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])
ol_by_s_i_id := (OrderLine $\bowtie$ Stock,(id,s_i_id),[ol_o_id,...])

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)
no_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

New Order :

```
# some non-interesting updates are eliminated
NewOrder(wh_id,dist_id,cust_id,item_list,ol_quant) := {
  wx= (warehouse_by_id).GET (id=wh_id) #Retrieve warehouse by PK
  dtx= (d_tax_by_id).GET (id=dist_id) #Retrieve d_tax by PK
  #Update d_next_o_id by PK:
  dnoix= (d_next_o_id_by_id).GET (id=dist_id)
  (d_next_o_id_by_id).PUT(dnoix[d_next_o_id ↦d_next_o_id+1]);
  cx= (c_info_by_id).GET (id=(cust_id,...)) #Retrieve customer by PK
  #Enter new rows into Order and NewOrder objects (3 Objects):
  (order_by_id).PUT(...); #new row is created from known values
  (o_info_by_o_c_id).PUT(...); #structure of the new row should match the
     denormalized object
  (no_by_d_id).PUT(...);

  FOREACH item_id IN item_list DO
     ix= (item_info_by_id).GET (id=item_id)
     #Retrieve Stock information by PK (from 3 objects):
     socx= (s_orderCnt_by_id).GET (id=(item_id,...))
     sqx= (s_quant_by_id).GET (id=(item_id,...))
     six= (s_info_by_id).GET (id=(item_id,...))
     IF (sqx - ol_quant < 10)
        (s_quant_by_id).PUT(sqx[s_quant↦(s_quant-ol_quant+91)]);
        olx= (ol_by_s_id).GET (s_id=(item_id)) #All OL using this stock
        FOREACH o_id IN ol_x DO
           (s_quant_by_ol_o_id).PUT(...,sqx[s_quant ↦(s_quant - ol_quant+91)],...);
        END;
     ELSE
        (s_quant_by_id).PUT(sqx[s_quant ↦s_quant - ol_quant]);
        olx= (ol_by_s_id).GET (s_id=(item_id)) #All OL using this stock
        FOREACH o_id IN ol_x DO #update the denormalized join object
           (s_quant_by_ol_o_id).PUT(...,sqx[s_quant ↦(s_quant - ol_quant)],...);
        END;
  END;


}SER
```

Listing 1: NewOrder Transaction

Payment :

```
Payment(wh_id,dist_id,cust_id,cust_name,amnt) := {
  wx= (warehouse_by_id).GET (id=wh_id) #Retrieve warehouse by PK
  (warehouse_by_id).PUT(wx[w_ytd ↦w_ytd+1]); #Update the ytd of the wrhs
  dx= (d_ytd_by_id).GET (id=dist_id) #Retrieve d_ytd by PK
  (d_ytd_by_id).PUT(dx[d_ytd ↦d_ytd+1]); #Update the ytd of the district

  # Retrive customer info (except c_balance):
  IF (cust_id = NULL) #Retrieve by id or name?
  THEN cx1= (c_info_by_name).GET (c_name=cust_name);
       cx = CHOOSE cx1 # pick the middle customer;
  ELSE cx= (c_info_by_id).GET (id=(cust_id,...)) #Retrieve customers by PK
  (c_info_by_id).PUT(cx
                [c_ytd_payment↦c_ytd_payment+amnt]
                [c_payment_cnt↦c_payment_cnt+1]);

  # Retrive and update customer's balance:
  IF (cust_id = NULL) #Retrieve by id or name?
  THEN cbx1= (c_balance_by_name).GET (c_name=cust_name);
       cbx = CHOOSE cbx1 # pick the middle customer;
       #Update both customer objects:
       (c_balance_by_id).PUT(cbx [c_balance↦c_balance-amnt]);
       (c_balance_by_name).PUT(cbx [c_balance↦c_balance-amnt]);
  ELSE cbx= (c_balance_by_id).GET (id=(cust_id,...))#Retrieve customers by PK
       #Retrieve the same customer's info
       cix= (c_info_by_id).GET (id=(cust_id,...))#Retrieve customer by PK
       # Update both objects:
       (c_balance_by_id).PUT(cbx [c_balance↦c_balance-amnt]);
       (c_balance_by_name).PUT((cix.name,cust_id,cbx.c_balance-amnt));
  (h_info_by_id).PUT(wh_id,dist_id,...);
}SER
```

Listing 2: Payment Transaction

Order Status :

```
OrderStatus(cust_id,cust_name) := {
  IF (cust_id = NULL) #Retrieve by id or name?
  THEN cx1= (c_info_by_name).GET (c_name=cust_name);
       cx = CHOOSE cx1 # pick the middle customer;
  ELSE cx= (c_info_by_id).GET (id=(cust_id,...)) #Retrieve customers by PK
  ox1= (o_info_by_o_c_id).GET(o_c_id=cx.id); #Retrieve orders by non-PK
```

```
  ox =  CHOOSE ox1 ; # pick the largest order o_id
  olx= (ol_info_by_ol_o_id).GET(ol_o_id=ox.o_id); #Retrieve OrdLn by non-PK
  print olx
}SER
```

Listing 3: OrderStatus Transaction

Stock Level :

```
StockLevel(dist_id,wh_id) := {
  #Retrieve d_next_o_id by PK:
  dnox= (d_next_o_id_by_id).GET (id=(wh_id,dist_id))
  sqx1 = (s_quant_by_ol_o_id).GET(ol_o_id=dnox.next_o_id)
  sqx = FILTER sqx1 #Filter by w_id and d_id and by s_quant
  print sqx
}SER
```

Listing 4: StockLevel Transaction

Delivery :

```
Delivery(dist_id, carr_num, curr_time) := {
  nox2= (no_by_d_id).GET(no_d_id=dist_id); #Retrive by partial key
  nox1 = FILTER(nox2); #Filter records by W_id
  nox = CHOOSE(nox1); #Pick the record with the lowest o_id
  (no_by_d_id).DELETE(id=(nox.o_id,...)); #Delete by PK
  ox = (order_by_id).GET(id=(nox.o_id,...)); #Retrive order by PK
  (order_by_id).PUT(ox[o_carier_id ↦carr_num]); #Update the carrier ID
  (o_id + ...._by_o_c_id).PUT(ox'[o_carier_id ↦carr_num]); #Update the carrier ID
  # ox' only includes interesting columns from ox
  olx1= (ol_info_by_ol_o_id).GET(ol_o_id=nox.o_id);
  olx = FILTER olx1 ; #Filter by w_id and d_id
  s = 0;
  FOREACH olr IN olx DO
     (ol_info_by_ol_o_id).PUT(olr[ol_info↦curr_time]);
     (ol_info_by_id).PUT(olr[ol_info↦curr_time]);
     s = s+ olr.ol_info
  END;
  cx← (c_info_by_id).GET (id=ox.c_id) #Retrive customer by PK
  #Update c_info_by_id and c_balance_by_id:
```

```
20    (c_info_by_id).PUT(cx[c_delivery_cnt ↦ c_delivery_cnt + 1]);#update deliveryCnt
21    (c_info_by_id).PUT(cx[c_balance ↦ c_balance - s]); #update delivery cnt
22    #Update c_info_by_name and c_balance_by_name:
23    (c_info_by_name).PUT(cx'[c_delvry_cnt↦c_delvry_cnt+1]);#update deliveryCnt
24    (c_info_by_name).PUT(cx'[c_balance ↦ c_balance - s]); #update delivery cnt
25  }SER
```

Listing 5: Delivery Transaction