社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－DBS－140（Ⅱ）（81）
2006／7／14

CREWデータモデルにおけるランキングクエリの実現

嶋村 勇志† 遠山 元道‡

†慶應義塾大学大学院 理工学研究科 開放環境科学専攻
‡慶應義塾大学 理工学部 情報工学科
E-mail: † shima@db.ics.keio.ac.jp, ‡ toyama@ics.keio.ac.jp

我々は地理情報検索において，既存のシステムの利用し，その上位レイヤーとして統一感のある操作系を提供するCREWを提案した．その検索には，少数の基本概念に基づくドメイン/マッピング型のGUIを用いる．本稿では，ドメイン/マッピング型クエリの処理において，ランクを導入することで，ユーザやシステムにおける出力結果の制御，及び中間処理データ量の削減によるパフォーマンスの向上を目指す．

キーワード：情報検索，地理情報システム，ランキング

Evaluating Ranking Queries on CREW Data Model

Takeshi SHIMAMURA † Motomichi TOYAMA ‡

†School of Science for OPEN and Environmental Systems,
Faculty of Science and Technology, Keio University.
‡Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.
E-mail : †shima@db.ics.keio.ac.jp ‡toyama@ics.keio.ac.jp

We enabled to rank output data within domain/mapping queries such as in CREW we have already proposed. CREW is a model for geographic information retrieval. It provides user with an intuitive retrieval with a few basic notion with our GUI. Ranking queries enables users and the system to control the output data and improves the query processing.

keyword : Geographic Information Retrieval , Data Model , Ranking

1 背景

近年、Google マップ [5] などに代表される地理情報検索を，単純な GUI を用いてウェブ上で簡単に行うことのできる Web 上のサービスが増加している．これらのサービスは直感的な操作をユーザに提供する一方で，汎用性の低いシステムであるといえる．そこで，大井ら [1] は既存のシステムを利用しそれらの上位モデルとして統一感のある操作系である CREW を提案した．

一方で，DB システムで扱われる膨大なデータの出力結果をランキングし，必要な量の情報だけを効果的に取り出すための研究も盛んに行われている [3, 4]．そこで，本稿では CREW にランクを導入し，地理情報システムにおいて不可欠である出力結果の制御，及びパフォーマンスの向上を目指す．

本稿の構成は次の通りである．まず 2 章において，CREW の概要と現状の問題点について述べる．次にそれらの問題を解決すべく 3 章において，本稿で提案する rank 関数の概要を説明する．次に 4 章でクエリ実行における高速化について説明し，5 章で評価・検討を行う．最後に 6 章で結論と今後の課題について述べる．

2 CREW とは

CREW[1] は，大井らによって提案された，地理情報検索において Google やリレーショナル DBMS など既存システムの利用を前提とし，それらの上位レイヤーとして統一感のある操作系である．CREW は空間・実体・Web の 3 つのドメインと関数群から成るモデルであり，これらを組み合わせることで，構成的な問い合わせを可能にする．また，ドメインと関数という少数の基本概念に基づく直感的なクエリ生成を行う，下位レイヤーに依存しないインターフェースを提供した．これらの CREW データモデルと GUI を合わせて CREW と呼ぶ．

2.1 ドメイン

CREW は空間・実体・Web の 3 つのドメインを持つ．各ドメインは要素の集合から成り，空間の集合を C，その要素を c と表現する．実体・Web についても同様に， $e \in E$ ， $w \in W$ と表現する．

2.2 関数

ドメイン間のマッピングの表現には 2 種類の関数を用いる．内部関数 (intra domain function) 及び，遷移関数 (inter domain function) である．内部関数は，演算がそのドメイン内で実行される関数である．例えば，Entity ドメインの内部関数とは，引数として，Entity ドメインの要素または集合，あるいは基本型の値をとり，出力結果も基本型，または Entity ドメインの要素，集合である関数である．これに対して遷移関数は，入力と出力が異なるドメインであるような関数である．

CREW における関数はユーザから見た分類は，大きく分けて，内部関数・遷移関数の二通りである．内部的な処理をユーザが意識する必要がないが，次項ではシステム側からみた関数の分類法を示す．

2.2.1 処理による分類

システム側の関数の処理の観点から，関数は 3 つの分類することが可能である．

データ参照 (DR: Data Referring) 関数
: あるドメインの要素を入力とし，他のドメインの値を出力とする関数など，計算処理を伴わない関数である．例として，与えられた領域内に含まれる Entity を返す within(C) ⇒ E 関数等が挙げられる．

内部処理 (IP: Internal Processing) 関数
: 計算処理を伴う関数のうち，処理が自システム内で完結する関数である．例えば，2 点間のユークリッド距離を計算する $euclid(point_1, point_2)$ や，二つの領域の和集合を求める $uni\_area(C_1, C_2)$ 等である．

外部処理 (EP: External Processing) 関数
: 計算処理を伴う関数のうち，処理をシステム外に依存する処理を担当する関数である．例えば，Google を用いることで，Web に対してキーワードを与え Web ページの集合を返す関数である $search\_web(strQuery)$ 等である．

2.2.2 入力・出力による分類

CREW における多くの関数は，ユーザから見て，多値入力・多値出力の関数である．しかし，実際の処理段階では，関数は入力・出力の観点から次の 4 つの関数に分類することができる．

(1) 単値入力/単値出力
(2) 単値入力/多値出力
(3) 多値入力/単値出力
(4) 多値入力/多値出力

例えば遷移関数 exist(E) ⇒ C は，Entity の集合 E が存在する領域の集合 C を返す関数であるが，実際の処理は，各要素 e に対応する領域 c を求め，それらの和領域を計算し返す．つまり，単値入力・多値出力関数で，それを E の各要素に対して計算しているといえる．

1. 明示的入力 (implicit input) 関数
   明示的入力関数とは，関数への入力データだけで処理が完結する関数である．集合の和を計算する *union* 関数や，与えられた領域の中心座標を返す *centroid* 関数などがある．

2. 暗示的入力 (explicit input) 関数
   暗示的入力関数とは，関数へ入力されたデータの他に．例えば，*within* 関数は，入力された領域内に含まれる Entity を返す関数であるが，これは全 Entity の中から，指定された領域に含まれる Entity を選ぶ選択演算であるため，入力として，全 Entity をシステムが暗示的に入力すると考えられる．このような関数を暗示的入力関数と呼ぶ．

2.3 GUI

大井らは CREW によるシステムの実装例として，図 1 の様な GUI を実装した．

1 つのアイコンが 1 つの関数に対応し，アイコン間を結ぶコネクタが関数間のデータフローを表す．アイコンの左側に付いてる結合部が入力（引数）部であり，右側に付いている結合部が出力部に当たる．1 つの出力部から複数の入力部に対してコネクタを結ぶ事も可能であるが，コネクタの両端のドメインは同じでなくてはならない．

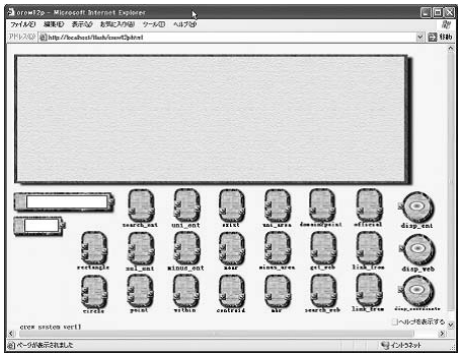

図 1: GUI

このように，CREW では，ドメインとマッピングという少ない基本概念に基づく，下位レイヤーに依存しないインターフェースを提供している．

2.4 問題点

CREW の問題点として次の 3 つが挙げられる．

1. ユーザが任意の重要度によって出力結果を並び替えたいという要求に応えられない．
2. データ数が多くなったときの出力結果の調節ができない．
3. 検索速度が遅い．

1. はユーザによる，2. はシステムによる出力結果の並び替え，及び出力件数の調節を可能にすることによって解決できる．3 章で説明する，rank 関数を実装することでこれらの問題を解決した．また，3. の問題は，関数への出入力を制限することで解決できると考える．4 章において，クエリの高速化について述べる．

3 rank 関数

3.1 rank 関数

rank 関数は，ソート関数 $\mu(a)$ によって計算されるスコアの順に各要素 $a \in A$ を昇順または降順に並び替え，上位 k 件の要素を出力するものである．

3.2 定義

rank 関数は以下の様に定義される．

$A^k_{\mu(a)} = rank(A, k, \mu(a), dir)$
A: 入力集合 $a \in A$
k : 出力件数 (省略可)
$\mu(a)$:ソート関数
dir : ソート方向 $< ASC|DESC >$ (省略可)

ここでソート関数 $\mu(a)$ とは，各要素 $a \in A$ に対し，基本型 (数値型，文字列型) を出力する関数で，ランク付けの基準となるスコアを計算する単値入力/単値出力関数である．

通常 $|A^k_{\mu(a),dir}| = k$ であるが，$|A| < k$ のときは，$A^k_{\mu(a)} = |A|$ になる．

また，トップ k に入る要素が k 個以上ある場合，つまり $\mu(a)$ によるスコアが同じものがある場合には，そのいずれかが入れば良いものとする．つまり，rank 関数による出力結果は一意ではない．これは，文献 [3] などにおいてもでも同様である．

3.3 ユーザによる rank 関数の利用

rank 関数を利用したクエリ例を示す．次のクエリは映画館を表す Entity を，ユーザの現在位置 ($HERE$) から近い順に 10 件出力する，というクエリである．

$disp\_ent($
  rank(
    $search\_ent(E), 10,$
    $euc\_dist(search\_ent(E), HERE)$
)
)

E = search_ent(category = ' 映画館')

この検索式は GUI では図 2 のように表現される．

ここで，HERE はユーザの現在位置の座標とする．このように，rank 関数を用いることで，2.4 節で挙げた問題点 1 のユーザが任意の重要度によって出力結果を並び替えて出力したいという要求に応えることができる．

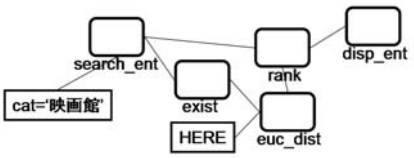

図 2: rank 関数を用いた検索例

図 3: CREW の出力結果

3.4 システムによる rank 関数

検索結果のデータ数が多くなると，システムはその出力結果をなんらかの形で制限して提供する必要がある．例えば、google では，検索結果が多かった場合，1 ページあたり 50 件に分割してユーザに提供される．CREW [1] では，$display\_entity$ 関数を用いた結果出力に図 3 に示すような Google Maps API を用いた実装例を示している．出力データ数が多くなりすぎると，アイコンが地図いっぱいになってしまい，検索結果が分かりづらくなってしまうことは容易に想像がつく．

そこで，$display\_entity$ に際して，次のような関数をシステム側で付加することにより，出力結果数の調整及び，出力結果の指定が可能になる．

$display\_coordinates($ $mbr(exist(rank($
ユーザクエリ$, k,$ システム条件$))))$

ここでユーザクエリとは，ユーザが入力したクエリであり，システム条件とは，システム管理者が何を基準に出力結果を制限するかの条件であるソート関数のことである．これらの関数をシステム管理者が

設定することで，2.4 節で挙げた問題点 2 を解決できる．

4 高速化

4.1 rank 関数の評価

4.2 ソートエッジ

AQuery[2] では，クエリの最適化手法におけるソーティングのアルゴリズムとして，sort-edge を用いている．rank 関数の処理ではバブルソートにこの sort-edge の考え方を加えた s 手法を用いる．

$A^k_{\mu(a)} = rank(A, k, \mu(a), dir)$ の処理には，まず大きさ k の配列 K[] を用意し，その中では，ソート関数 $\mu(a)$ の従って，昇順 (降順) にデータが並べておく．$K[k] > a(K[k] < a)$ ならば，K[k]=a とし，バブルソートによって K[] 内でソートを行う．これによって，ソートにおいて，比較回数を大きく改善することができる．

4.3 プリフィルタリング

プリフィルタリングとは，ユーザやシステムによって与えられたドメイン/マッピング型のクエリに対して，中間処理にプリフィルタをシステム側で挿入することにより，ドメイン間を遷移するデータフローを小さくし，計算量を少なくする手法のことを指す．

2.2.2 節で述べたように，関数には，明示的な入力を持つ (II 関数) 関数と，暗示的な入力を持つ関数 (EI 関数) が存在する．暗示的な入力は，データ量がそのドメインの値全てと大きいことに注目した．このような入力に対し，フィルタを挿入することで得られる効果は大きい．そこで，次の様にプリフィルタを生成，挿入する．

f_1 を EI 関数とし，その明示的な入力を II，暗示的な入力を EI とする．f_2 を EI と同じドメインにおける内部関数であるとする．

$f_2(f_1(II, EI))$ のような場合，$f_2(f_1(II, f_2(EI)))$ のように，暗示的な入力に対しフィルタを挿入することができる．

このようなルールを追加していくことで，中間計算量を削減し，処理を高速化することが可能であると考えられる．

5 評価・検討

本章では，主な Web 上における地理情報検索システムにおける操作を CREW において表現することで，CREW の表現力を示した．評価対象として，NTT Open Lab の提供するジオリンク京都 [6] と Google の提供する Google マップ [5] を用いた．

5.1 ジオリンク京都

ジオリンク京都は，地域対象を京都市内とし，その中に各施設を地図上に表示し，関連付けられた Web ページを検索することができる．

可能な操作と CREW における表現について以下に考察した．

分類による検索 あらかじめ，各施設等は買物，食事，観光などカテゴリ別に分類されており，画面左のチェックボックスをクリックすることで 1 つのカテゴリのみの表示を行うことができる．

CREW による表現：各施設を Entity とし，その属性としてカテゴリを持たせることで，$disp\_ent(search\_ent(category =' 買物'))$ で容易に実現できる．

画面の拡大 ジオリンク京都では，画面上方の 2 つのタブによって、2 段階の縮尺で地図を表示することが可能である．縮尺による表示件数の変化はない．

CREW による表現：タブによる縮尺の大きさは固定的である．現在の画面の中心座標を (c_1, c_2) とし，縮尺に応じて表示する範囲の南北の距離を l，東西の距離を w とすると，$disp\_coordinates(\ rectangle(\ point(c_1 - \frac{w}{2}, c_2 - \frac{h}{2}),\ point(c_1 + \frac{w}{2}, c_2 + \frac{h}{2})\ ))$ で表示範囲を指定すれば良い．

キーワード検索 キーワードにより各施設を絞り込むことができる．タイトル検索・全文検索の 2 通りが用意されている．

CREW による表現：全文検索では，あらかじめ，各施設の情報としてその説明を属性として含んでいると考えられる．その属性を explantion とすると，全文検索は

$$disp\_ent(search\_ent(\ explanation\ LIKE\ \%KEYWORD\%\ ))$$

に他ならない．またタイトル検索も同様で，

$$disp\_ent(search\_ent(\ name\ LIKE\ \%KEYWORD\%\ ))$$

の操作と等価である．出力結果の分布に応じて地図の表示範囲を調整する方法については，3.4 節で述べた通りである．

Web ページ表示 各施設のアイコンをクリックすることで，あらかじめ対応付けられている Web ページを表示することができる．ここで Web ページは，公式サイトの他，グルメ情報サイトの情報等である．

CREW による表現：$disp\_web(official(e))$ と等しい操作である．ここで e とはクリックされた施設を指す Entity である．

5.2 Google マップ

Google マップでは，地図とキーワードによる連携した検索を行うことができる．また，その検索技術を活かしキーワードによる施設の検索を行うことができる．

地図の拡大・縮小 Google マップでは最大 18 段階の縮尺による表示が可能であり，地図左上のスライドバーを用いて調節する．縮尺は固定的である．

CREW による表現：領域の範囲指定はジオリンク京都と同様に実現可能である．

地図のスクロール 地図をスクロールすることで，直感的に範囲指定をすることが可能である．

CREW による表現：範囲指定において，範囲の対角頂点の座標を $rectangle(point1, point2)$ 関数を用いる．

キーワードによる地図検索 住所や地名を入力すると，該当する範囲を含む領域の地図を適切な縮尺で表示することが可能．ただし，ここで地名以外のキーワードを指定した場合は，次に示すお店やサービスの検索に切り替わる．

CREW による表現：住所や地名を位置情報（座標等）に変換するデータベースの利用前提とし，住所や地名から求められた領域を c とする．display_ent(mbr(c)) により，適切な範囲を表示することが可能である．

キーワードによるお店やサービスの検索 表示されている地図の範囲内において，お店やサービスのキーワード検索を行うことができる．検索の際には，Google 独自のランキング手法により，検索結果を制限し，一度に 10 件のデータがリスト及び地図上にアイコンとして表示される．検索結果には，住所・電話番号の基本情報に加えて，関連 Web サイトも表示される．範囲の指定には，地図による範囲指定の他に,「銀座」などのキーワード，住所による指定の他，範囲を指定しないでキーワード検索を行うことも可能である．この際，アイコンの分布に応じて，地図の縮尺が調整される．

CREW による表現: ここで，お店などの Entity のキーワードによるランク付けを行う関数を $google\_ranking(E, keyWord)$ とする．

$$disp\_ent(\ rank(E, 10, google\_ranking(E, keyWord))\ )$$
$$E = within(rectangle(d_1, d_2)))$$

ここで d_1, d_2 は表示されている地図の対角頂点の座標である．また，範囲を指定しないでキーワード検索を行った場合の地図の表示範囲を調整する手法については，3.4 節で述べた通りである．

以上の結果から，rank 関数を定義することで，既存の Web 上の地理情報検索サービスにおける操作の多くを表すことが可能であることを示した．

6　結論・今後の課題

本稿では，CREW においてランキングクエリを実現する方法について述べた．rank 関数を定義し，その実行例を示すことでその有用性を示した．また，rank 関数の処理方法や，プリフィルタリングの手法を用い，クエリ処理の高速化の可能性について述べた．最後に既存のシステムとの表現力の比較を行うことで，CREW の検索式の網羅性を示した．

今後は，プリフィルタリングを行う際のルールについて検討し，その有用性を示すための実験を行っていく予定である．

参考文献

[1] 大井 峻, 遠山 元道: CREW データモデルに基づく空間・実体・Web データの横断的表現と検索. 電子情報通信学会大 17 回データ工学ワークショップ (DEWS2006) 論文集, 2006.

[2] A. Lerner and D. Shasha. AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments. *Proceedings of the 29th VLDB Conference, Berlin*, pages 345-356, 2003

[3] S.Chaudhuri and L.Gravano. Evaluating top-k selection queries. In *VLDB*, pages 397-410, 1999.

[4] C. Li, K.C. Chang, I.F. Ilyas, and S. Song. RankSQL: Query Algebra and Optimization for Relational Top-k Queries. In *SIGMOD*, pages 131-142, 2005

[5] Google: Google マップ, http://local.google.com/.

[6] NTT Open Lab: ジオリンク京都, http://www.digitalcity.gr.jp/openlab/kyoto/map_guide_j.html.