Incluindo Consultas por Similaridade em SQL*

**Maria Camila N. Barioni, Humberto L. Razente,
Agma J. M. Traina, Caetano Traina Jr.**[1]

[1]ICMC – Instituto de Ciências Matemáticas e de Computação
Avenida do Trabalhador Sãocarlense, 400 - 13566-590 – São Carlos, SP
USP – Universidade de São Paulo – Brasil

{mcamila, hlr, agma, caetano}@icmc.usp.br

Abstract. *In order to adequately support the types of queries required by modern database applications, the DBMS needs to allow posing similarity queries. However, the standard SQL query language does not provide effective support for such queries. This paper describes a similarity retrieval engine that allows posing similarity queries using an extension of the SQL language in a relational DBMS. The engine allows evaluating every aspect of the proposed extension, including the DDL and DML statements, and employ metric access methods to accelerate the queries.*

Resumo. *Para atender às buscas necessárias às aplicações de banco de dados modernas é preciso fornecer o suporte para buscas por similaridade em SGBD. Entretanto, a linguagem SQL não permite representar essas consultas. Este artigo descreve um protótipo de uma ferramenta que implementa um mecanismo para realizar consultas por similaridade usando uma extensão da linguagem SQL em um SGBD relacional. O protótipo permite demonstrar todos os aspectos da extensão, incluindo os comandos da DDL e DML, e utiliza métodos de acessos métricos para acelerar as consultas.*

1. Introdução

Os Sistemas de Gerenciamento de Bases de Dados (SGBD) foram desenvolvidos para armazenar e recuperar dados formados apenas por números ou cadeias de caracteres. Entretanto, nas últimas décadas houve um aumento expressivo não só da quantidade, mas também da complexidade dos dados manipulados em banco de dados, dentre eles dados de natureza multimídia (como imagens, áudio e vídeo), informações geo-referenciadas, séries temporais, entre outros. Assim surgiu a necessidade do desenvolvimento de novas técnicas que permitam a manipulação eficiente de tipos de dados complexos.

As consultas suportadas pelos SGBD são baseadas na propriedade de relação de ordem total, que permite que números e cadeias de caracteres sejam comparados com operadores como $<$, $>$ e $=$. Entretanto, consultas sobre dados complexos são raramente baseadas em relação de ordem, mas sim em uma noção de similaridade, específica para cada domínio. Assim, como a estratégia de busca dos SGBD não satisfaz as necessidades das aplicações que manipulam esses tipos de dados, o conceito de busca por similaridade – consultas que realizam busca por objetos da base similares a um objeto de consulta, de acordo com uma medida de similaridade – está se tornando cada vez mais relevante.

Embora haja a necessidade de fornecer suporte para a realização desse tipo de consultas nos SGBD, o atual padrão da linguagem SQL não prevê a realização de consultas

*Este trabalho tem o apoio da FAPESP (processos 01/11987-3, 01/12536-5 e 02/07318-1), do CNPq (processos 52.1685/98-6, 52.1267/96-0 e 860.068/00-7), e da CAPES.

por similaridade. Este artigo apresenta um mecanismo, denominado SIREN (*SImilarity Retrieval ENgine*), que permite a realização de consultas por similaridade em SQL.

O SIREN foi implementado para validar a adequação de uma proposta de extensão da linguagem SQL padrão que permite a realização de consultas por similaridade [Barioni et al., 2005] (uma versão de demonstração da ferramenta e a sintaxe completa da linguagem podem ser acessadas em http://gbdi.icmc.usp.br/siren/). Essa versão implementa um serviço entre um SGBD convencional e os programas de aplicação, interceptando todo comando SQL enviado por uma aplicação. Caso o comando não possua nenhuma cláusula que envolva construções de similaridade ou nenhuma referência a um objeto complexo, ele é enviado para um SGBD e retorna a resposta de volta para a aplicação. Assim, quando uma aplicação submete apenas comandos em SQL padrão, estes são executados de maneira transparente. Por outro lado, caso o comando possua alguma cláusula que envolva construções de similaridade ou alguma referência a um objeto complexo definido pela extensão da linguagem, o comando é re-escrito e as operações de similaridade são executadas internamente, utilizando o SGBD em questão para executar as operações sobre os dados tradicionais.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2 é apresentado o mecanismo que realiza as consultas por similaridade em SQL – SIREN. A Seção 3 apresenta alguns exemplos de consultas realizadas por meio da utilização de uma aplicação Web conectada ao SIREN. E a Seção 4 finaliza o artigo apresentando as conclusões.

2. O mecanismo SIREN

O protótipo SIREN é constituído por três componentes principais (veja Figura 1(a)):

- Um componente responsável pela interpretação da especificação de uma extensão da sintaxe SQL para a definição e manipulação de objetos complexos considerando aspectos relacionados à realização de consultas por similaridade;
- Um componente responsável pela extração de características que são utilizadas para a representação e indexação de objetos complexos, como imagens;
- Um componente responsável pela utilização de estruturas de indexação apropriadas para responder às consultas por similaridade.

Nas próximas seções são abordados os principais aspectos relacionados a cada um desses componentes.

2.1. A extensão da linguagem SQL

Para executar consultas por similaridade sobre objetos em um domínio complexo é preciso definir como a similaridade será mensurada e um novo tipo de dados complexo. De uma maneira geral foram considerados dois tipos de objetos complexos – aqueles armazenados como um conjunto de atributos tradicionais (séries temporais, posições geográficas, etc.), e aqueles armazenados como um único objeto binário BLOB (imagens, trilhas de áudio, etc.) – que são contemplados respectivamente pelos domínios `PARTICULATE` e `MONOLITHIC`. Assim, foram definidos dois novos tipos de dados, `PARTICULATE` e `STILLIMAGE`, sendo que este último é uma especialização do domínio `MONOLITHIC`, específico para o armazenamento de imagens.

Também não é possível atualmente definir medidas de similaridade (métricas) em SQL. Assim, foi necessário criar novos comandos, seguindo o estilo dos comandos da DDL (*Data Definition Language*). Foram definidos três comandos para lidar com as medidas de similaridade: `CREATE METRIC`, `ALTER METRIC` e `DROP METRIC`.

Uma vez criadas, as métricas podem ser associadas a um ou vários objetos complexos definidos como atributos em qualquer relação. A definição de como comparar um par

de objetos de tipo complexo é expressa como uma restrição para o atributo, seguindo as duas maneiras usuais de definição de restrições em um comando `CREATE TABLE`: como uma restrição de coluna ou como uma restrição de tabela. Além disso, como essas medidas podem ser utilizadas na criação de índices para acelerar a realização de consultas, elas também podem ser definidas por meio de um comando `CREATE INDEX`.

Os comandos `SELECT`, `UPDATE` e `DELETE` da DML (*Data Manipulation Language*) também foram alterados para permitir a representação de novas construções para expressar predicados por similaridade. A sintaxe do comando `INSERT` da DML não precisou ser alterada, embora a sua implementação precise contemplar os novos tipos de dados. Alguns exemplos da utilização dessa sintaxe serão apresentados na Seção 3.

2.2. Aspectos relacionados à implementação

Os dados do tipo `PARTICULATE` são armazenados pelos atributos que os compõem, e assim não apresentam nenhum requisito especial de armazenamento. Por outro lado, o armazenamento de dados do tipo `STILLIMAGE` precisa considerar também as características associadas a eles. Embora esses tipos de dados sejam armazenados em atributos do tipo `BLOB`, é preciso armazenar as características extraídas deles. A extração de características é uma tarefa geralmente custosa, e deve ser executada uma vez quando um objeto é armazenado na base de dados. As características são armazenadas como atributos textuais ou numéricos associados ao objeto complexo. Como o usuário não especifica atributos nas suas relações para armazenar as características extraídas, o sistema deve criar o local para armazenar essas características e suas associações com os seus respectivos objetos complexos de modo transparente ao usuário.

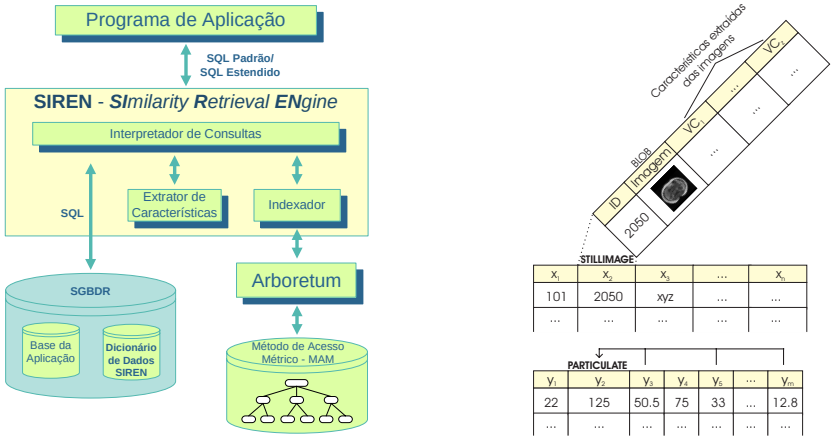


Figura 1: Protótipo SIREN. (a) Arquitetura. (b) Esquema de armazenamento dos novos tipos de dados complexos.

Para objetos do tipo `STILLIMAGE` isso é feito modificando a estrutura das tabelas definidas pelo usuário, que contenham atributos `STILLIMAGE`, da seguinte maneira. Cada atributo `STILLIMAGE` é substituído por uma referência a uma tabela controlada pelo sistema, que possui como atributos um identificador, o atributo `BLOB` que armazena a imagem e o conjunto de atributos que armazenam as características extraídas pelos extratores utilizados em cada métrica definida para o atributo `STILLIMAGE`. Assim, uma nova tabela é criada para cada atributo `STILLIMAGE`. Toda vez que uma nova imagem é armazenada na relação, o SIREN intercepta o comando `INSERT`, armazena os atributos tradicionais na tabela definida pelo usuário e as imagens nas respectivas tabelas do sistema. Em seguida, o SIREN realiza chamadas para os extratores de características e armazena a saída desses extratores nas tabelas do sistema correspondentes.

Toda vez que um usuário solicita dados de suas tabelas, o SIREN realiza uma junção entre as tabelas do sistema e as tabelas do usuário e remove os atributos que armazenam as características, de maneira que o usuário não vê a tabela dividida nem suas características. A Figura 1(b) apresenta uma ilustração do esquema de armazenamento desses novos tipos de dados complexos.

Quando um usuário submete consultas que envolvem predicados de similaridade, o SIREN utiliza as características extraídas dos objetos do tipo `STILLIMAGE` ou o conjunto de atributos que compõem o objeto `PARTICULATE` para executar as operações de similaridade. A versão atual do protótipo conta com três tipos de extratores de características: um extrator de textura (`TEXTUREEXT`), um extrator de forma baseado em momentos de Zernike (`ZERNIKEEXT`) e um extrator de cor baseado no histograma normalizado de cores (`HISTOGRAMEXT`) [Long et al., 2002]. Conta também com cinco operadores de similaridade.

Dois operadores realizam a seleção por similaridade, que são os dois tipos tradicionais de busca por similaridade: *Range query* (Rq) e *k-Nearest Neigbor query* (k-NNq) [Chávez et al., 2001]. Os outros três operadores contemplam a junção por similaridade: *Range Join*, *k-Nearest Neigbors Join* e *k-Closest Neigbors Join* [Böhm and Krebs, 2002].

O método de acesso métrico (MAM) utilizado pelo SIREN, para indexar ambos atributos `PARTICULATE` e `STILLIMAGE`, é a Slim-tree [Traina-Jr. et al., 2002] disponibilizada pela biblioteca de MAM de código livre Arboretum [GBDI-ICMC-USP, 2004]. Nessa implementação já existem procedimentos para a execução de Rq e k-NNq. Entretanto ainda não há procedimentos para a execução das operações de junção por similaridade. Dessa maneira, se um atributo complexo é associado a um índice então o SIREN executa a seleção por similaridade utilizando a Slim-tree. Porém, as junções por similaridade são realizadas por meio de busca seqüencial.

3. Exemplos de utilização

O protótipo SIREN vem sendo desenvolvido em C++ para ambiente Windows e utiliza o protocolo ODBC para conexão com o SGBD Oracle 9i. Para a sua validação foi implementada uma aplicação web (*cgi*) em C++ que fornece um ambiente para a submissão de comandos em SQL.

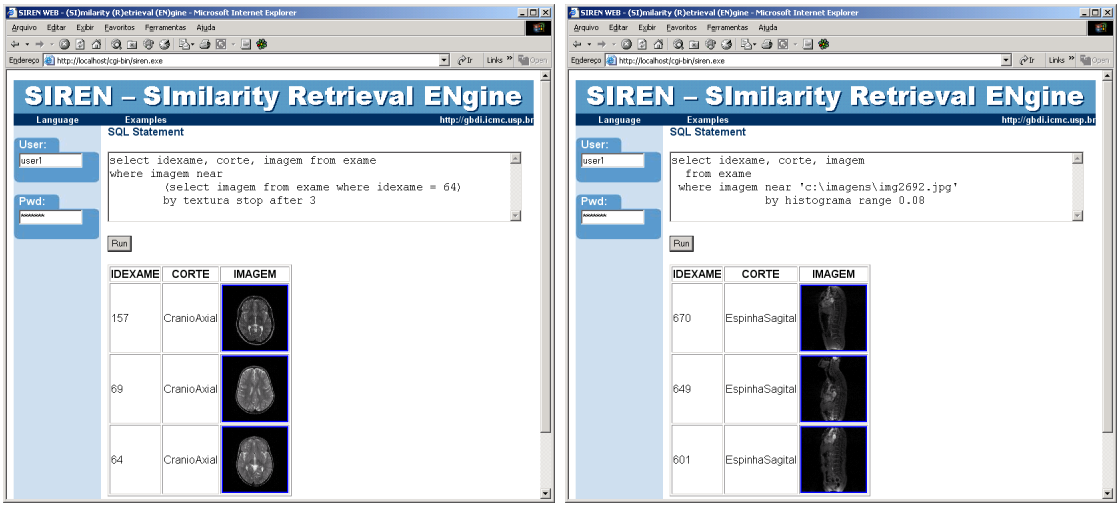
A seguir são apresentados exemplos de consultas por similaridade executadas pelo SIREN. Para exemplificar os dois novos tipos de dados definidos, foram utilizados dois conjuntos de dados. O primeiro conjunto, denominado Exame, é formado por 800 imagens de exames de tomografia computadorizada (CT) divididas em quatro classes: crânio axial, crânio coronal, crânio sagital e espinha sagital. Cada tupla é constituída por um identificador da imagem, a imagem, e um atributo que especifica a classe da imagem. Essas imagens podem ser consultadas de várias maneiras. Elas podem ser comparadas, por exemplo, pela similaridade da distribuição de suas cores ou de suas texturas. Para tanto devem ser definidas duas métricas, uma utilizando o extrator `HISTOGRAMEXT` e outra o extrator `TEXTUREEXT`. Os comandos utilizados para a criação das métricas e da tabela descritas acima são os seguintes.

```
CREATE METRIC histograma
  FOR STILLIMAGE (HISTOGRAMEXT
    (histogram AS histo))

CREATE METRIC textura
  FOR STILLIMAGE (TEXTUREEXT
    (texture AS text))
```

```
CREATE TABLE Exame (
  idexame INTEGER PRIMARY KEY,
  imagem STILLIMAGE METRIC USING
    (histograma DEFAULT, textura),
  corte CHAR(20))
```

Exemplos de consultas que utilizam cada uma dessas métricas são apresentados na Figura 2.



(a) (b)

Figura 2: Exemplos da execução de consultas por similaridade sobre a tabela Exame. (a) *k-NN query* usando a métrica textura. (b) *Range query* usando a métrica histograma.

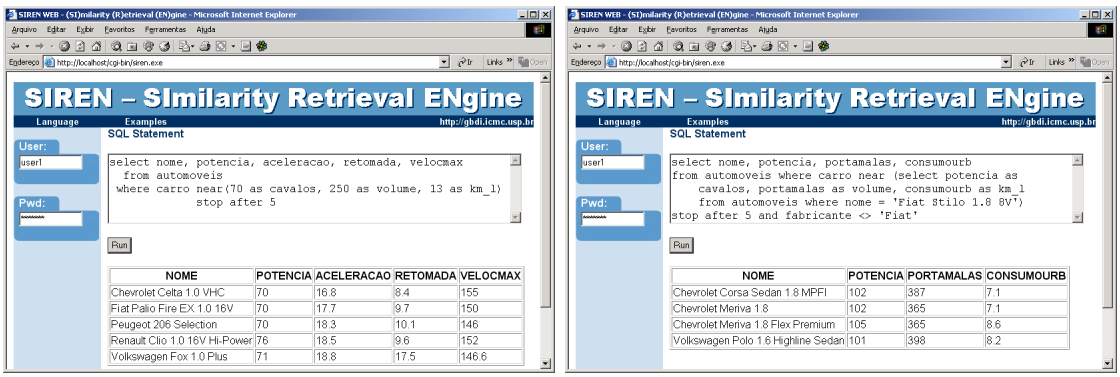
O segundo conjunto de dados usado, denominado Automóveis, é formado pelo resumo dos testes realizados pela revista Quatro Rodas, entre maio de 2001 e junho de 2005, com 140 carros de vários fabricantes [Revista Quatro Rodas, 2005]. Cada tupla desse conjunto possui o nome do modelo e do fabricante do carro testado além dos resultados dos seguintes itens avaliados: número de cilindros, potência em cavalos, aceleração de 0 a 100 km/h em segundos, retomada de 40 a 80 km/h em segundos, velocidade máxima em km/h, frenagem de 80 a 0 km/h em metros, ruído interno em dB, volume do porta malas em litros e consumo urbano em km/l. Consultas por similaridade podem ser empregadas para explorar esse conjunto de dados em diversas questões, por exemplo, a relação custo/benefício considerando itens como potência, volume do porta malas e consumo urbano. Os comandos que definem essa métrica e tabela são apresentados a seguir.

```
CREATE METRIC custobeneficio FOR PARTICULATE (
  cavalos NUMBER, volume NUMBER 0.5, km_l NUMBER 10)

CREATE TABLE Automoveis (
  nome CHAR(60) PRIMARY KEY,
  fabricante CHAR(15),
  cilindros NUMBER,    potencia NUMBER,    aceleracao NUMBER,
  retomada NUMBER,     velocmax NUMBER,    frenagem NUMBER,
  ruidointerno NUMBER, portamalas NUMBER, consumourb NUMBER,
  carro PARTICULATE METRIC REFERENCES (
    potencia AS cavalos, portamalas AS volume,
    consumourb AS km_l) USING (custobeneficio DEFAULT))
```

Considerando a métrica criada para analisar o conjunto de dados Automóveis é possível fazer consultas como as apresentadas na Figura 3.

O tempo médio de análise de um comando SQL pelo SIREN, incluindo a leitura do dicionário de dados, para uma tabela com um atributo `STILLIMAGE` é de 0.2 segundo e para uma tabela com um atributo `PARTICULATE` é de 0.14 segundo em um microcompudator de 2 GHz. Para a execução da consulta, o tempo de busca no MAM



(a) (b)

Figura 3: Exemplos da execução de consultas por similaridade sobre a tabela Automóveis. (a) "Quais são os 5 carros que apresentam as características mais próximas de 70 cavalos de potência, 250 litros de volume de porta malas e 13 km/l de consumo urbano?". (b) "Retorne os 5 carros mais próximos de um determinado carro e que não sejam do mesmo fabricante deste".

depende de fatores como: o tamanho das características ou o número de atributos indexados; o número de objetos complexos indexados; e o tempo de leitura de um bloco no disco rígido. Informações sobre o desempenho da Slim-Tree podem ser encontradas em [Traina-Jr. et al., 2002]. Como referência, o tempo total, incluindo a análise e execução das consultas da Figura 2(a) foi de 0.25 segundo e da Figura 3(a) foi de 0.17 segundo.

4. Conclusões

Este artigo apresenta a ferramenta SIREN, um interpretador para uma extensão à linguagem SQL, que permite a representação e execução de consultas por similaridade sobre dados complexos armazenados em SGBD relacionais. A ferramenta permite executar operações tanto de seleção quanto de junção baseadas na similaridade entre objetos de um tipo complexo. Estão definidos dois tipos de dados complexos: imagens e atributos descritos por um subconjunto dos atributos que formam cada tabela. A ferramenta permite executar tanto consultas simples quanto qualquer combinação das operações desenvolvidas entre si e entre as operações de seleção e junção tradicionais, provendo um meio poderoso de realizar consultas por similaridade em bases de dados complexos.

Referências

Barioni, M. C. N., Razente, H., Traina-Jr., C., and Traina, A. J. M. (2005). Querying complex objects by similarity in SQL. In *SBBD*, Uberlândia, MG. A ser apresentado.

Böhm, C. and Krebs, F. (2002). High performance data mining using the nearest neighbor join. In *IEEE-ICDM*, pages 43–50, Maebashi City, Japan.

Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. (2001). Searching in metric spaces. *ACM Computing Surveys (CSUR)*, 33(3):273–321.

GBDI-ICMC-USP (2004). GBDI Arboretum Library. http://gbdi.icmc.usp.br/arboretum/.

Long, F., Zhang, H., and Feng, D. D. (2002). Fundamentals of Content-based Image Retrieval. *Springer*.

Revista Quatro Rodas (2005). Resumo dos Testes. Editora Abril. http://quatrorodas.abril.com.br/carros/resumo/index.shtml, acessado em 28/06/2005.

Traina-Jr., C., Traina, A. J. M., Faloutsos, C., and Seeger, B. (2002). Fast Indexing and Visualization of Metric Datasets using Slim-trees. *IEEE TKDE*, 14:244–260.