1 Syntax of simpSQL

The following is the formal definition of the simpSQL language based on Kartik's document, representing a simple programing language with realistic standard SQL queries.

Figure 1: Syntax of simpSQL

2 Syntax of kvSQL

Figure 2 presents the kvSQL language which is used to write generic key-value backed applications. The language is not very different from SQL; it simply replaces tables with (denormalized) objects supporting restricted queries. We will later formally define the translation from simpSQL to kvSQL[1].

$$t \in \texttt{TableName} \qquad f_{id}, f_v \in \texttt{FieldName} \qquad v \in \texttt{Value}$$
$$x \in \texttt{Variable} \qquad txn \in \texttt{TxnName}$$
$$\odot \in \{<, \leq, =, \neq, >, \geq\} \qquad \oplus \in \{\cap, \cup\} \qquad \otimes \in \{\wedge, \vee\}$$

$$\begin{array}{lcl}
pk & ::= & (\overline{f_{id}}, f_v) \\
obj & ::= & (t, pk, \overline{f_v}) \\
r_{obj} & ::= & \bar{v} \\
\phi_{pk} & ::= & pk_{id} \odot v \mid pk_v \odot v \mid \phi_{pk} \otimes \phi_{pk} \\
e & ::= & x \mid \texttt{CHOOSE}\ x \mid r_{obj} \mid e \oplus e \\
\phi_c & ::= & r^i_{obj} \odot v \mid r\ \texttt{IN}\ e \mid \phi_c \otimes \phi_c \\
op & ::= & obj.put(r) \quad \mid \quad x \leftarrow obj.get(\phi_{obj}) \\
c & ::= & \{\overline{op}\}_{DC} \mid x \leftarrow e \mid \\
 & & \texttt{IF}\ \phi_c\ \texttt{THEN}\ c\ \texttt{ELSE}\ c \mid c; c \mid \{c\}_{SER} \mid \\
 & & \texttt{FOREACH}\ r\ \texttt{IN}\ x\ \texttt{DO}\ c\ \texttt{END}
\end{array}$$

Figure 2: Syntax of kvSQL

[1] proving this translation correct is NOT going to be very challenging, since we will initially translate the simpSQL program to a kvSQL version *with SER transactions everywhere* and the difference will only be in the data models

3 Definition of the denormalizer

3.1 Data Modeling Rules

3.2 Program rewriting rules

4 Example: TPC-C in simpSQL and kvSQL

SimpSQL Table: Warehouse

w_id	w_name	w_address	w_tax	w_ytd

kvSQL Object(s): Warehouse

id := (w_id)
warehouse_by_id := (Warehouse,(id,_),[w_name;w_address;w_tax;w_ytd])

SimpSQL Table: District

d_id	d_w_id	d_info	d_ytd	d_tax	d_next_o_id

kvSQL Object(s): District

id := (d_id,d_w_id)
d_info_by_id := (District,(id,_),[d_info])
d_ytd_by_id := (District,(id,_),[d_ytd])
d_tax_by_id := (District,(id,_),[d_tax])
d_next_o_id_by_id := (District,(id,_),[d_next_o_id])

SimpSQL Table: Customer

c_id	c_d_id	c_w_id	c_name	c_ytd	c_delivery_cnt	c_payment_cnt	c_balance

kvSQL Object(s): Customer

id := (c_id,c_d_id,c_w_id)
c_name+ytd+..._by_id := (Customer,(id,_),[c_name;c_ytd;...])
c_balance_by_id := (Customer,(id,_),[c_balance])
c_ytd+..._by_name := (Customer,(id,c_name),[c_ytd;...])
c_balance_by_name := (Customer,(id,c_name),[c_balance])

SimpSQL Table: Orders

o_id	o_d_id	o_w_id	o_c_id	o_carrier_id	o_entry_d

kvSQL Object(s): Orders

id := (o_id,o_d_id,o_w_id)
order_by_id := (Orders,(id,_),[o_c_id;o_carrier_id;o_entry_d])
o_id+entryD+CarriedID_by_o_c_id := (Orders,(id,o_c_id),[o_id;...])

SimpSQL Table: Item

i_id	i_info

kvSQL Object(s): Item

id := (i_id)
i_info_by_id := (Item,(id,_),[i_info])

SimpSQL Table: OrderLine

ol_o_id	ol_d_id	ol_w_id	ol_number	ol_info

kvSQL Object(s): OrderLine

id := (ol_o_id,ol_d_id,ol_w_id,ol_number)
ol_info_by_id := (OrderLine,(id,_),[ol_info])
ol_number+info_by_ol_o_id := (OrderLine,(id,ol_o_id),[ol_number;ol_info])

SimpSQL Table: Stock

s_i_id	s_w_id	s_quant	s_order_cnt	s_info

kvSQL Object(s): Stock

id := (s_i_id,s_w_id)
s_quant_by_id := (Stock,(id,_),[s_quant])
s_orderCnt_by_id := (Stock,(id,_),[s_order_cnt])
s_info_by_id := (Stock,(id,_),[s_info])

SimpSQL Table: OrderLine JOIN Stock

ol_o_id	ol_d_id	ol_w_id	ol_number	ol_info	s_i_id	s_w_id	s_quant

kvSQL Object(s): OrderLine JOIN Stock

id := (ol_o_id,ol_d_id,ol_w_id,ol_number)
s_quant_by_ol_o_id := (OrderLine $\bowtie$ Stock,(id,ol_o_id),[s_quant])

SimpSQL Table: NewOrder

ol_o_id	ol_d_id	ol_w_id

kvSQL Object(s): NewOrder

id := (no_o_id,no_d_id,no_w_id)
?_by_no_d_id := (NewOrder,(id,no_d_id),[])

SimpSQL Table: History

h_id	h_info

kvSQL Object(s): History

id := (h_id)
h_info_by_id := (Item,(id,_),[h_info])

---

simpSQL TPC-C

kvSQL TPC-C

4.0.1 kv transactions

```
naddew order
NEW ORDER!
```

Listing 1: NewOrder Transaction

```
payment
payment
payment
payment
payment
```

Listing 2: Payment Transaction

```
# Group columns are sometimes replaced with "info"
# for better readability
OrderStatus(cust_id,cust_name,) := {
  IF (cust_id ≠ NULL)
  THEN cx← (c_info_by_id).get (id=cust_id)
  ELSE cx1← (c_info_by_name).get (c_name=cust_name);
```

```
7          cx = CHOOSE cx1 # pick the middle customer;
8    ox1← (o_info_by_o_c_id).get(o_c_id=cx.id);
9    ox =  CHOOSE ox1; # pick the largest order id
10   olx← (ol_info_by_ol_o_id).get(ol_o_id=ox.o_id);
11   print olx
12 }SER
```

Listing 3: OrderStatus Transaction

```
1 # Group columns are sometimes replaced with "info"
2 # for better readability
3 StockLevel(dist_id) := {
4   dnox← (d_next_o_id_by_id).get (id=dist_id)
5   sqx1 ← (s_quant_by_ol_o_id).get(ol_o_id=dnox.next_o_id)
6   sqx = CHOOSE sqx1 #Filter by w_id and d_id | Filter by s_quant
7   print sqx
8 }SER
```

Listing 4: StockLevel Transaction

```
# Group columns are sometimes replaced with "info"
# for better readability
Delivery(dist_id, carr_num, curr_time) := {
  nox1← (no_by_no_d_id).get(no_d_id=dist_id);
  nox CHOOSE nox1; #Pick the lowest order id
  (d_next_o_id_by_id).delete(id=nox.o_id);
  ox← (order_by_id).get(id=nox.o_id);
  (oder_by_id).put(ox[o_carier_id ↦ carr_num]);
  olx1← (ol_info_by_ol_o_id).get(ol_o_id=nox.o_id);
  olx CHOOSE olx1; #Filter by w_id and d_id
  FOREACH r IN olx DO
    (ol_info_by_ol_o_id).put(r[ol_info↦curr_time]);
    s = s + r.ol_info
  END;
  cx← (c_info_by_id).get (id=ox.c_id)
  (c_info_by_id).put(cx[c_delivery_cnt ↦c_delivery_cnt + 1][c_balance ↦ c_balance - s]);
  # denormalized objects are not updated yet
}SER
```

Listing 5: Delivery Transaction