SetQL Constructs

Muttaqi Islam

July 2023

1 Introduction

This white paper describes mathematical constructs that allow us to model modern databases in a formal set-oriented manner without losing any major features. We model column types (eg. int, double, strings) as sets (eg. $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{S}$ respectively). We define $\mathbb{T}$ as the universe of all types (ie. $\mathbb{Z} \cup \mathbb{R} \cup \mathbb{S} \cup ...$). We model tables as a set of objects τ which belong to a universe of objects $\mathbb{O}$ such that $\tau \subseteq \mathbb{O}$. We model columns of a table τ as functions $f : \tau \mapsto \mathbb{T}_f$ where $\mathbb{T}_f \subseteq \mathbb{T}$ is its range (eg. $\mathbb{R}$); this corresponds to its type. Further, we establish that $\forall x, y \in \tau : x = y \iff \forall f \in \mathbb{T}^\tau : f(x) = f(y)$. We will generally use S^T where S and T are sets to denote the set of functions that have a domain that is a super set of T and a range that is a super set S. This follows the convention described by the University of Arizona *here*.

A table where rows represent objects in τ and columns represent functions on those objects is called a tabular view of τ.

2 Object Identity Function

The object identity function $\eta_T : \tau \mapsto \mathbb{Z}^+$ uniquely identifies all members of any set T with values 0 to $|T|$. Any tabular view of T is ordered by η_T so that it is deterministic, but there is no guarantee that η_T is ordered in any meaningful way such as insert time or last update time.

3 View Function

The view function ν allows us to create a new object belonging to a set similar to τ, $\tau*$, with only a subset of functions defined for it.

$$\nu(x, f_1, ...f_n) : \tau \times (\mathbb{T}^\tau)^n \mapsto \tau*$$

$$\forall i \in \mathbb{Z}, 1 \leq i \leq n : f_i(x) = f_i(\nu(x, f_1, ...f_n))$$

This is useful for defining lighter weight objects that are similar to a heavier object, just like select statements that specify a subset of columns in SQL.

4 Object Minimum Function

The object minimum function μ lets us select an object that has the minimum value for a specific function.

$$\mu(T, f) : P(\tau) \times \mathbb{T}^\tau \mapsto \tau$$

$$\mu(T, f) = x \Rightarrow x \in T \wedge \forall y \in T : f(y) \geq f(x) \wedge f(y) = f(x) \Rightarrow \eta_\tau(x) < \eta_\tau(y)$$

The final clause ensures that this function is defined and deterministic. This is useful for a number of more sophisticated constructs.

5 Set Successor Function

The set successor function σ lets us get a set of objects whose value for a specific function is subsequent to the minimum value of the function for the given values, with respect to the table as a whole.

$$\sigma(T, f) : P(\tau) \times \mathbb{T}^\tau \mapsto P(\tau)$$

$$\sigma(T, f) = \{x \in \tau | f(x) > \mu(f, T), \forall y \in \tau : f(y) > f(x) \vee f(y) < f(\mu(f, T))\}$$

We can also define repeated applications of the set successor using a superscript.

$$\sigma^i(T, f) = \sigma(\sigma^{i-1}(T, f))$$

The base case of this recursion will be a function that gives us all members of τ whose value of f is equal to the minimum value of f in T.

$$\sigma^0(T, f) = \{x \in \tau | f(\mu(f, T)) = f(x)\}$$

6 Set Tier Function

Using repeated set index functions, we can define a tier function ρ. These tiers contain members of T corresponding to a particular value of f. The first tier corresponds to members of τ that have the minimum value of f, and subsequent tiers correspond to strictly increasing values of f.

$$\rho(T, f, i) : \mathbb{T}^\tau \times \mathbb{Z}^+ \mapsto P(\tau)$$

$$\rho(T, f, i) = \sigma^i(f, T)$$

7 Index Function

The index function ι lets us get the index of the tier that an object belongs to.

$$\iota(T, f, x) : \mathbb{T}^{\tau} \times \tau \mapsto \mathbb{Z}^{+}$$

$$\forall x \in \tau : x \in \rho(T, f, \iota(f, x))$$

In modern databases, an index is a lookup table from a field to its corresponding row, sorted by the value of the field. We will define the objective index function I to denote this. It returns a set of tuples of values, with the first being its index and the second being the object.

$$I(T, f) : P(\tau) \times \mathbb{T}^{T} \mapsto P(\mathbb{Z}^{+} \times \tau)$$

$$I(T, f) = \{(\iota(T, f, x), x) | x \in T\}$$

Any tabular view of a set formed by the set index function is ordered by the first value in the tuple. That is to say the set identity function of any set formed by the set index function partially depends on the value of the specified function.

$$\forall (i, x), (j, y) \in I(T, f, \tau) : i > j \Rightarrow \eta_{I(T,f,\tau)}((i, x)) > \eta_{I(T,f,\tau)}((j, y))$$

8 Group Function

The group function γ lets us group multiple objects into one based on a field f and use aggregator functions a_{g_i} to define other fields g_i. These objects belong to a set similar to T, $T*$.

$$\gamma(T, f, a_{g_1}, ...a_{g_m}) : P(\tau) \times \mathbb{T}^{\tau} \times (P(\tau) \mapsto \tau)^{m} \mapsto P(\tau*)$$

$$\gamma(T, f, a_{g_1}, ...a_{g_m}) = \{x_i \in T* | \exists i \in \mathbb{Z}^{+} : f(x_i) = f(\mu(f, \rho(T, f, i))), g_1(x_i) = a_{g_1}(\rho(T, g_1, i)), ...g_m(x_i) = a_{g_m}(\rho(T, g_m, i))\}$$

9 Join Function

The join function χ lets us group multiple objects in different tables $\tau_1 and \tau_2$ based on fields f and g, and consolidate them into a single object belonging to a new set $\tau*$. Again, we can use aggregator functions a_{h_i} to define other fields h_i.

$$\chi(T, S, f, g, a_{h_1}, ...a_{h_m}, ...a_{h_{m+l}}) : \mathbb{T}^{\tau_1} \times \mathbb{T}^{\tau_2} \times (P(\tau_1) \mapsto \tau_1)^{m} \times (P(\tau_2) \mapsto \tau_2)^{l} \mapsto P(\tau_1*)$$

$$\chi(T, S, f, g, a_{h_1}, ..a_{h_m}, ...a_{h_{m+l}}) =$$

$$\{x_i | \exists i \in \mathbb{Z}^{+} : f(x_i) = f(\mu(f, \rho(T, f, i))), g(x_i) = g(\mu(g, \rho(S, g, i))), h_1(x_i) = a_{h_1}(\rho(T, h_1, i)), ...h_{m+l}(x_i) = a_{h_{m+l}}(\rho(S, h_{m+l}, i))\}$$

10 Ordering

Since I is implicitly ordered in SetQL, using it for any set S should order it by the specified field, for example $I(f, S)$. When an index set is converted to a table, there should only be one column for f, not two corresponding to the tuple and the object itself.

11 Limits

Limits can be applied by defining a set as being a subset of another, and specifying the size of a set, for example $S \subseteq \tau, |S| = 5$. Note that mathematically there are many different possible values of S, but in a SetQL system this should be deterministic because τ is sorted by η_τ.

12 Offset

The offset function ω lets us ignore the first $n \in \mathbb{Z}^+$ rows of T by filtering objects by η_T.

$$\omega(T, n) : P(\tau) \times \mathbb{Z}^+ \mapsto P(\tau)$$

$$\omega(T, n) = \{x | \forall (i, x) \in I(T, \eta_T) : i > n\}$$

13 Distinct

The distinct function δ uses ν with all functions defined for τ except for η_T. For objects whose values for all functions are the same, they will not be duplicated in the set since they no longer have η_T values to differentiate between them.

$$\delta(T) : P(\tau) \mapsto P(\tau*)$$

$$\delta(T) = \{\nu(x, f_1, ...f_n) | x \in T, \forall f \in \mathbb{T}^T \setminus \{\eta_T\} : \exists 1 \leq i \leq n : f = f_i\}$$

14 Defining Sets

Defining a new set τ involves defining it as an empty set and defining a number of functions on it.

$$\tau = \{\}, f : \tau \mapsto \mathbb{Z}$$

After any SetQL statement is executed, the system will implicitly perform the substitution $\tau \leftarrow \tau'$ if τ' is defined. Thus, deleting a set involves simply defining τ as an empty set.

$$\tau' = \{\}$$

15 Updates

Updates can be expressed by defining τ' and using the objective update function υ as well as the set update function Υ.

$$\upsilon(x, f, v) : \tau \times \mathbb{T}^\tau \times \mathbb{T}_f \mapsto P(\tau*)$$

$$\upsilon(x, f, v) = y \Rightarrow f(y) = v \wedge \forall g \in \mathbb{T}^\tau \setminus \{f\} : g(x) = g(y)$$

$$\Upsilon(T, f, v) : P(\tau) \times \mathbb{T}^\tau \times \mathbb{T}_f \mapsto P(\tau*)$$

$$\Upsilon(T, f, v) = \{\upsilon(x, f, v) | x \in T\}$$

We can update only a subset of objects by removing it and adding the updated set to τ'.

$$S = \{x \in \tau | g(x) = w\}, \tau' = \tau \setminus S \cup \Upsilon(S, f, v)$$

16 Inserts

We can perform inserts by defining τ' and using set unions.

$$\tau' = \tau \cup \{x | f(x) = v, ...\}$$

We establish that all user-defined sets must be finite, so in this case all functions of x that have infinite domain must be defined otherwise there will be an infinite number of objects added to the set.

17 Deletes

We can performing deletes by defining τ' and using set differences.

$$\tau' = \tau \setminus \{x \in \tau | f(x) = v\}$$

18 Defining Functions

A function f is defined by describing its range, domain, and, if its range has objects in it, a default value for all objects.

$$f : \tau \mapsto \mathbb{Z}, f(x) = 2$$

19 Modifying functions

Similar to tables, for any function f a SetQL system will implicitly perform the substitution $f \leftarrow f'$. Thus, re-defining the function can be used to make column-level updates or change its range.

$$f' : \tau \mapsto \mathbb{R}, \forall x \in \tau : f'(x) = \frac{f(x)}{2}$$

Setting either the domain or range of f to $\{\}$ is equivalent to deleting it, since it means there is no meaningful value in storing the function any more.

$$f' : \{\} \mapsto \{\}$$

20 SetQL Reserved Keywords

Symbol	SetQL
$\mathbb{Z}$	`/Z`
$\mathbb{R}$	`/R`
$\mathbb{S}$	`/S`
ν	`/v`
μ	`/m`
σ	`/s`
ρ	`/t`
ι	`/i`
I	`/I`
γ	`/g`
χ	`/j`
ω	`/o`
δ	`/d`
Υ	`/U`
$\cup$	`/u`
$\in$	`/e`
$\mapsto$	`->`
$\forall$	`/A`
$\subseteq$	`/c`