiaheng Lu. 2021. Vídeo de demonstração MultiCategory. https://youtu.be/ uceIi91AGsg. Acessado em 19 de julho de 2021.