

1 Syntax of simpSQL

The following is the formal definition of the simpSQL language based on Kartik's document, representing a simple programming 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 kvSQL¹.

$$\begin{aligned}
 & t \in \text{TableName} \quad f_{id}, f_v \in \text{FieldName} \quad v \in \text{Value} \\
 & x \in \text{Variable} \quad txn \in \text{TxnName} \\
 & \odot \in \{<, \leq, =, \neq, >, \geq\} \quad \oplus \in \{\cap, \cup\} \quad \otimes \in \{\wedge, \vee\} \\
 & pk ::= (\overline{f_{id}}, \overline{f_v}) \\
 & obj ::= (t, pk, \overline{f_v}) \\
 & r_{obj} ::= \bar{v} \\
 & \phi_{pk} ::= pk_{id} \odot v \mid pk_v \odot v \mid \phi_{pk} \otimes \phi_{pk} \\
 & e ::= x \mid \text{CHOOSE } x \mid r_{obj} \mid e \oplus e \\
 & \phi_c ::= r_{obj}^i \odot v \mid r \text{ IN } e \mid \phi_c \otimes \phi_c \\
 & op ::= obj.put(r) \mid x \leftarrow obj.get(\phi_{obj}) \\
 & c ::= \{\overline{op}\}_{DC} \mid x \leftarrow e \mid \\
 & \quad \text{IF } \phi_c \text{ THEN } c \text{ ELSE } c \mid c; c \mid \{c\}_{SER} \mid \\
 & \quad \text{FOREACH } r \text{ IN } x \text{ DO } c \text{ END}
 \end{aligned}$$

Figure 2: Syntax of kvSQL

¹proving 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

<u>w_id</u>	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

<u>d_id</u>	<u>d_w_id</u>	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

<u>c_id</u>	<u>c_d_id</u>	<u>c_w_id</u>	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

<u>o_id</u>	<u>o_d_id</u>	<u>o_w_id</u>	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

<u>i_id</u>	i_info

kvSQL Object(s): Item

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

SimpSQL Table: OrderLine

<u>ol_o_id</u>	<u>ol_d_id</u>	<u>ol_w_id</u>	<u>ol_number</u>	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

<u>s_i_id</u>	<u>s_w_id</u>	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

<u>ol_o_id</u>	<u>ol_d_id</u>	<u>ol_w_id</u>	<u>ol_number</u>	ol_info	<u>s_i_id</u>	<u>s_w_id</u>	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

<u>ol_o_id</u>	<u>ol_d_id</u>	<u>ol_w_id</u>

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

<u>h_id</u>	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 :

```
1 # some non-interesting updates are eliminated
2 NewOrder(wh_id,dist_id,cust_id,item_list) := {
3   wx=(warehouse_by_id).GET (id=wh_id) #Retrieve warehouse by PK
4   dtx=(d_tax_by_id).GET (id=dist_id) #Retrieve d_tax by PK
5   #Update d_next_o_id by PK:
6   dnoix=(d_next_o_id_by_id).GET (id=dist_id)
7   (d_next_o_id_by_id).PUT(dnoix[d_next_o_id ↦ d_next_o_id+1]);
8   cx=(c_info_by_id).GET (id=(cust_id,...)) #Retrieve customer by PK
9   #Enter new rows into Order and NewOrder objects (3 Objects):
10  (order_by_id).PUT(...); #new row is created from known values
11  (o_info_by_o_c_id).PUT(...); #structure of the new row should match the
    denormalized object
12  (no_by_d_id).PUT(...);
13
14  FOREACH item_id IN item_list DO
15    ix=(item_info_by_id).GET (id=item_id)
16    #Retrieve Stock information by PK
17    socx=(s_orderCnt_by_id).GET (id=(item_id,...))
18    sqx=(s_quant_by_id).GET (id=(item_id,...))
19    six=(s_info_by_id).GET (id=(item_id,...))
20    IF (sqx - ol_quant < 10)
21      (s_quant_by_id).PUT(sqx[s_quant ↦ s_quant - ol_quant + 91]);
22      (s_quant_by_ol_o_id).PUT(...);
23    ELSE
24      (s_quant_by_id).PUT(sqx[s_quant ↦ s_quant - ol_quant]);
25
26  END;
27
28
29 }SER
```

Listing 1: NewOrder Transaction

Payment :

```
1 Payment(wh_id,dist_id,cust_id,cust_name,amnt) := {
2   wx=(warehouse_by_id).GET(id=wh_id) #Retrieve warehouse by PK
3   (warehouse_by_id).PUT(wx[w_ytd↦w_ytd+1]); #Update the ytd of the wrhs
4   dx=(d_ytd_by_id).GET(id=dist_id) #Retrieve d_ytd by PK
5   (d_ytd_by_id).PUT(dx[d_ytd↦d_ytd+1]); #Update the ytd of the district
6
7   # Retrive customer info (except c_balance):
8   IF (cust_id = NULL) #Retrieve by id or name?
9   THEN cx1=(c_info_by_name).GET(c_name=cust_name);
10      cx = CHOOSE cx1 # pick the middle customer;
11   ELSE cx=(c_info_by_id).GET(id=(cust_id,...)) #Retrieve customers by PK
12   (c_info_by_id).PUT(cx
13      [c_ytd_payment↦c_ytd_payment+amnt]
14      [c_payment_cnt↦c_payment_cnt+1]);
15
16   # Retrive and update customer's balance:
17   IF (cust_id = NULL) #Retrieve by id or name?
18   THEN cbx1=(c_balance_by_name).GET(c_name=cust_name);
19      cbx = CHOOSE cbx1 # pick the middle customer;
20      #Update both customer objects:
21      (c_balance_by_id).PUT(cbx [c_balance↦c_balance-amnt]);
22      (c_balance_by_name).PUT(cbx [c_balance↦c_balance-amnt]);
23   ELSE cbx=(c_balance_by_id).GET(id=(cust_id,...))#Retrieve customers by PK
24   #Retrieve the same customer's info
25   cix=(c_info_by_id).GET(id=(cust_id,...))#Retrieve customer by PK
26   # Update both objects:
27   (c_balance_by_id).PUT(cbx [c_balance↦c_balance-amnt]);
28   (c_balance_by_name).PUT((cix.name,cust_id,cbx.c_balance-amnt));
29   (h_info_by_id).PUT(wh_id,dist_id,...);
30 }SER
```

Listing 2: Payment Transaction

Order Status :

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

```

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

```

Listing 3: OrderStatus Transaction

Stock Level :

```

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

```

Listing 4: StockLevel Transaction

Delivery :

```

1 Delivery(dist_id, carr_num, curr_time) := {
2   nox2=(no_by_d_id).GET(no_d_id=dist_id); #Retrive by partial key
3   nox1 = FILTER(nox2); #Filter records by W_id
4   nox = CHOOSE(nox1); #Pick the record with the lowest o_id
5   (no_by_d_id).DELETE(id=(nox.o_id,...)); #Delete by PK
6   ox =(order_by_id).GET(id=(nox.o_id,...)); #Retrive order by PK
7   (order_by_id).PUT(ox[o_carrier_id ↦ carr_num]); #Update the carrier ID
8   (o_id + ..._by_o_c_id).PUT(ox[o_carrier_id ↦ carr_num]); #Update the carrier ID
9   # ox' only includes interesting columns from ox
10  olx1=(ol_info_by_ol_o_id).GET(ol_o_id=nox.o_id);
11  olx = FILTER olx1 ; #Filter by w_id and d_id
12  s = 0;
13  FOREACH olr IN olx DO
14    (ol_info_by_ol_o_id).PUT(olr[ol_info ↦ curr_time]);
15    (ol_info_by_id).PUT(olr[ol_info ↦ curr_time]);
16    s = s+ olr.ol_info
17  END;
18  cx← (c_info_by_id).GET(id=ox.c_id) #Retrive customer by PK
19  #Update c_info_by_id and c_balance_by_id:

```



```

20  (c.info.by_id).PUT(cx[c_delivery_cnt  $\mapsto$  c.delivery_cnt + 1]);#update deliveryCnt
21  (c.info.by_id).PUT(cx[c_balance  $\mapsto$  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 $\mapsto$ c.delvry_cnt+1]);#update deliveryCnt
24  (c.info.by_name).PUT(cx'[c_balance  $\mapsto$  c.balance - s]); #update delivery cnt
25  }SER

```

Listing 5: Delivery Transaction