08_Tables

September 22, 2019

1 8. Tables

- Characteristics of q tables
  - First-class entities
  - A collection of named columns implemented as a dictionary of lists
  - Column oriented
  - Ordered records as lists are ordered
  - Efficient at storing, retreiving and manipulating sequential data
  - Kdb+ handles relational and time series data in a unified environment of q tables
    - No separate data definition language
    - No separate stored procedure language
    - No need to map internal representations to a separate form of persistence

1.1 8.1. Table definition

- Method 1: flip `colName1`colName2!(list1;list2)
  - creates a named column list and flips it to get a table
- Method 2: ([] colname1:list1;colname2:list2)

- Functions on tables:
  - cols: column names
  - meta: metadata of the table:
    - c:column names;
    - t: type char of the colulmn;
    - f: foreign key domain
    - a: attributes associated with the column
  - count (or records/rows)
  - value table[rowIndex]

1.2 8.2. Empty tables and schema

[ ]:
```
emptyTable:([] colName1:();colName2:()) / creating an empty table
```

[ ]:
```
emptyTable
```

[ ]:
```
teTable:([] name:`$();attribute:`int$())
```

[ ]:
```
teTable,:(`ati;20) / add row to table
```

[ ]:
```
teTable[0;`name]:`qqq
```

[ ]:
```
teTable
```

- Basic select syntax

[ ]:
```
select ticker from trade_table
```

- Basic update syntax

[ ]:
```
update name:`aaa from teTable where name=`qqq
```

[1]:
```
testt:([] name:`Dent`Beeblebrox`Prefect; iq:98 42 126)
```

[2]:
```
meta testt
```

[2]:
```
c   | t f a
----| -----
name| s
iq  | j
```

[ ]:
```
update iq:iq%100 from testt
```

1.3 8.4. Primary keys and keyed tables

- A keyed table in q is like a table in SQL with a primary key column
- A keyed table is a dictionary mapping a table of key records to a table of value records
- Uniqueness is not checked in primary keys:
  - values associated with non-unique keys can only be retreived with a select clause

[ ]:
```
ktr[1;`ticker]
```

- Get multiple records from a keyed table:

[ ]:
```
ktr[(flip enlist 1 2)] / method 2
```

[ ]:
```
([] id: 1 3)#ktr / method 2
```

[ ]:
```
select from ktr where id in (2;4;6)
```

- Reverse lookup: use an anonymous table:

[ ]:
```
ktr?([] ticker:`aapl`ibm)
```

[ ]:
```
key ktr
cols ktr
value ktr
meta ktr
```

- Create key from a table's column:
  - keyColumn xkey table_with_key_column
  - columnNumber!table_with_key_column
- Remove key:
  - () xkey keyed_table
  - 0!keyed_table

[ ]:
```
2!() xkey keyed2
```

- Compound primary key: just use two columns in keyed table definition
- (Typed) empty keyed table: same as typed empty table
- Getting multiple values: list of compound keys
- Extracting column data: table[anonymous table with list of keys][column name]

1.4 8.5. Foreign keys and virtual columns

[ ]:
```
```

1.5 Reading in a table from .csv

[ ]:
```
t_path:"/home/iguana/1_Code/00_datasets/kaggle_us-stocks/csv_data/Stocks/"
ticker_name:"aapl"
ext:".us.csv"
pattern:"DFFFFII"
```

[ ]:
```
file_name:t_path,ticker_name,ext
```

[ ]:
```
file_h:hsym `$file_name
```

[ ]:
```
csvread:{[fn;ptrn] (ptrn;enlist csv) 0: hsym `$fn}
```

[ ]:
```
ttable:csvread[file_name;pattern]
```

[ ]:
```
ttable
```

1.6 Creating a table with random data

[ ]:
```
mySym:`aapl`googl`ibm`fb
```

[ ]:
```
symEnum:`mySym$10?mySym / tickers
```

[ ]:
```
symList:10?mySym
```

[ ]:
```
myDates:10#2019.03.01 / dates
```

[ ]:
```
myTimes:asc 10?24:00:00.000000000
```

[ ]:
```
myVolume:10*1+10?1000
```

[ ]:
```
id:1+til 10
```

[ ]:
```
prices:90.0+(10?2001)%100
```

[ ]:
```
trade_table:([] ticker:symList;date:myDates;time:myTimes;price:prices;volume:
 ↪myVolume)
```

[ ]:
```
ktr:([id:1+til 10] ticker:symList;date:myDates;time:myTimes;price:prices;volume:
 ↪myVolume)
```

[ ]:
```
unkeyed:([] id:id;ticker:symList;date:myDates;time:myTimes;price:prices;volume:
 ↪myVolume)
```

[ ]:
```
keyed2:`id xkey unkeyed
```