# 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.

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

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

Figure 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 standard SQL to kvSQL[1].

[1]proving this translation correct is NOT giong to be 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

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

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

Figure 2: Syntax of kvSQL

# 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

```
new order
NEW ORDER!
```

Listing 1: NewOrder Transaction

```
payment
payment
payment
payment
payment
```

Listing 2: Payment Transaction

```
order status
order status
order status
order status
```

Listing 3: OrderStatus Transaction

```
Stock level
Stock level
Stock level
Stock level
Stock level
```

Listing 4: StockLevel Transaction

```
delivery
delivery
delivery
delivery
delivery
```

Listing 5: Delivery Transaction