Databases and Information Systems VIII
H.-M. Haav et al. (Eds.)

doi:10.3233/978-1-61499-458-9-159

A Survey on Selectivity Estimation for Preference Database Queries

Markus ENDRES [a,1]
[a] *University of Augsburg, Institute for Computer Science, Germany*

Abstract. Skyline query processing and the more general preference queries become reality in current database systems. Preference queries select those tuples from a database that are optimal with respect to a set of designated preference attributes. In a *Skyline query* these preferences only refer to *minimum* and *maximum*, whereas the more general approach of *preference queries* allow a more granular specification of user wishes as well as the specification of the relative importance of individual preferences. The incorporation of preferences into practical relational database engines necessitates an efficient and effective selectivity estimation module: A better understanding of the preference selectivity is useful for better design of algorithms and necessary to extend a database query optimizer's cost model to accommodate preference queries. This paper presents a survey on selectivity and cardinality estimation for arbitrary preference queries. The paper presents current approaches and discusses their advantages and disadvantages, such that one could decide which model should be used in a database engine to estimate optimization costs.

Keywords. preference queries, skyline, selectivity, cardinality, optimization

Introduction

Preferences in databases – as shown by a recent survey [1] – are a well established framework to create personalized information systems. By using well designed preference models, users can be provided with just the information they need, thereby overcoming the dreaded empty result set and flooding effect [2,3]. Traditional database engines or query languages do not support preference queries. However, in the last decade some extensions to the SQL language have been proposed, such as *Skyline queries* [4,5].

Skyline queries are not outside the expressive power of SQL. However, it is cumbersome to write Skyline-like queries in SQL. Therefore, the `SKYLINE OF` clause was introduced by [4] as a useful syntactic addition to SQL. Its basic form is as follows:

```
SELECT ... FROM ... WHERE ...
  GROUP BY ... HAVING ...
  SKYLINE OF A1 [min | max | diff], ...,
             An [min | max | diff];
```

[1]Corresponding Author: Markus Endres, Institute for Computer Science at the University of Augsburg, 86135 Augsburg, Germany; E-mail: endres@informatik.uni-augsburg.de.

The Columns `A1, ..., An` are the attributes over which the preferences apply. Their domains must have a natural total ordering, such as integers. The directives `min` and `max` specify whether one prefers low or high values, respectively. The `diff` states that one wants to retain the best choices with respect to each distinct value of that attribute.

A more general approach of Skylines are *preference queries*, which allow a more granular specification of user wishes. *Preference SQL* is an extension to SQL to specify such arbitrary user preferences [6]. Preferences in the sense of [6] are strict partial orders and a preference query returns the maximal elements according to this order, also called the *Best-Matches Only set* (BMO-set). For example, the wish for a car having the highest power and a price around $ 34000 can be expressed in Preference SQL as follows:

```
SELECT id, power, price FROM car
PREFERRING power HIGHEST AND price AROUND 34000;
```

The `PREFERRING` keyword introduces a *preference*. The connection of two preferences by `AND` is the *Pareto composition* and states the equal importance of preferences. If we restrict the attention to LOWEST (`min`) / HIGHEST (`max`) as input preferences, then Pareto preference queries coincide with the traditional *Skyline queries* above.

A preference or Skyline query returns only those tuples from the dataset for which there is no other tuple that is better with respect to all preferences. Assume the sample dataset in Table 1. Then the above query would identify the tuples with ID 3 and 5 as best objects. The tuple with ID 3 has a perfect match concerning the price, but the power of the tuple with ID 5 is higher. Therefore, both tuples are indifferent and form the result.

Table 1. Sample dataset of cars.

car	**id**	**make**	**power**	**price**
	1	BMW	180	35000
	2	Mercedes	200	38000
	3	BMW	230	34000
	4	Audi	170	32000
	5	Mercedes	250	20000

While preferences and Skylines can be expressed in standard SQL using a rewriting technique, it is widely recognized that an efficient implementation requires introducing an operator inside the database engine for computing the best objects [4,7]. Several physical implementations for the *Skyline operator* and the more general Pareto preference have been proposed, e.g. [4,6,8,9]. However, the integration of special Skyline or preference operators into relational database engines also require heuristic optimization rules [3,10] as well as *cost and cardinality estimation*. Following [11] a better understanding of the selectivity of Skyline and preference queries *is useful for a better design of algorithms* and *is indispensable for a good query optimizer's cost model to accommodate preferences*. Cost-based query optimizers generate a number of equivalent query plans, use query size estimates to approximate query plan costs and select one which is (nearly) optimal. Thereby the cost of the implementation algorithms is determined by the size of the input arguments. Therefore, it is crucial to know, compute, or estimate the sizes of arguments.

This paper is a survey on selectivity and cardinality estimation of Skyline and preference database queries. We introduce the background on Skylines and preference queries in Section 1. We consider the state of the art in cardinality estimation for Skyline queries in Section 2, and in Section 3 we present selectivity estimation methods for arbitrary preference database queries. In Section 4 we conclude this paper.

1. Background on Skyline and Preference Database Queries

In this section, we review the definition of *Skyline* and *preference database queries*.

Definition 1 (Skyline) *Assume a set of vectors $D \in \mathbb{R}^d$. The Skyline is the subset of all points that are not dominated:*

$$S = \{t \in D | \neg \exists t' \in D : t \prec t'\} \tag{1}$$

A point t' dominates t, written $t \prec t'$, iff

$$\forall j \in \{1,\ldots,d\} : t_j \leq t'_j \ \wedge \ \exists i \in \{1,\ldots,d\} : t_i < t'_i$$

Thus, the Skyline consists of points where no point exists that is at least as good in all attributes and strictly better in at least one attribute. Note that Definition 1 only holds for *maximum* (analogously *minimum*) preferences [4].

The more general *preference queries* allow a more granular specification of user wishes as well as the specification of the relative importance of these preferences [2,12, 3]. Thereby we follow the preference model of [2,12].

Definition 2 (Preference) *Let R be a relational table with a list of attributes $\{A_1,\ldots,A_n\}$ and their domain* $\mathrm{dom}(A_i)$. *A preference $P = (A, <_P)$ is a strict partial order on the attribute list A, where $A \subseteq \{A_1,\cdots,A_n\}$. The term $\mathbf{x} <_\mathbf{P} \mathbf{y}$ is interpreted as "I like y more than x". Two tuples x and y are* indifferent, *if $\neg(x <_P y) \ \wedge \ \neg(y <_P x)$, i.e., neither x is better than y nor y is better than x.*

The result of a preference is computed by the *preference selection operator* [2], also called *winnow* in [3].

Definition 3 (Preference Selection, BMO-set) *The result of a preference query is called* Best-Matching-Only *(BMO) set and is computed by the* preference selection operator $\sigma[P](R)$*:*

$$\sigma[P](R) := \{t \in R \mid \neg \exists t' \in R : t[A] <_P t'[A]\} \tag{2}$$

The BMO-set contains all tuples t from an input relation R which are not dominated w.r.t. the preference P. A preference can also be evaluated in grouped mode, given an attribute list B:

$$\sigma[P \text{ grouping } B](R) := \{t \in R \mid \neg \exists t' \in R : t[A] <_P t'[A] \ \wedge \ t[B] = t'[B]\} \tag{3}$$

Obviously, Eq. (1) and Eq. (2) are equivalent when using a numerical minimum or maximum preference. In this sense Skylines are a specialization of preference queries.

To specify a database preference, a variety of intuitive preference constructors have been defined, cp. [6]. Preferences on single attributes are called *base preferences*. There are *base preference constructors* for *discrete* (*categorical*) and for *continuous* (*numerical*) domains. Most of the base preferences can be specified by a SCORE function.

Definition 4 (SCORE$_d$ **Preference)** *Given a scoring function* $f : \mathrm{dom}(A) \to \mathbb{R}_0^+$*, and some* $d \in \mathbb{R}_0^+$*. Then P is called a* SCORE$_d$ *preference, iff for* $x, y \in \mathrm{dom}(A)$*:*

$$x <_P y \iff f_d(x) > f_d(y)$$

where $f_d : \mathrm{dom}(A) \to \mathbb{R}_0^+$ *is defined as:* $$f_d(v) := \begin{cases} f(v) & \text{if } d = 0 \\ \left\lceil \frac{f(v)}{d} \right\rceil & \text{if } d > 0 \end{cases}$$

In the case of $d = 0$ the function $f(v)$ models the *distance* to the best value. That means $f_d(v)$ describes how far the domain value v is away from the optimal value. A value $d > 0$ represents a discretization of $f(v)$, which is used to group ranges of scores together. The d-parameter maps different function values to a single number. Choosing $d > 0$ effects that attribute values with identical $f_d(v)$ value become substitutable.

There are several sub-constructors of SCORE$_d$, for example the interval preference BETWEEN$_d(A, [low, up])$ expresses the wish for a value between a *lower* and an *upper* bound. If this is infeasible, values having the smallest distance to $[low, up]$ are preferred, where the distance is discretized by the discretization parameter d. The scoring function for example is $f(v) = \max\{low - v, 0, v - up\}$. Specifying $low = up$ $(=: z)$ we get the AROUND$_d(A, z)$ preference, where the desired value should be z. Furthermore, the constructors LOWEST$_d(A, \inf)$ and HIGHEST$_d(A, \sup)$ prefer the minimal and maximal values within the distance d, where inf and sup are the infimum and supremum of the attribute values. In the ATLEAST$_d(A, z)$ preference the desired values should be greater or equal to z. If this is infeasible, values within a distance of d are acceptable. ATMOST$_d(A, z)$ is its dual preference.

In a categorical domain the LAYERED$_m(A, \{L_1, \ldots, L_m\})$ preference expresses that a user has a set of preferred values given by the disjoint sets L_i, which form a partition of $\mathrm{dom}(A)$. Thereby the values in L_1 are the most preferred values. If they are not available in the database, than the alternative values listed in L_2 are the second choice, and so on. The scoring function equals $f(v) = i - 1 \iff x \in L_i$. For convenience there are several sub-constructors of LAYERED$_m$. The positive preference POS$(A, \mathit{POS\text{-}set}) :=$ LAYERED$_2(A, \mathit{POS\text{-}set}, \mathrm{dom}(A) \setminus \mathit{POS\text{-}set})$ expresses that a user has a set of preferred values given by the *POS-set*. The negative preference NEG$(A, \mathit{NEG\text{-}set})$ is the counterpart to the POS preference. It is possible to combine these preferences to POS/POS or POS/NEG, cp. [2].

If one wants to combine several preferences into more *complex preferences*, one has to decide the relative importance of these given preferences. Intuitively, people speak of "this preference is more important to me than that one" or "these preferences are all equally important to me". Equal importance is modeled by the so-called *Pareto preference*, whereas for ordered importance we introduce prioritization.

Definition 5 (Pareto) *In a* Pareto preference $P := P_1 \otimes P_2 = (A_1 \times A_2, <_P)$ *all preferences* $P_i = (A_i, <_{P_i})$ *on the attributes* A_i *are of equal importance, i.e., for two tuples* $x = (x_1, x_2)$, $y = (y_1, y_2) \in \mathrm{dom}(A_1) \times \mathrm{dom}(A_2)$ *we define:*

$$(x_1, x_2) <_P (y_1, y_2) \iff$$
$$(x_1 <_{P_1} y_1 \wedge (x_2 <_{P_2} y_2 \vee x_2 = y_2)) \vee (x_2 <_{P_2} y_2 \wedge (x_1 <_{P_1} y_1 \vee x_1 = y_1))$$

If we restrict the attention to LOWEST/HIGHEST as input preferences, then Pareto preference queries coincide with the traditional *Skyline queries*, cf. Definition 1.

Definition 6 (Prioritization) *The* Prioritization preference *allows the modeling of combinations of preferences that have different importance. Assume preferences* $P_1 = (A_1, <_{P_1})$ *and* $P_2 = (A_2, <_{P_2})$*, then prioritization denoted by* $P := P_1 \,\&\, P_2$ *is defined as:*

$$(x_1, x_2) <_P (y_1, y_2) \iff x_1 <_{P_1} y_1 \vee (x_1 = y_1 \wedge x_2 <_{P_2} y_2)$$

Note that a generalization of Pareto and Prioritization to more than two preferences is straight forward, cp. [6].

Example 1 *In this example we refer to the dataset in Table 1. The wish for an Audi or BMW leads to a* POS *preference* $P_1 := \mathrm{POS}(\mathrm{make}, \{\mathrm{Audi}, \mathrm{BMW}\})$. *The preference selection* $\sigma[P_1](\mathrm{car})$ *leads to the cars with IDs {1,3,4}.*

If we prefer a horsepower around 170, where a difference of 5 does not matter, i.e. $d = 5$*, this can be expressed by* $P_2 := \mathrm{AROUND}_5(\mathrm{power}, 170)$. *The result is given by the tuple with ID 4 because it has a perfect match of 170 hp.*

Now let P_1 *and* P_2 *be equally important. Then we construct a Pareto preference* $P_3 := P_1 \otimes P_2$. *The result of* $\sigma[P_3](car)$ *is the tuple* $(\mathrm{Audi}, 170)$.

There are several other complex preference constructors which are essential in rule based preference query optimization, cp. [2,12].

Definition 7 (Special Preferences)

a) A preference $P = (A, <_P)$ *is an* Antichain *iff* $<_P = \emptyset$. *The Antichain on an attribute A is denoted as* $A^{\leftrightarrow}$.

b) The Dual *preference* $P^{\delta} = (A, <_{P^{\delta}})$ *reverses the order on the preference P.*

c) Given a preference $P = (A, <_P)$ *on an attribute A of relation R. Then the following holds for the* grouping preference *from Eq. 3:* $\sigma[P \text{ grouping } B](R) = \sigma[B^{\leftrightarrow} \,\&\, P](R)$.

2. Skyline Cardinality Estimation

In this section we review the state of the art methods for cardinality estimation of traditional Skyline queries. Estimating the expected Skyline size for a given data distribution is a key issue for the design of good cost models for Skyline queries. The problem has been addressed in several papers; here we report some major results. For this we denote $s_{d,n}$ the *exact* Skyline size for n d-dimensional objects, and $\hat{s}_{d,n}$ denotes the *expected* value of $s_{d,n}$.

▷ **Bentley et al. [13]** were one of the first considering the problem of estimating the size of a Skyline result. They derived a recurrence relation for computing the average number $\hat{s}_{d,n}$ of Skyline points (called *maximal vectors* in [13]) in a set of n vectors in d-space under the assumption that (i) *all attributes are statistically independent* of each others, and (ii) *the probability of sampling the same attribute value twice is negligible*.

Theorem 1 *[13] The Skyline expected value $\hat{s}_{d,n}$ for $d > 1$ and $n > 0$ obeys the following recurrence equation:*

$$\hat{s}_{d,n} = \frac{1}{n}\hat{s}_{d-1,n} + \hat{s}_{d,n-1} \tag{4}$$

For $d \geq 1$ we have $\hat{s}_{d,1} = 1$ and for $n \geq 1, \hat{s}_{1,n} = 1$.

To see why the recurrence in Theorem 1 holds, consider the tuple that has the smallest value with respect to the first preference. For this tuple to be in the Skyline, it must be in the Skyline of the remaining preferences. The probability of this is $\frac{1}{n} \cdot \hat{s}_{d-1,n}$ since there are n total tuples and $\hat{s}_{d-1,n}$ of them are in the Skyline of the remaining preferences. Also, this tuple cannot dominate any other tuple. So, out of the remaining $n-1$ tuples, $\hat{s}_{d,n-1}$ are expected to be in the Skyline. Hence, we get the above recurrence equation.

In addition to the recurrence equation above, Bentley et al. proved that when all d dimensions are *independent*, the expected Skyline cardinality $\hat{s}_{d,n}$ is related to the *harmonic* numbers. The *harmonic* of n for $n > 0$ is $\mathrm{H}_n = \sum_{i=1}^{n} 1/i$, and the k-th order harmonic of n, for integers $d > 0$, is $\mathrm{H}_{d,n} = \sum_{i=1}^{n} \frac{\mathrm{H}_{d-1,i}}{i}$.

Theorem 2 *[13]*

$$\frac{1}{(d-1)!} H_{d-1,n} \;\leq\; \hat{s}_{d,n} \;\leq\; H_{d-1,n} \tag{5}$$

Therefore, $\hat{s}_{d,n} = O((\ln n)^{d-1})$ for fixed d.

This cardinality bound is often cited and employed in Skyline work. However, this is a loose upper-bound [11].

▷ Under the same assumptions and that the components of each vector are distributed independently, **Buchta [14]** provided a closed form and proved the tighter bound $\Theta((\ln n)^{d-1}/(d-1)!$:

Theorem 3 *[14]*

$$\hat{s}_{d,n} = \sum_{k=1}^{n} (-1)^{k+1} \cdot \binom{n}{k} \frac{1}{k^{d-1}} \tag{6}$$

▷ **Godfrey [11]** considers Skyline cardinality for relational processing. Within a basic model with assumptions of *sparseness* (no duplicate values on the attribute across the tuples) and *statistical independence*, he established a model to estimate the Skyline cardinality.

Theorem 4 *[11] The recurrence for $\hat{s}_{d,n}$ in Theorem 1, Eq. (4) is related to the harmonic numbers, that means*

$$\hat{s}_{d+1,n} = H_{d,n} = \sum_{i_1=1}^{n} \sum_{i_2=1}^{i_1} \cdots \sum_{i_d=1}^{i_{d-1}} \frac{1}{i_1 i_2 \cdots i_d} \tag{7}$$

The $H_{d,n}$ are easy to compute and so could be used within a cost model on-the-fly. From this though, Godfrey [11] proved without the restriction of no duplicate values on any dimension that for sufficiently large n the following holds: $\hat{s}_{d,n} = \mathrm{H}_{d-1,n} \approx (\ln n)^{d-1}/(d-1)!$.

▷ **Chaudhuri et al. [15]** proposed to use integrals in the case of independent preferences to estimate the Skyline cardinality, but the integrals have no closed form. They derived a precise estimation when all preferences are numeric. Thereby it is assumed that the attributes take values in the interval $[0,1]$.

Theorem 5 *[15] Let n be the number of d-dimensional objects $(x_1,\ldots,x_d)$. Given d numeric independent preferences, then*

$$\hat{s}_{d,n} = n \cdot \int_{[0,1]^d} (1 - x_1 \cdots x_d)^{n-1} \, \mathrm{d}x_1 \cdots \mathrm{d}x_d \tag{8}$$

Eq. (8) gives an alternative expression for $\hat{s}_{d,n}$ that is equivalent to the solution of the recurrence in Eq. 4 in Theorem 1. The integral has no closed form, but several numerical methods exist to evaluate the integral, cf. [15]. Furthermore, they are the first who derive cardinality estimates for categorical attributes, because Eq. (4) does not hold when the set of preferences also contain predicate preferences.

By empirical studies, [15] observerd that when the data distribution does not have significant correlations/anti-correlations, the Skyline cardinality follows the model: $A(\log n)^B$, where A and B are two constants. Log Sampling (LS) is based on this hypothetical model LS draws a small sample to estimate the two parameters A and B, which are then used to estimate the Skyline cardinality of the whole dataset. More specifically:

Theorem 6 *[15] Let S be a small sample from the dataset drawn from a simple random sampling without replacement, and $S_1 \cup S_2 = S$, where $|S_1| \neq |S_2|$. Let $|Sky_{S_1}|$ and $|Sky_{S_2}|$ the Skyline cardinality of S_1 and S_2, respectively. According to the model $|Sky_{S_1}| = A(\log|S_1|)^B$ and $|Sky_{S_2}| = A(\log|S_2|)^B$ solve A and B as*

$$B = \frac{\log(|Sky_{S_2}|) - \log(|Sky_{S_1}|)}{\log(\log(|S_2|)) - \log(\log(|S_1|))} \quad \text{and} \quad A = \frac{|Sky_{S_1}|}{(\log(|S_1|))^B} \tag{9}$$

Then, $\hat{s}_{d,n} = A(\log n)^B$

▷ **Lu et al. [16]** established specific parametric formulae to estimate the Skyline cardinality over *uniformly and arbitrary distributed data* on *data streams*, keeping the *independence assumption* between dimensions. They propose a robust approach for estimating the Skyline cardinality over sliding windows in the stream environment using Spec-

tral Bloom Filters. The idea of [16] is based on the approaches for Skyline cardinality estimation proposed in Eq. (4), and Eq. (8). Lu et al. notice that these approaches suffer from the strong assumptions of statistical independence across dimensions and no duplicate values over each dimension, and hence do not apply to real-life applications. They relax the restriction on the data distribution. Since skewed data is very common in real-life datasets [16] gives an approach for estimating the Skyline cardinality over *arbitrarily distributed data*, in which probability functions of all dimensions are considered.

Theorem 7 *[16] Suppose that $\bar{x}_1, \ldots, \bar{x}_n$ are n d-dimensional live elements in a sliding window, where $\bar{x}_i = (x_{i1}, \ldots, x_{id})$. The probability function of the data over the j-th dimension is f_j; $\mathbb{P}\{x_{ij} = v_{jt}\} = f_j(t)$, $v_{j1} < v_{j2} < \cdots < v_{jc_j}$, where c_j is the number of distinct elements of the j-th dimension. Under assumptions of statistical independence across dimensions, the expected number of Skyline elements is*

$$\hat{s}_{d,n} = n \cdot \sum_{t_1=1}^{c_1} \cdots \sum_{t_d=1}^{c_d} f_1(t_1) \cdots f_d(t_d) \left(1 - \prod_{j=1}^{d} \sum_{k=1}^{t_j} f_j(k)\right)^{n-1} \tag{10}$$

Estimating the Skyline cardinality using Theorem 7 has a computational complexity of $O\left(\prod_{j=1}^{d} c_j\right)$, where c_j is the value cardinality. If the number of dimensions and the value cardinalities of some dimensions are large, the computational cost overhead is unacceptable. Therefore, [16] introduced a better approximation distinguishing high and low value cardinalities to approximate Eq. (10), which is explained in detail in [16].

▷ **Zhang et al. [17]** propose a kernel-based approach to approximate the Skyline cardinality. They use kernels to estimate the probability density function (PDF) at an arbitrary position in the data space from a random sample. Given a random sample S and an arbitrary position q, the PDF at q can be estimated by the formula

$$PDF(q) = \sum_{s_i \in S} \frac{1}{|S| h^d \sqrt{\det(\sum)}} K\left(\frac{dist_{\sum}(q, s_i)}{h}\right) \tag{11}$$

where h is the kernel bandwidth, $\sum$ (a $d \times d$ matrix) is the kernel orientation, $\det(\sum)$ is the determinant of $\sum$, $dist_{\sum}(q, s_i)$ is the Mahalanobis distance between q and the sample point s_i, and K is the kernel function, which in practice is usually implemented as Gaussian Kernel, as defined in the equation $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.

Theorem 8 *[17] Let S be a sample drawn from the dataset D with $n = |D|$ using a simple random sampling without replacement, and sky_S the local Skyline points of S, $m = |S|$. Then,*

$$\hat{s}_{d,n} = n \cdot \frac{1}{m} \sum_{p \in sky_S} (1 - \Omega_p)^{n-m} \tag{12}$$

where $\Omega_p = \int_{IDR(p)} PDF(q)\, \mathrm{d}q$ is the probability that a random point in the dataset falls in its inverse dominance region (IDR), $IDR(p) = \{q \mid q[i] \leq p[i], \forall\ 1 \leq i \leq d\}$.

This is a robust approach with non-parametric methods and without any assumptions about dataset properties. Since this kernel-based approach uses complex mathematical calculations, it is comparatively complex.

▷ **Jin et al. [18]** present a method to estimate the size of Skylines in join results in order to allow the query optimizer to produce efficient plans for evaluating Skyline queries with joins. They propose a estimation method when the *join size is known*. When the join size is *unknown* [18] suggest to use the End-biased sampling method or a uniform random sample. They remark that applying some classic Skyline estimation methods like in [11] do not produce good results, because previous work on Skyline estimation require that all attributes are independent, and that all values of the same attribute are distinct. However, a joined table does not meet this requirement since one tuple may correspond to several tuples with the same join attribute value.

For their approach two basic assumptions are made: First, every data attribute of relation R or S is *drawn randomly from some probability distribution and is distinct*. Second, all attributes of R and S are *independent*. Note that this is only for the input tables but not for the join table. In their first approach they calculated the probability that an object z is a Skyline tuple using a complex integral, i.e., none of the records in $R \bowtie S$ dominates z. Unfortunately it is difficult to solve the given integration directly. Instead, [18] deduce robust upper and lower bounds, where $p(m,d) \approx \frac{(\log m)^{d-1}}{m(d-1)!}$ is the probability that a random d-dimensional object is a Skyline member among m random records.

Theorem 9 *[18] Given two relation R and S with d_R attributes in R and d_S attributes in S. Let $R = \bigcup_{i=1}^{t} R_i$ and $r_i = |R_i|$. We define S_i similarly. Let $a = \sum_{j=1}^{t} \min\{r_j, s_j\}$. Under the assumption that the join size $n = |R \bowtie S|$ is known, the expected number of Skyline objects $\hat{s}$ in $R \bowtie S$ is bounded by*

$$p(|R|, d_R) \cdot \sum_{i=1}^{t} r_i \cdot s_i \cdot p(s_i, d_S) \;\leq\; \hat{s}_{d,n} \;\leq\; n \cdot p(a-1, d_R + d_S) \tag{13}$$

▷ **Luo et al. [19]** use a non-parametric sampling-based approach to estimate the Skyline cardinality. They do not assume any particular data distribution and their method is more robust than log sampling [15].

Theorem 10 *[19] Let S be a sample of size $m = |S|$ drawn from a simple random sampling without replacement, and Sky_S the global Skyline point set contained in the sample. Then*

$$\hat{s}_{d,n} = \frac{|Sky_S|}{m} \cdot n \tag{14}$$

Following [19] the value $\hat{s}_{d,n}$ is an unbiased estimator of the global Skyline cardinality. The above approach is a non-parametric purely sampling-based method for Skyline cardinality estimation. It does not assume any particular data distribution, and therefore it is more robust than other techniques and applicable to any dataset.

▷ **Tiakas et al. [20]** provide different methodologies for estimating the maximum domination value under the *distinct values* and *attribute independence assumptions*. They use an estimation method with roots and generalized the results to the final proposed estimation formula shown in Theorem 11.

Theorem 11 *[20] Let n be the number of d-dimensional data points. Then,*

$$\hat{s}_{d,n} \approx n - n \cdot \left(1 - \sqrt[d]{\frac{1}{n^{1+\frac{1}{10}\log n + \frac{2}{100}\log^2 n}}}\right)^d \tag{15}$$

The additional terms $\frac{1}{10}\log n$ and $\frac{2}{100}\log^2$ in the exponent of n enhance the estimation accuracy of the Skyline cardinality.

▷ **Shang and Kitsuregawa [21]** established a cardinality model for *anti-correlated distributions* and proposed a polynomial estimation for the expected value of the Skyline cardinality. Their contribution can be summarized as follows:

Theorem 12 *[21] Let $d \geq 2$ and $\Gamma(n) = \int_0^\infty e^{-t} t^n \, \mathrm{d}t$ and $\hat{s}_{d,n}$ the expected value of the Skyline cardinality with n points and d dimensions.*

- *The lower and upper bounds of the Skyline cardinality is*

$$\sum_{k=1}^{d} (-1)^{k-1} \binom{d-1}{k-1} \cdot n \frac{\Gamma\left(\frac{k}{d}\right) \cdot \Gamma(n)}{\Gamma\left(n+\frac{k}{d}\right)} \;\leq\; \hat{s}_{d,n} \;\leq\; n \tag{16}$$

- *The lower bound of the Skyline cardinality can be estimated by the following polynomial equation*

$$\sum_{k=1}^{d} (-1)^{k-1} \binom{d-1}{k-1} \cdot \Gamma\left(\frac{k}{d}\right) \cdot n^{1-\frac{k}{d}} \;\leq\; \hat{s}_{d,n} \;\leq\; n \tag{17}$$

  This estimation can be further abbreviated as

$$O\left(n^{\frac{d-1}{d}}\right) \;\leq\; O\left(\hat{s}_{d,n}\right) \;\leq\; O(n)$$

In [21] it is summerized that the Skyline operator on anti-correlated data is an operator with high output cardinality and this cardinality increases with the dimensionality.

3. Preference Selectivity Estimation

Most of the work on preference selectivity estimation is based on Skyline / Pareto queries having only numerical minimum (LOWEST) or maximum (HIGHEST) preferences as described in Section 2. However, for cost-based query optimization in relational engines not only Pareto must be considered, but all kind of database preferences.

Endres [22] presents an estimation for the *selectivity* of *database preferences* under the *uniformity and attribute independence assumptions*. His estimating and calculating methods rely on a differentiated consideration of the individual preferences modeled as strict partial orders. His approach is rather simple but very effective and usable with any database preference query described in Section 1, whereas Skyline estimations are restricted to minimum / maximum preferences.

In [22] the estimation of the *selectivity* $\mathrm{sel}(P)$ of an arbitrary database preference P on an input dataset R is discussed. The input can be a relational base table of the database or a virtual table which is the intermediate result in a query's evaluation. The *preference selectivity* $\mathrm{sel} = \frac{|out|}{|in|}$ is the ratio of the size of output objects to the size of the current input objects, and leads to a numerical value between 0 and 1. Estimating the preference selectivity is equivalent to estimate the *cardinality* of the output relation of a preference / Skyline operator based upon its input relation R. Formally [22] uses $|\sigma[P](R)|$ for the *cardinality* of the *preference selection* and $\mathrm{sel}(P) \cdot |R|$ for the *estimated size*. Note that $\mathrm{sel}(P) \cdot |R|$ is the same as $\hat{s}_{d,n}$ for Skyline cardinality estimations as in Section 2. Furthermore, let $V(A,R)$ denote the number of distinct values of attribute A in R. $\mathrm{range}(A,R)$ is used to describe the values of attribute A which are available in the table R. The terms $\min_{\mathrm{A,R}}$ and $\max_{\mathrm{A,R}}$ denote the *minimal value* and the *maximal value* of attribute A, i.e., $\min_{\mathrm{A,R}} = \min(\mathrm{range}(A,R))$ and $\max_{\mathrm{A,R}} = \max(\mathrm{range}(A,R))$.

In [22] the following *basic model* of assumptions about the input relation is used: First, it is assumed that the attribute dimensions are *statistically independent*, and second, there is a *uniformly data distribution* for each attribute. These basic assumptions are widely used in selectivity estimation, cp. e.g., [11,23,24], and in Section 2. Note that the assumption of *no duplicate values* as in [11] is not necessary.

3.1. Selectivity of Base Preferences

Endres [22] used the simple method of computing the ratio of possible matches of a preference query to the complete range of database values. The results for base preferences are depicted in Table 2.

For example, the $\mathrm{LOWEST}_d(A)$ and $\mathrm{HIGHEST}_d(A)$ preferences retrieve all extremal values from the attribute A. If $d = 0$ and under the assumption of uniformly distributed data and that there are $V(A,R)$ distinct attribute values, only one of these distinct values can be the minimal or maximal numerical value. This is also similar to a hard selection $\sigma_{A=\min_{\mathrm{A,R}}}(R)$ or $\sigma_{A=\max_{\mathrm{A,R}}}(R)$ which has a selectivity of $1/V(A,R)$, cp. [23]. In the presence of the d parameter [22] computes the ratio of d to the complete range of database values, because the best objects may lie inside a distance d from the minimal or maximal value, i.e., $\mathrm{sel}(P) = d/(\max_{\mathrm{A,R}} - \min_{\mathrm{A,R}})$.

Endres [22] presents also a technique to estimate predicate preferences as in [15]. In contrast to [15] which only relies on 1 (predicated is satisfied) and 0 (predicate is not satisfied), [22] presents a more general method, cp. Table 2, using $\mathbf{LAYERED}_m(A, \{L_1, \ldots, L_m\})$. Since all categorical preferences are sub-constructors of LAYERED, it is sufficient to show the selectivity of $\mathrm{LAYERED}_m$, cp. Section 1. The set $\{L_1, \ldots, L_m\}$ partitions the domain of A, in which each L_i represents a set of objects. This is similar to an `IN` query in standard SQL, cp. [23]. Therefore the selectivity can be estimated by $\mathrm{sel}(P) = |L_i|/V(A,R)$. However, the set L_1 may contain objects which are not available in the database, hence L_2 must be considered. On the other hand, L_1

could contain objects which are available in the database, and some not. Therefore the L_i sets are consecutively checked to determine the selectivity of LAYERED_m. Note that for this kind of selectivity estimation a strong statistical profile for the relation R must be available, but this is a closed formula in contrast to [15]. For more details and the proofs we refer to [22].

Table 2. Base preference selectivity under the assumptions of uniformly data and attribute independence.

P	**sel(P), if d=0**	**sel(P), if d > 0**	**Condition**
SCORE_d	1	1	-
$\text{LOWEST}_d(A)$	$\frac{1}{V(A,R)}$	$\frac{d}{\max_{A,R}-\min_{A,R}}$	-
$\text{HIGHEST}_d(A)$	$\frac{1}{V(A,R)}$	$\frac{d}{\max_{A,R}-\min_{A,R}}$	-
$\text{ATLEAST}_d(A,z)$	1	1	$z \leq \min_{A,R}$
	$\frac{1}{V(A,R)}$	$\frac{\lceil\frac{z-\max_{A,R}}{d}\rceil\cdot d-(z-\max_{A,R})}{\max_{A,R}-\min_{A,R}}$	$z \geq \max_{A,R}$
	$\frac{\max_{A,R}-z}{\max_{A,R}-\min_{A,R}}$	$\frac{\max_{A,R}-z}{\max_{A,R}-\min_{A,R}}$	$z \in \text{range}(A,R)$
$\text{ATMOST}_d(A,z)$	$\frac{1}{V(A,R)}$	$\frac{\lceil\frac{\min_{A,R}-z}{d}\rceil\cdot d-(\min_{A,R}-z)}{\max_{A,R}-\min_{A,R}}$	$z \leq \min_{A,R}$
	1	1	$z \geq \max_{A,R}$
	$\frac{z-\min_{A,R}}{\max_{A,R}-\min_{A,R}}$	$\frac{z-\min_{A,R}}{\max_{A,R}-\min_{A,R}}$	$z \in \text{range}(A,R)$
$\text{AROUND}_d(A,z)$	$\frac{1}{V(A,R)}$	$\frac{\lceil\frac{\min_{A,R}-z}{d}\rceil\cdot d-(\min_{A,R}-z)}{\max_{A,R}-\min_{A,R}}$	$z < \min_{A,R}$
	$\frac{1}{V(A,R)}$	$\frac{\lceil\frac{z-\max_{A,R}}{d}\rceil\cdot d-(z-\max_{A,R})}{\max_{A,R}-\min_{A,R}}$	$z > \max_{A,R}$
	$\frac{1}{V(A,R)}$	$\frac{2d}{\max_{A,R}-\min_{A,R}}$	$z-d \geq \min_{A,R} \wedge z+d \leq \max_{A,R}$
	-	$\frac{d+(z-\min_{A,R})}{\max_{A,R}-\min_{A,R}}$	$z-d < \min_{A,R} \wedge z \in \text{range}(A,R)$
	-	$\frac{d+(\max_{A,R}-z)}{\max_{A,R}-\min_{A,R}}$	$z+d > \max_{A,R} \wedge z \in \text{range}(A,R)$
	-	1	$z-d < \min_{A,R} \wedge z+d > \max_{A,R}$
$\text{BETWEEN}_d(A,low,up)$	1	1	$low \leq \min_{A,R} \wedge up \geq \max_{A,R}$
	$\frac{1}{V(A,R)}$	$\frac{\lceil\frac{\min_{A,R}-up}{d}\rceil\cdot d-(\min_{A,R}-up)}{\max_{A,R}-\min_{A,R}}$	$up < \min_{A,R}$
	$\frac{1}{V(A,R)}$	$\frac{\lceil\frac{low-\max_{A,R}}{d}\rceil\cdot d-(low-\max_{A,R})}{\max_{A,R}-\min_{A,R}}$	$low > \max_{A,R}$
	$\frac{up-\min_{A,R}}{\max_{A,R}-\min_{A,R}}$	$\frac{up-\min_{A,R}}{\max_{A,R}-\min_{A,R}}$	$low < \min_{A,R} \wedge up \in \text{range}(A,R)$
	$\frac{\max_{A,R}-low}{\max_{A,R}-\min_{A,R}}$	$\frac{\max_{A,R}-low}{\max_{A,R}-\min_{A,R}}$	$low \in \text{range}(A,R) \wedge up > \max_{A,R}$
	$\frac{up-low}{\max_{A,R}-\min_{A,R}}$	$\frac{up-low}{\max_{A,R}-\min_{A,R}}$	$low,up \in \text{range}(A,R),\ low < up$
$\text{LAYERED}_m(A,\{L_1,\ldots,L_m\})$	$\frac{1}{V(A,R)}\cdot l(1)$	$l(i)=\begin{cases}\lvert\{t\in L_i : t\in\text{range}(A,R)\}\rvert & \text{if } \exists t\in L_i : t\in\text{range}(A,R)\\ l(i+1) & \text{otherwise}\end{cases}$	

3.2. Selectivity of Complex Preferences

Following [22], Table 3 reports selectivity estimates for complex preferences.

Table 3. Selectivity estimation for complex preferences.

P	$A^{\leftrightarrow}$	P^{δ}	$\&_{i=1}^{m} P_i$	$A^{\leftrightarrow} \,\&\, P$	$\bigotimes_{i=1}^{m} P_i$
sel(P)	1	$1 - \mathrm{sel}(P)$	$\prod_{i=1}^{m} \mathrm{sel}(P_i, R)$	$\mathrm{sel}(P)$	$\sum_{i=1}^{m} \mathrm{sel}(P_i, R)$

$\triangleright$ $\mathbf{A^{\leftrightarrow}}$: In the antichain preference $A^{\leftrightarrow}$ all values in the attribute A are incomparable. Therefore the selectivity is $\mathrm{sel}(P) = 1$.

$\triangleright$ $\mathbf{P^{\delta}}$: The dual preference reverses the order on P, hence $\mathrm{sel}(P^{\delta}) = 1 - \mathrm{sel}(P)$.

$\triangleright$ $\mathbf{\&_{i=1}^{m} P_i}$: The prioritization preference is similar to the `AND` predicate in standard SQL [23]. Each preference which takes part in the prioritization reduces the cardinality of the result, therefore [22] multiplies the selectivity of the child preferences: $\mathrm{sel}(\&_{i=1}^{m} P_i) = \prod_{i=1}^{m} \mathrm{sel}(P_i)$. Note that this assumes that column values are independent.

$\triangleright$ $\mathbf{A^{\leftrightarrow} \,\&\, P}$: A special prioritization preference is the grouped preference selection $\sigma[P \text{ grouping } A](R)$. This leads to $\mathrm{sel}(A^{\leftrightarrow} \,\&\, P) = \mathrm{sel}(A^{\leftrightarrow}) \cdot \mathrm{sel}(P) = \mathrm{sel}(P)$.

$\triangleright$ $\mathbf{\bigotimes_{i=1}^{m} P_i}$: Endres [22] relies on a rather simplistic approach to estimate the selectivity of a Pareto preference. The selectivity is computed recursively for each child preference and all selectivities are summed up: $\mathrm{sel}(\bigotimes_{i=1}^{m} P_i) = \sum_{i=1}^{m} \mathrm{sel}(P_i)$. Of course, the sum is restricted to be less or equal to 1.

Note that this estimation is less accurate than the approaches in Section 2, but on the one hand this allows the usage of arbitrary numerical *and* categorical base preferences in a Pareto query and on the other hand this *also allows complex* preferences in a Pareto constructor.

4. Conclusion

In this paper we considered the problem of estimating the cardinality of a Skyline or preference query. Estimating the cardinality or the selectivity is crucial for cost-based query optimization, since the Skyline operator does not stand alone in a preference query. We gave an overview on existing techniques and discussed their advantages and disadvantages. We presented robust and sampling techniques for Skyline estimation as well as simple and closed forms for arbitrary preference database queries. From this knowledge one can decide which technique should be used in a database engine.

References

[1] Stefanidis K., Koutrika G., Pitoura E. A Survey on Representation, Composition and Application of Preferences in Database Systems. *ACM Transaction on Database Systems*, 2011, 36(3), 19:1–19:45.

[2] Kießling W. Foundations of Preferences in Database Systems. *Proceedings of the 28th International Conference on Very Large Data Bases [VLDB].* Hong Kong, China. VLDB, 2002. 311–322.

[3] Chomicki J. Preference Formulas in Relational Queries. *TODS*, 2003, 28(4), 427–466.

[4] Börzsönyi S., Kossmann D., Stocker K. In: Georgakopoulos D., Buchmann, A. (eds.) *Proceedings of the 17th International Conference on Data Engineering [ICDE].* Washington, DC, USA. IEEE Computer Society, 2001. 421–430.

[5] Chomicki J., Ciaccia P., Meneghetti N. Skyline Queries, Front and Back. *ACM SIGMOD Record*, 2013, 42(3), 6–18.

[6] Kießling W., Endres M., Wenzel F. The Preference SQL System – An Overview. *Bulletin of the Technical Commitee on Data Engineering*, 2011, 34(2), 11–18.

[7] Arvanitis A., Koutrika G. Towards Preference-aware Relational Databases. In: Kementsietsidis A., Salles M. A. V. (eds.) *Proceedings of the 28th International Conference on Data Engineering [ICDE].* Washington, DC, USA. IEEE Computer Society, 2012. 426–437.

[8] Tan K.-L., Eng P.-K., Ooi B. C. Efficient Progressive Skyline Computation. In: Apers P., Atzeni P., Ceri S., Paraboschi S., Ramamohanarao K., Snodgrass R. T. (eds.) *Proceedings of the 27th International Conference on Very Large Data Bases [VLDB].* San Francisco, CA, USA. VLDB, 2001. 301–310.

[9] Chomicki J., Godfrey P., Gryz J., Liang D. Skyline with Presorting. In: Dayal U., Ramamritham K., Vijayarama T. M. (eds.) *Proceedings of the 19th International Conference on Data Engineering [ICDE].* March 5-8, Bangalore, India. IEEE Computer Society, 2013. 717–719.

[10] Hafenrichter B., Kießling W. Optimization of Relational Preference Queries. In: Williams H. E., Dobbie, G. (eds.) *Proceedings of the 16th Australasian Database Conference [ADC].* Newcastle, Australia. Australian Computer Society, 2005. 175–184.

[11] Godfrey P. Skyline Cardinality for Relational Processing. In: Seipel D., Torres J. M. T. (eds.) *Foundations of Information and Knowledge Systems [FoIKS].* February 17-20, Wilhelminenberg Castle, Austria. Springer, LNCS, 2004. 78–97.

[12] Kießling W. Preference Queries with SV-Semantics. In: Haritsa J. R., Vijayaraman T. M. (eds.) *Proceedings of the 11th International Conference on Management of Data [COMAD].* Goa, India. Computer Society of India, 2005. 15–26.

[13] Bentley J. L., Kung H. T., Schkolnick M., Thompson C. D. On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of the ACM*, 1978, 25(4), 536–543.

[14] Buchta, C. On the Average Number of Maxima in a Set of Vectors. *Inf. Process. Lett.*, 1989, 33(2).

[15] Chaudhuri S., Dalvi N., Kaushik R. Robust Cardinality and Cost Estimation for Skyline Operator. In: Liu L., Reuter A., Whang K.-Y., Zhang J. (eds.) *Proceedings of the 22nd International Conference on Data Engineering [ICDE].* Atlanta, GA, USA. IEEE Computer Society, 2006. 64.

[16] Lu Y., Zhao J., Chen L., Cui B., Yang D. Effective Skyline Cardinality Estimation on Data Streams. In: Bhowmick S. S., Küng J., Wagner R. (eds.) *Proceedings of the 19th International Conference on Database and Expert Systems Applications [DEXA].* Turin, Italy. Springer, 2008. 241–254.

[17] Zhang Z., Yang Y., Cai R., Papadias D., Tung A. Kernel-Based Skyline Cardinality Estimation. In: Cetintemel U., Zdonik S. B., Kossmann D., Tatbul N. (eds.) *Proceedings of the 35th SIGMOD International Conference on Management of Data [SIGMOD].* New York, NY, USA. ACM, 2009. 509–522.

[18] Jin W., Morse M. D., Patel J. M., Ester M., Hu Z. Evaluating Skylines in the Presence of Equijoins. *Proceedings of the 2010 IEEE International Conference on Data Engineering [ICDE].* Long Beach, CA, USA. IEEE, 2010. 249–260.

[19] Luo C., Jiang Z., Hou W.-C., He S., Zhu Q. A Sampling Approach for Skyline Query Cardinality Estimation. *Knowl. Inf. Syst.*, 2012, 32(2), 281–301.

[20] Tiakas E., Papadopoulos A. N., Manolopoulos Y. On Estimating the Maximum Domination Value and the Skyline Cardinality of Multidimensional Data Sets. *International Journal of Knowledge-Based Organizations*, 2013, 3(4).

[21] Shang H., Kitsuregawa M. Skyline Operator on Anti-correlated Distributions. *PVLDB*, 2013, 6(9).

[22] Endres M. Preference Selectivity Estimation for Cost-Based Query Optimization. *Proceedings of the 11th International Baltic Conference on DB and IS [DB&IS].* Tallin, Estonia. 2014

[23] Selinger P. G., Astrahan M. M., Chamberlin D. D., Lorie R. A., Price T. G. Access Path Selection in a Relational Database Management System. *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data [SIGMOD].* New York, NY, USA. ACM, 1979. 23–34.

[24] Piatetsky-Shapiro G., Connell C. Accurate Estimation of the Number of Tuples Satisfying a Condition. In: Yormark B. (ed.) *Proceedings of the International Conference on Management of Data [SIGMOD].* New York, NY, USA. ACM, 1984. 256–276.