《kdb+中文教程》第一章 简介

Original kdbcn kdb中文教程 2020年12月4日 05:30

提示：（1）建议按本书章节顺序阅读；（2）本书含有代码、表格等，在PC上阅读可获得更好体验。

第一章 简介

kdb+/q是一款独特的数据库产品。本章将简单介绍它的核心优势及许可类型，并进一步通过实例，快速演示如何安装启动、如何在控制台进行一些常见操作，如何实现数据表的简单操作等。

第一节 概况

一、核心优势

kdb+是一款独一无二、性能极高、堪称天才之作的数据库产品，由ArthurWhitney开发，KxSystems公司（https://kx.com）于2003年推出，其前身k和kdb/ksql分别于1993年和1998年推出。kdb+软件大小不足1MB，却集时间序列数据库、内存数据库、磁盘数据库等为一体，

提供了流数据、实时数据、历史数据的采集、存储、分析、检索一站式解决方案。kdb+主要应用于金融证券行业，目前扩大到制造业、电信、汽车、航天、物联网、零售等各个领域。

kdb+内置专用矢量语言q，可以直接对数据库中的数据进行操作，而无需将数据传输到其他应用程序进行分析。q语言是一种抽象编程的APL系语言，是一种基于矢量的处理语言，非常适合低成本地对大量数据进行复杂的计算。而且q语言跟SQL有一些类似，对于有数据库基础的人较容易掌握。

kdb+/q的核心优势[参考文献2]主要有：

（一）强大的一体化平台

1、流数据处理、内存数据库和磁盘数据库的一体化。许多应用场景需要同时具备流数据处理、内存数据库和磁盘数据库等，三者通常由不同的软件实现，kdb+用单一软件即可用于流数据、实时数据及历史数据的处理。

2、数据的采集、存储和分析的应用一体化。通常数据的采集、存储和分析是分离的，同一架构无法同时支持，而kdb+轻松实现了三者的一体化。

3、数据库和开发语言和的一体化。传统数据分析在某种意义上是一种数据移动的过程，要分析的数据从数据库转移到分析服务器或者分析程序上进行处理，kdb+通过库内分析（in-databaseanalytics），消除将海量数据移动到分析程序的开销，提升了数据分析处理效率。

4、代码与数据一体化。传统的数据库或语言，代码是代码，数据是数据，一般难以以统一的方式进行处理，kdb+将代码和数据进行了抽象，实现了代码与数据的一体化。

（二）高性能的列式数据库

1、列式存储结构简化了索引与联接，提高了查询速度。传统关系数据库针对事务更新进行了优化，采用行式存储和磁盘随机分布以便于并发快速写入，但查询时必须进行磁盘扫描。kdb+采用列式存储和有序化存储，数据聚集和列向查询性能得到极大提升，同时在数据库运行时可以方便地对数据列进行增删改。

2、kdb+可以利用列的有序性(如时间戳字段)优化查询，迅速定位到所需数据子集。

3、内存数据库可以实时更新索引，方便对流数据进行分类汇总统计，比如VWAP计算。

（三）强大的编程和查询语言q

1、kdb+内置了高速应用开发和数据查询的一体化语言—q。q可以直接对数据做出运算，最大限度的减少中间成本。它无需先读取数据，然后再送到外部的程序进行分析，当接收到数据时就能马上进行处理。q语言使用同样方式处理原子数据和高维数组，代码精炼，运行高效，易于并行扩展。

2、数组、字典和表都是原始数据类型，而其核心原语是专为此类数据而设计，例如对表作算术运算。单一操作就能作用于千万笔记录上，就像操作一笔记录那么容易。

3、q语言内置的时间数据类型，高度优化了对时间序列数据的查询。日期、时间和纳秒级时间戳都是基本数据类型，可高效实现时间序列统计和OLAP查询。

4、每秒钟能处理数以千万计的记录，而每天则以亿计。历史数据库的记录则以万亿计。它的速度能应付最高的数据流量，更可以配合硬件加速器以获得最高速度和最大的灵活性。

5、函数式编程语言和交互式开发环境，使得开发效率极高。

（四）简单的数据管理和低廉的成本

1、原生64位架构，可以适应当前海量数据处理需求。

2、kdb+自身占用极少的内存和磁盘资源，通常部署于多处理器服务器上，但可任意动态扩展。

3、kdb+无需复杂的安装和配置，可简单部署在多种操作系统下共同运行。

4、与传统数据库繁重的管理任务相比，kdb+平台极其简单明了，提供更低的拥有成本和更高的运作效率。

5、数据库就是操作系统中的原生文件，数据库管理由操作系统功能直接完成。

6、访问控制、高可用性、事务日志、容量规划和其他企业级特性都在应用层完成，可方便与现有系统集成。

（五）优越的可移植性和互操作性

1、kdb+使用ANSI C开发，未使用任何专有扩展，可快速移植到最新的芯片和操作系统之上。

2、API非常简洁，可以很容易地连接到外部的图形、表格生成和旧系统。除了用于标准数据库技术的ODBC和JDBC连接器之外，还有一些接口C/C++，Java，Python和.Net，可帮助在kdb+与传统数据库或Excel等应用程序之间迁移数据。

3、kdb+可以直接解析常见的数据文件格式如csv和xml。

4、用户可以通过kdb+内置的web服务器功能直接访问数据。

5、动态索引让kdb+能有效地利用实时数据。

6、kdb+系统可运行于Linux、Solaris、Windows、MacOSX等64位服务器平台。

（六）分布式并行扩展以保证查询速度

1、内置多线程并行计算功能。计算效率跟CPU的数目成正比，应用程序能利用多核心处理器的优势，而无需编写特殊的多线程程序。

2、支持并行访问庞大的历史数据库，所以能将查询分配至多个内核或多台机器。

3、kdb+使用分区技术支持并行扩展，可以分配特定的CPU和磁盘到特定查询操作。

4、kdb+的进程间通讯功能可以轻易支持多服务器的并行计算或网格计算。

二、许可类型

为满足不同的需求，kdb+提供了多种许可类型可供选择。下表列出了各种许可的比较。[https://kx.com/connect-with-us/licenses/]

版本	学术版	32 位个人版	64 位个人版	不限期限版	订阅版	按需定(On Demand)	OEM 版
费用	免费	免费	免费	按核数收授权费，按年收维护费（可选）	按每年每核收费	按核数及使用分钟数收费	按安排
Linux/Mac/Windows 版本	√	√	√	√	√	√	√
64 位版本	√	X	√	√	√	√	√
允许商业用途	X(仅用于研究和教学)	X	X	√	√	√	√
需要联网	X	X	√	X	X	√	X
允许在云上运行	√	√	X	√	√	√	√
允许在本地运行	√	√	√	√	√	√	√
支持 Kx Developer 模块	√	X	√	√	√	√	√
支持 Kx Analyst 模块	√	X	√	√	√	√	√
获得技术支持	X	X	X	√	√	按安排	√
进入客户问答论坛	X	X	X	√	√	√	√
获得社区支持（GitHub / Google 网上论坛等）	√	√	√	√	√	√	√
立即获得升级、补丁和 Beta 版本	X	X	X	√	√	√	√
定期升级	√	√	√	√	√	kdb中文教程	

注：√表示支持，X 表示不支持。

第二节 快速入门

本节将介绍kdb+软件的下载与安装，常见操作及数据表的基本操作。

一、安装与启动

kdb+32 位版本和 64 位个人版可以分别从 https://kx.com/download 和 https://ondemand.kx.com下载（中国大陆地区可能无法直接访问），64位个人版还可以从 https://anaconda.org/kx/kdb/files下载。64位个人版需要申请授权文件kc.lic。

kdb+软件主要包括q.exe和q.k两个文件，大小不足1MB，可以简单直接安装。下面以kdb+forWindows 32位版本为例，介绍kdb+软件具体安装及启动。

将下载的压缩文件解压至c:\q，确保q.k位于c:\q文件夹，q.exe位于c:\q\w32文件夹，点击c:\q\w32\q.exe即可启动。如果解压到其它文件夹如d:\kdb\q（本书大多假设q安装于该文件夹），需要设置环境变量QHOME指向q.k所在目录，即set QHOME=d:\kdb\q（或者setx QHOME"d:\kdb\q"将在注册表中写入环境变量），然后运行d:\kdb\q\w32\q.exe。为了使其它软件可以连接kdb+，通常在启动时通过-p选项指定监听端口，如d:\kdb\q\w32\q.exe-p 5001。

可以创建批处理文件（如q.bat），用于启动kdb+：

```
set   QHOME=d:\kdb\q
start   "q(5001)"    d:\kdb\q\w32\q.exe -p 5001
```

运行成功后，会显示如下窗口。窗口的第一行显示所运行kdb+的版本序列，第二行显示位数、核数、授权信息等，用户可以在提示符“版本序后可输入q语句。

q为一个控制台程序，对于新手而言在控制台输入命令并不方便，往往需要借助IDE或代码编辑器。主要IDE或编辑器有：Studiofor kdb+、qpad、qstudio、Notepad++、VSCode、sublime等。不过我们仍然建议初学者在控制台测试本书各种命令，当语句比较复杂时，再使用IDE，IDE相关内容详见第十章。

二、基础操作

我们通过在q控制台输入一些语句，来了解一下q的简单操作。

q)1+2　/在q控制台输入语句，回车后显示计算结果

3

注意，“/”前面要有至少一个空格，表示“/”后面的内容为注释。

q)"Hello kdb+"　/字符串

"Hellokdb+"

q)a:1　/a为一变量，:为赋值操作，将1赋值到变量a

q)a　/显示变量值

1

q)A:2　/变量区分大小写

q)A

2

q)a:aa:1　/多个赋值

q)a

1

q)aa

1

q)a:1;A:2;　/分号为语句结束符，一行可以含有多条语句

q)a+A　/ 加法计算

3

q)1-2　/减法计算

-1

q)1*2　/乘法计算

2

q)1%2　/除法计算，除号为%，不是/

```
0.5
q)2xexp 3  /乘方计算，不是用**或^
8
q)5div 2  /整除
2
q)2*3+4  /结果不是10
14
```

q与其他软件不同，表达式计算顺序是从右到左。2*3+4将先计算右侧的3+4，得到7，然后计算2*7。注意：特殊计算顺序很容易让q新手出现错误，即使q老手也可能出错！如果要按照先乘除后加减运算规则计算，可以用小括号改变运算顺序，即(2*3)+4将得到10，也可以改变一下顺序，如4+2*3。

```
q)1=1  /关系运算：相等，返回布尔值：1b为真，0b为假
1b
q)1>2 /关系运算:大于
0b
q)1>=2 /大于等于
0b
q)1<2  /小于
1b
q)1<=2 /小于等于
1b
q)1<>2  /不等于
1b
q)not1b  /非运算
```

```
0b
q)not0b  /非运算
1b
q)not1>2
1b
q)1band 1b   /且，同  1b & 1b
1b
q)1bor 0b   /或，同  1b | 0b
1b
q)`000001.SH    /符号symbol
`000001.SH
q)`$"中文"   /中文符号不能直接这样写:`中文
`中文
q).z.D  /取当前日期
q).z.N  /取当前时间（纳秒级）
```

除了在控制台直接运行语句，我们还可以将q语句保存在脚本文件中，然后在q中加载运行。例如，我们创建一个名为hello.q的文件，文件内容为show"hello kdb+"，我们将文件保存到q.k所在文件夹，然后在q控制台用系统命令\l加载：

```
q)\lhello.q
```

脚本文件将加载到q，执行结果：

```
"hellokdb+"
```

我们可以用\\或exit 0 退出q。

```
q)\\    /或  exit 0
```

三、数据表操作

（一）数据表创建

创建数据表的方式有许多种，这里介绍两种：一是直接输入数据生成kdb+表，二是从文件中读取数据生成表。

1、输入数据

	sym	date	close
0	000001.SH	2020.03.03	2992.9
1	000001.SH	2020.03.04	3011.67
2	000001.SH	2020.03.05	3071.68
3	399001.SZ	2020.03.04	11493.02
4	399001.SZ	2020.03.05	11711.37
5	399001.SZ	2020.03.06	11582.82

假设要创建一个kdb+表，保存上证综指2020年3月3日、4日、5日以及深证成指2020年4日、5日、6日的收盘指数点位（数据如上图所示）。通过“t:([]字段名:值数组)”的方式，可以创建表，并保存于变量t中，即：

```
q)t:
([]sym:`000001.SH`000001.SH`000001.SH`399001.SZ`399001.SZ`399001.SZ;date:2020.03.032020.03.04 2020.03.05 2020.03.04 2020.03.05 2020.03.06;close:2992.9 3011.673071.68 11493.02 11711.37 11582.82)
q)t
sym      date       close
-----------------------------
000001.SH2020.03.03 2992.9
000001.SH2020.03.04 3011.67
000001.SH2020.03.05 3071.68
399001.SZ2020.03.04 11493.02
399001.SZ2020.03.05 11711.37
399001.SZ2020.03.06 11582.82
```

2、从文件中读取数据

（1）读取q格式文件

在日常应用中，保存数据与读取数据是数据库常见操作。在kdb+中，q格式文件可直接读取。由于此前已创建了t表，我们将其保存后再读取。

```
q)`:d:/kdb/data/tset t;
q)t1:get `:d:/kdb/data/t
q)t1
```

我们将t保存在“d:\kdb\data”路径下，创建了t文件，再通过读取路径下文件创建了t1表。

（2）读取csv格式文件

在日常应用中，我们也经常有导出或导入q以外文件的需求。以逗号分隔的csv文件为例，同样t保存为csv文件再将其导入。

```
q)`:d:/kdb/data/t.csv0: csv 0:t;
q)t2:("SDF";enlist",") 0:`:d:/kdb/data/t.csv
```

我们将t保存在“d:\kdb\data\”路径下创建了t.csv文件，再通过读取路径下文件创建了t2表。

（二）数据表操作

下面简要介绍数据表的基本操作，主要包括表查询及表的分类汇总计算。让读者能够对如何运用kdb+处理表有一个基本认识。需要说明的是，本部分内容仅是基本介绍，数据表操作的详细介绍请参见第六章和第八章。

1、查询

为了便于理解，继续以前面已创建的t表为例，说明kdb+如何实现对数据表的查询。

（1）查看数据表字段数据类型

```
q)meta t
```

```
c   | t f a
-----| -----
sym  | s   s
date | d
close| f
```

“meta表名”可查看表包含哪些字段，各字段是什么类型。kdb+对数据类型准确性要求较高，这意味使用者需要掌握数据类型，避免错误。数据类型的介绍请详见第二、四、五、七章等。

（2）符合条件的所有字段查询

```
q)select from t where sym=`000001.SH
sym      date      close
---------------------------
000001.SH 2020.03.03 2992.9
000001.SH 2020.03.04 3011.67
000001.SH 2020.03.05 3071.68
```

例子中在where语句后加上了筛选条件sym=`000001.SH。运行后将显示满足条件的相关结果。

多个条件用逗号分隔，例：

```
q)select from t where sym=`000001.SH,close>3000
sym      date      close
---------------------------
000001.SH 2020.03.04 3011.67
000001.SH 2020.03.05 3071.68
```

（3）指定字段查询

q)select date,close from t where sym=`000001.SH

date close

2020.03.03 2992.9

2020.03.04 3011.67

2020.03.05 3071.68

当不需要输出所有的变量，上例中我们挑选了date和close变量，也可以用delete加上字段名，去掉不想要的变量，输出想要的变量。

q)delete sym from select from t where sym=`000001.SH

date close

2020.03.03 2992.9

2020.03.04 3011.67

2020.03.05 3071.68

（4）生成新字段

q)update close2:close*1.1 from select from t where sym=`000001.SH

sym date close close2

000001.SH 2020.03.03 2992.9 3292.19

000001.SH 2020.03.04 3011.67 3312.837

000001.SH 2020.03.05 3071.68 3378.848

例子中包含了两个查询，先读取t中sym为`000001.SH的数据，然后根据close数据，计算生成close2变量。

（5）计算并转换类型后生成新字段

q)update datestr:string date from select from t wheresym=`000001.SH

sym date close datestr

000001.SH 2020.03.03 2992.9 "2020.03.03"

000001.SH 2020.03.04 3011.67 "2020.03.04"

000001.SH 2020.03.05 3071.68 "2020.03.05"

经计算并将类型转为字符串得到变量datestr。

（6）查询满足取值范围的记录

q)select from t where i within 2 3

sym date close

000001.SH 2020.03.05 3071.68

399001.SZ 2020.03.04 11493.02

例子中i代表行号，第一行为0。通过限定i的范围，可查询到对应行号的数据。

（7）查询结果按指定变量排序

q)`date xasc t

sym date close

000001.SH 2020.03.03 2992.9

000001.SH 2020.03.04 3011.67

399001.SZ 2020.03.04 11493.02

000001.SH 2020.03.05 3071.68

399001.SZ 2020.03.05 11711.37

399001.SZ 2020.03.06 11582.82

q)`date xasc select from t where date within 2020.03.04 2020.03.05

sym date close

000001.SH 2020.03.04 3011.67

399001.SZ 2020.03.04 11493.02

000001.SH 2020.03.05 3071.68

399001.SZ 2020.03.05 11711.37

例子中按变量date的大小升序（xasc）排列，降序则用xdesc。

（8）结合上下行信息创建变量

q)update ret:-1+close%prev close from `sym`date xasc select from twhere sym=`000001.SH

sym date close ret

000001.SH 2020.03.03 2992.9

000001.SH 2020.03.04 3011.67 0.006271509

000001.SH 2020.03.05 3071.68 0.01992582

由于金融数据多数为时间序列数据，往往需要结合上下行信息进行操作。kdb+可轻松实现这一目的。例子中计算每个交易日较上一交易日的指数涨跌幅，通过prev函数可读取上一行信息。“nxprev 字段名”可以读取前n行信息，若n为负数则可实现往后读取信息。

2、分类汇总

（1）取出每个分组最后一条记录

q)select by sym from t

sym | date close

---| ---

000001.SH| 2020.03.05 3071.68

399001.SZ| 2020.03.06 11582.82

例子通过“bysym”实现按指数代码分组，读取每一指数最后一条满足条件的记录。

（2）分组时使用各种聚合函数

q)select n:count i,date1:first date,date2:last date,close1:minclose,close2:max close by sym from t

sym | n date1 date2 close1 close2

---------| --

000001.SH| 3 2020.03.03 2020.03.05 2992.9 3071.68

399001.SZ| 3 2020.03.04 2020.03.06 11493.02 11711.37

例子中实现按指数分组，统计记录数、第一日、最后一日、最大值、最小值。

（3）分组滚动计算

q)update sumsclose:sums close,maxsclose:maxs close,minsclose:minsclose by sym from t

sym date close sumsclose maxsclose minsclose

000001.SH 2020.03.03 2992.9 2992.9 2992.9 2992.9

000001.SH 2020.03.04 3011.67 6004.57 3011.67 2992.9

000001.SH 2020.03.05 3071.68 9076.25 3071.68 2992.9

399001.SZ 2020.03.04 11493.02 11493.02 11493.02 11493.02

399001.SZ 2020.03.05 11711.37 23204.39 11711.37 11493.02

399001.SZ 2020.03.06 11582.82 34787.21 11711.37 11493.02

sums可实现对数据滚动加总，maxs实现滚动取最大值，mins实现滚动取最小值。

（4）移动平均计算

q)updatemavgclose:mavg[2;close],mmaxclose:mmax[2;close],mminclose:mmin[2;close] by symfrom t

```
 sym       date       close    mavgclose mmaxclose mminclose
-------------------------------------------------------------
000001.SH 2020.03.03 2992.9   2992.9    2992.9    2992.9
000001.SH 2020.03.04 3011.67  3002.285  3011.67   2992.9
000001.SH 2020.03.05 3071.68  3041.675  3071.68   3011.67
399001.SZ 2020.03.04 11493.02 11493.02  11493.02  11493.02
399001.SZ 2020.03.05 11711.37 11602.2   11711.37  11493.02
399001.SZ 2020.03.06 11582.82 11647.09  11711.37  11582.82
```

mavg可实现每n个数据求平均值，mmax和mmin可分别实现每n个数据求最大值、最小值。这些函数可轻松实现数据的移动计算，在金融数据计算中应用十分普遍。

通过以上例子可以看出，与一般SQL相比，kdb+可以更加轻松地实现各种计算。上述例子只是一些简单操作，更多操作将在后续章节逐步介绍。