prefSQL fact sheet

The prefSQL framework allows preference-based database queries against relational databases. It is an extension of the Structured Query Language (SQL). In its natural form SQL allows only hard constraints with help of the WHERE-Clause.

Skyline Example

The prefSQL framework allows calculating the Pareto optimal result according to the preferences. Example 1 shows a preference for cars that are inexpensive and have low mileage with help of the “SKYLINE OF” clause.

```sql
SELECT t1.id, t1.title, t1.price, t1.mileage
FROM cars_small t1
SKYLINE OF t1.price LOW, t1.mileage LOW
```

Skyline Features and Syntax

Table 1.1 shows all available preference operators. For better understanding, an example for each preference is given.

Preference	Description	Condition	Example
Around	Closer to x is better	Numeric	t.price AROUND 10000
Low/High	Lower is better Higher is better	Numeric Numeric	t.price LOW t.enginesize HIGH
Low/High Step Equal	Lower is better, Step function x (Levelling) Values on same level indifferent	Numeric	t.price LOW 1000 EQUAL
Low/High Step Incomparable	Low is better Step function x (Levelling) Values on same level incomparable	Numeric	t.price LOW 1000 INCOMPARABLE
Favour **Disfavour**	Like a value more than others Dislike a value more than others	Text	t.color FAVOUR ‘red’ t.color DISFAVOUR ‘red’
Categorical	Define an order for an attribute	Text	t.color (‘red’ >> ‘blue’ == ‘yellow’)
Categorical Multi-value	Define incomparable values inside an order	Text	t.color ({‘red’, ‘blue’} >> ‘yellow’)
Categorical Equal	Define an order for an attribute Other values are indifferent	Text	t.color (‘red’ >> OTHERS EQUAL >> ‘blue’)
Categorical Incomparable	Define an order for an attribute Other values are incomparable	Text	t.color (‘red’ >> ‘blue’ >> OTHERS INCOMPARABLE)

Table 1.1: Complete list of prefSQL operators for preferences

As shown in Table 1.1, only numeric and text attributes are supported. Other data type, like date, must be converted first. The following skyline algorithms are available:

Algorithm	Supports Incomparable	Supports Implicit OTHERS	MS SQL CLR
BNL	X	X	X
BNLSort	X	X	X
DQ			X
Hexagon	X[1]		X
SQL	X	X	X
MultipleBNL[2]	X	X	X

Table 1.2: Complete list of prefSQL skyline algorithms

[1] It is not recommended to use it. The tree width increases with the amount of unique incomparable values

[2] As the names indicate the multiple skyline algorithm calculates multiple skylines. It is based on the BNLSort algorithm.

The framework supports the usual statements for the order by clause. Additionally it knows the special sort orders in Table 1.3

Keyword	Description
ORDER BY BEST_RANK()	Sorted according to the best preference rank
ORDER BY SUM_RANK()	Sorted according to sum of all the preference ranks

Table 1.3: Complete list of additional prefSQL sort operators

Weighted Sum Example

The prefSQL framework allows giving importance to preferences. Example 2 shows that the price is four times more important than low mileage with help of the "RANKING OF" clause. The result of a weighted sum is the full dataset sorted according to the preferences.

```
SELECT t1.id, t1.title, t1.price, t1.mileage
FROM cars_small t1
RANKING OF t1.price LOW 0.8, t1.mileage LOW 0.2
```

Weighted Sum Features and Syntax

Weighted Sum does not support incomparable tuples and needs an explicit OTHERS EQUAL keyword for categorical preferences. The weight of the preference must be defined after the preference, for example, t1.price LOW 0.8.

Preference	Description	Condition	Example
Around	Closer to x is better	Numeric	t.price AROUND 10000 0.5
Low/High	Lower is better	Numeric	t.price LOW 0.5
	Higher is better	Numeric	t.enginesize HIGH 0.5
Low/High Step Equal	Lower is better, Step function x (Levelling) Values on same level indifferent	Numeric	t.price 1000 LOW EQUAL 0.5
Categorical Equal	Define an order for an attribute Other values are indifferent	Text	t.color ('red' >> OTHERS EQUAL >> 'blue') 0.5

Table 1.4: Complete list of prefSQL operators for preferences