Information Sciences 211 (2012) 18–35

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Interactive skyline queries ☆

Jongwuk Lee [a], Gae-won You [a], Seung-won Hwang [a,*], Joachim Selke [b], Wolf-Tilo Balke [b]

[a] Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea
[b] Institut für Informationssysteme, Technische Universität Braunschweig, Braunschweig, Germany

ARTICLE INFO

Article history:
Received 22 October 2009
Received in revised form 29 November 2010
Accepted 2 April 2012
Available online 24 April 2012

Keywords:
Skyline query
Preference elicitation

ABSTRACT

When issuing user-specific queries, users often present vague and imprecise information needs. Skyline queries with an intuitive query formulation mechanism identify the most interesting objects for *incomplete* user preferences. However, the applicability of skyline queries suffers from a severe drawback because incomplete user preferences often lead to an impractical skyline size. To address this problem, we develop an *interactive preference elicitation* framework – while user preferences are collected at each iteration, the framework iteratively updates skylines. In this process, the framework aims to both minimize user interaction and maximize skyline reduction size, while the query formulation is still intuitive. All that users need to do is thus to answer a few well-chosen questions generated from the framework. We validate the effectiveness and efficiency of our framework in extensive experimental settings, and demonstrate that a few questions are enough to acquire a skyline with a manageable size.

1. Introduction

Querying in databases and information systems has evolved beyond SQL-style exact match queries. Modern query languages and processing algorithms account for vague and imprecise information needs. As prime techniques, top-k retrieval and skyline queries have been recently introduced.

In top-k retrieval, a query consists of an integer k and a *utility function*, which assigns a numerical score to each object. The database system returns a ranked list of k objects scoring the highest. Thus, top-k queries always provide a focused and manageable result set, but users usually find it difficult to come up with a utility function that resembles their personal preferences.

In contrast, skyline queries do not require users to specify a single all-inclusive utility function. A skyline query consists of a set of utility functions, each of which resembles a single aspect of object quality. Typically, each utility function possesses a simple structure, e.g., a sorted list, and can come with natural preference order. In other words, each utility function can directly reflect a *qualitative* user preference on a single attribute defined in the database's relational schema.

The skyline query is to return a set of *Pareto-optimal* objects, called a *skyline*. Specifically, an object a can be a *skyline object* if there exists no other object b dominating a – an object a is said to *dominate* another object b if a scores better than b with respect to at least one utility function, and a does not score worse than b with respect to any other utility functions. Note that only qualitative order induced by each utility function matters for the skyline query, which makes it more intuitive.

The following example illustrates a typical skyline query.

☆ This paper is based on and significantly extends preliminary work [31] presented at DEXA 2008.

* Corresponding author.
E-mail address: swhwang@postech.ac.kr (S.-w. Hwang).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.04.007

Table 1
A toy dataset in Examples 1 and 2.

ID	Type	Color	Brand	Price	Speed
1	Convertible	Red	Ferrari	80	190
2	Sedan	Blue	Ferrari	150	160
3	Convertible	Blue	Honda	100	160
4	Sedan	Red	Honda	70	200
5	Roadster	Blue	Honda	200	150

Example 1. Consider a customer named Alice shopping for a car that should be optimal with respect to five attributes: *type*, *color*, *brand*, *price*, and *speed* (Table 1). Alice prefers convertibles to sedans, sedans to roadsters, red cars to blue ones, and Ferraris to Hondas. Also, price and speed are naturally ordered by "the cheaper the better" and "the faster the better" respectively. Her preferences can thus be represented by "*type*: convertible $\succ$ sedan $\succ$ roadster; *color*: red $\succ$ blue; *brand*: Ferrari $\succ$ Honda; *price*: minimize, *speed*: maximize", where $\succ$ denotes a *dominance* relationship. We can rule out cars 2, 3, and 5, because they are all dominated by car 1. The remaining cars 1 and 4 make up skyline objects.

The skyline query in Example 1 is useful for identifying objects satisfying user preferences, but it does not scale to attributes having a large domain without natural order. In that case, users are required to define large preference orderings within their queries, which is very demanding and tedious. Thus, this manual preference elicitation for *complete* user preferences can make the paradigm of skyline query useless.

To overcome this problem, we allow users to present skyline queries with *incomplete* user preferences, as illustrated by the following example.

Example 2. Alice is shopping for a car, but now she solely specifies her incomplete preferences on some attributes. There are several ways to leave out some preference information as follows: (1) "*type*: convertible $\succ$ sedan; *color*: red $\succ$ blue; *brand*: Ferrari $\succ$ Honda; *price*: minimize, *speed*: maximize". The skyline is cars 1, 4, and 5, because her preferences do not offer whether roadsters are better than convertibles or sedans; (2) another query with incomplete preferences is "*type*: convertible $\succ$ sedan $\succ$ roadster; *color*: ?; *brand*: Ferrari $\succ$ Honda; *price*: minimize, *speed*: maximize", where? is no preference information. The incomplete preferences yield cars 1–4 as the skyline.

Example 2 implies that a large number of different skyline queries can involve incomplete preferences. Clearly, the user want to get the skyline whose size is as small as possible, while providing as little preference information as necessary. In practical scenarios, the amount of preference information available at query time is usually limited. On the other hand, when the system requires the user to ask complex or complete preferences before querying, the user often need to make considerable efforts to specify her preferences in sufficient detail.

A solution to this problem would be to provide only the "most informative" preferences within the query. However, because the user does not know what objects are contained in the database system, it is impossible for the user to choose informative preferences. On the other hand, the database system knows the objects stored well.

In this paper, we thus study how the user and the database system can refine skyline queries involving attributes without natural order in an interactive way. Specifically, we devise a preference elicitation framework, which not only minimizes user interaction but also maximizes skyline reduction size. Because user-specific preferences are not known *a priori*, our proposed framework is based on a *probabilistic* approach for *missing knowledge* about user preferences, and iteratively requires the user to elicit her preferences.

To summarize, this paper makes the following contributions:

- We state a problem definition for our framework that captures the notion of informative preferences (Section 2).
- We propose an effective elicitation framework and discuss its efficient implementation (Section 3).
- We extend our framework for general datasets including both attributes with and without natural order (Section 4).
- We evaluate the effectiveness and efficiency of the proposed algorithms (Section 5).

2. Problem statement

This section states preliminaries to address the preference elicitation problem. We first introduce notations used to define preference queries, skyline queries, and preference elicitation. Table 2 summarizes the notations used in this paper.

2.1. Preference queries

Preference queries are commonly used to retrieve the "best" tuples. Throughout this paper, we deal with a relational schema that consists of d attributes, i.e., $A_1, A_2, \ldots, A_d$. The schema's domain is denoted by $\mathcal{D}$, i.e., $\mathcal{D} = D_1 \times D_2 \times \cdots \times D_d$, where D_i is the domain of attribute A_i. A dataset $\mathcal{O}$ is a finite subset of $\mathcal{D}$, i.e., $\mathcal{O} \subseteq \mathcal{D}$. As usual in databases, $\mathcal{O}$ is a multiset, implying that $\mathcal{O}$ could contain duplicate tuples.