版权声明

目录

（第一、二章所有内容以及第三至十章每章部分内容可免费阅读，详见微信公众号 kdbcnbook 或 https://kdbcn.gitee.io）

前　言

kdb+号称是世界上最快的内存数据库，q 是 kdb+的内置语言。事实上 kdb+/q 不只是内存数据库，更是一款高性能大数据平台，它使用统一的数据库处理实时数据和历史数据，同时具备 CEP(复杂事件处理)引擎、内存数据库、磁盘数据库等功能。与 SQL Server、Oracle、MySQL 等传统关系数据库及 Hadoop、Spark 等现代大数据平台相比，kdb+/q 具有更快的速度和更低的总拥有成本，非常适合海量数据的快速采集、存储、分析、处理和检索等。kdb+/q 最初主要被用于金融机构海量数据分析和高频交易，目前在人工智能、机器学习、物联网、智能电网、航天、国家情报等领域发挥越来越大的作用。

据作者所知，本书是国内第一本关于 kdb+/q 的中文入门书籍。在 https://code.kx.com 等网站上有关于 kdb+/q 的英文教程，但目前还没有关于 kdb+/q 的中文书籍。因此，我们若干个 kdb+/q 爱好者组织起来，编写了这本中文教程，为有兴趣学习 kdb+/q 的读者，提供一本快速入门教程。

我们认为，本书将为数据处理人员、量化投资者、量化平台开发人员、软件工程师、统计和机器学习专业人员和学生，以及对大数据分析感兴趣的人员提供一本方便有用的入门教程，以便他们能够尽快地熟悉和使用 kdb+/q。

本书分为三大部分。第一部分（第一章）为 kdb+/q 简介，主要介绍 kdb+/q 的优势及不同许可类型，同时介绍了 kdb+/q 的下载、安装、基本操作及数据表操作等。通过学习本章，读者可以快速了解 kdb+/q 的特性，同时对 kdb+/q 的独特、简洁等有一个初步直观感受。

第二章至第八章为第二部分，是本书的核心内容，分别为数据类型、数组、函数、字典、表与 qSQL、I/O 操作及数据库。首先介绍了 kdb+/q 的基本数据类型，然后介绍了 kdb+/q 的数组（列表）、函数、字典、表，最后介绍了文件 I/O 操作、进程间通讯、数据库构建等。通过这部分内容的学习，读者可以掌握 kdb+/q 的基础知识，为 kdb+/q 的运用打下坚实基础，逐步将 kdb+应用于实际场景，同时能进一步感受 kdb+/q 的简洁、灵活与强大等。

第三部分包括第九章和第十章。第九章通过实例介绍 kdb+在股票期货数据处理方面的常见应用，包括历史行情数据库构建、策略回测与优化、实时行情处理等，并简单介绍了企业级开源证券期货交易平台 Tx。第十章把一些可能有用的知识点以问答形式列出来，方便读者需要时查找。通过这部分内容的学习，读者可以参考常见应用实例，举一反三。

本书由陈青、李书恣、钱嘉韵、吴晶、杨琨、张文璋、张瀛文、郑轶（按姓名拼音顺序排列，下同）负责撰写，由张文璋、郑轶负责统稿。本书在撰写过程中，参考了一些互联网资料和书籍，在此表示感谢。感谢 Tx 平台作者 itfin、kx 公司许美琳小姐等的帮助。限于作者水平有限和时间限制，书中错误和不足之处在所难免，敬请 kdb+/q 专家及广大读者不吝批评指正（联系方式：微信 kdbcnbook，邮箱 kdbcn@qq.com，网址 kdbcn.github.io / kdbcn.gitee.io）。

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 Arthur Whitney 开发，由 Kx Systems 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[1]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

[1] 参考文献 2。

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-database analytics），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—置。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。[2]

版本	学术版	32 位个人版	64 位个人版	不限期限版	订阅版	按需定制 (On Demand)	OEM 版

[2] 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	按需定制 (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	√	√	√	√	√	√	√

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。

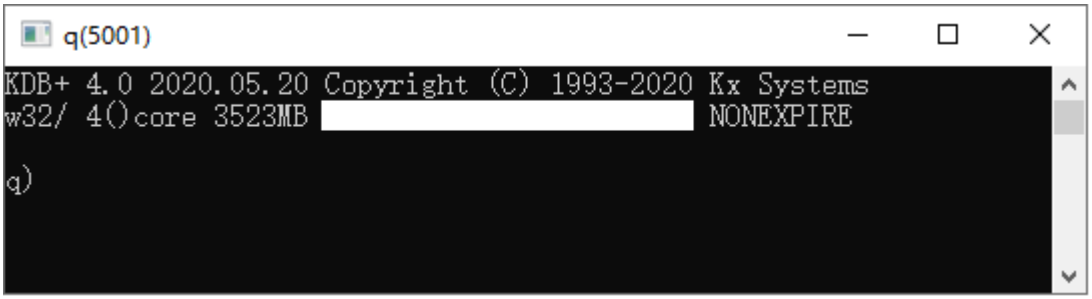
kdb+软件主要包括 q.exe 和 q.k 两个文件，大小不足 1MB，可以简单直接安装。下面以 kdb+for Windows 32 位版本为例，介绍 kdb+软件具体安装及启动。

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，会显示以下窗口。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：Studio for 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
"Hello kdb+"
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)2 xexp 3　/乘方计算，不是用**或^

8

q)5 div 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)not 1b　/非运算

0b

q)not 0b　/非运算

1b

q)not 1>2

1b

q)1b and 1b　/且，同　1b & 1b

1b

q)1b or 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)\l hello.q

脚本文件将加载到 q，执行结果：

"hello kdb+"

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:2020.03.03 2020.03.04 2020.03.05 2020.03.04 2020.03.05 2020.03.06;close:2992.9 3011.67 3071.68 11493.02 11711.37 11582.82)
q)t
sym      date       close
-------------------------
000001.SH 2020.03.03 2992.9
000001.SH 2020.03.04 3011.67
000001.SH 2020.03.05 3071.68
399001.SZ 2020.03.04 11493.02
399001.SZ 2020.03.05 11711.37
399001.SZ 2020.03.06 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/t set 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.csv 0: 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 where sym=`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 t where 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。“n xprev 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“by sym”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:min close,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:mins close 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)update mavgclose:mavg[2;close],mmaxclose:mmax[2;close],mminclose:mmin[2;close] by sym from 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。

第二章 数据类型

本章将对 q 语言的基本数据类型及类型转换进行介绍。

第一节 概述

q 中所有数据均由原子（atom）构成，原子是 q 中不可再分割的最基本的数据类型。q 的基本数据类型与传统编程语言类似（见表 2.1），同时还具有丰富的时间相关数据类型，便于时间序列运算。

表 2.1 q 与其他语言数据类型对比

q	SQL	Java	.Net
boolean	boolean	boolean	boolean
guid		UUID	GUID
byte	byte	byte	byte
short	smallint	short	int16
int	int	integer	int32
long	bigint	long	int64
real	real	float	single
float	float	double	double
char	char(1)	character	char
symbol	varchar	string	(string)
timestamp		timestamp	DateTime
month			
date	Date	Date	Date
datetime	timestamp	timestamp	DateTime
timespan		timespan	timespan
minute			
second			
time	time	time	timespan

q 中的数据类型有多种表示方式：类型符号（type symbol）、类型字符（type char）、类型后缀（trailing type indicator）、类型代码（type number）。其中，类型符号是由反引号`和类型名称构成的 symbol，类型字符为一个代表数据类型的大写字母，类型代码则为代表该数据类型的

短整型，类型后缀则是与类型字符相对应的小写字母，如下表所示：

表 2.2 q 的数据类型

分类	类型全称	字节	符号	字符	代码	后缀	示例
二进制数据	boolean	1	`boolean	B	1h	b	1b 代表 true，0b 代表 false
	guid	16	` guid	G	2h		
	byte	1	` byte	X	4h	x	0x10(以 0x 开头)
数值数据	short	2	`short	H	5h	h	1h -1h
	int	4	` int	I	6h	i	1i -1i
	long	8	` long	J	7h	j	1 -1 1j -1j
	real	4	` real	E	8h	e	-1e
	float	8	` float	F	9h	f	1.0f
文本数据	char	1	` char	C	10h		"a"
	symbol	*	`	S	11h		`a `abc
日期与时间数据	timestamp	8	`timestamp	P	12h		2000.01.01D00:00:00.000000001
	month	4	`month	M	13h		2000.01m
	date	4	`date	D	14h		2000.01.01
	datetime	4	`datetime	Z	15h		2000.01.01T00:00:00.001
	timespan	8	`timespan	N	16h		0D00:00:00.000000001
	minute	4	`minute	U	17h		00:01
	second	4	`second	V	18h		00:01:02
	time	4	`time	T	19h		00:01:02:034

我们可以借助 type 函数获取该数据的类型代码，原子型数据的类型代码为负数，简单数组（数组内容见第三章）的类型代码为正数。需要注意的是，q 语言是动态类型语言，变量的类型会随着赋值数据类型的变化而变化。

例：

q)a:73 /给变量 a 赋值 73

q)type a /a 是一个长整型数据，类型代码为-7h，type 函数用于显示类型代码

-7h

q)type 1 2 3　/ 1 2 3 是简单数组，类型代码为 7h

7h

q)a:"xyz"　/给变量 a 赋值 xyz

q)type a /重新赋值后，a 是一个 char 简单数组，类型代码为 10h

10h

第二节 数值数据

本节主要介绍整数与浮点数两种数值数据。

一、整数

整数数据类型包含长整型（long）、短整型（short）、整型（int）三种类型，一般使用 long 类型。

（一）long

在 q 语言 3.0 版本以后，一个整数数字默认的数据类型是长整型。长整型的大小为 8 字节，类型代码为 7h，我们一般会在数字后加上类型后缀 j，但由于长整型是默认数据类型所以后缀并非强制要求。

例：

```
q)type 1j
-7h
q)type 1
-7h
q)type 1 3 5   /长整型简单数组
7h
q)a:56 /给变量 a 赋值 56
q)type a /a 的类型代码为-7h，是一个长整型
-7h
```

（二）short 和 int

顾名思义，短整型（short）在三种整数类型中所占内存最小，为 2 字节，类型代码为 5h，

注意必须在数字后加上类型后缀 h。我们之前提到，q 语言中类型代码就是用短整型来表示的。因此，当需要进行数据类型转换时，短整型可以用来决定目标类型，在本章最后我们会详细介绍进行数据类型转换的多种方式。

例：

q)type 1h

-5h

q)7h$2i /int 转换为 long

2

q)9h$2 /long 转换为 float

2f

整型（int）大小介于长整型和短整型之间，为 4 字节，类型后缀为 i。需要注意的是，在 q 语言 2.8 及之前的版本中，整数默认的类型为 int，由于 q 语言的默认数据类型在不同版本间会产生变化，为保证兼容性，建议尽量明确定义数据类型。

例：

q)type -1i

-6h

需要注意的是，q 语言在算数运算中会自动提升类型，例如布尔值可以参与数值计算，但在具有相同“宽”类型（字节数大）的数组中更新或添加“窄”类型（字节数小）的元素时，“窄”类型数据无法进行类型提升而导致出错。

例：

q)type 1j+2i

-7h

q)L:1 2 3f

q)L[2]:4j

‘type

二、浮点数

浮点数分为双精度（float）和单精度（real）两种类型。一般使用 float 类型。

（一）float

双精度浮点数（float）相当于其他编程语言中的 double，大小为 8 字节，类型后缀为 f，类型代码为 9h。对于小数点后为 0 的情况，输入后会自动省略 0 并加上后缀 f。此外，float 也可以用科学记数法表示，注意不要将“e”与类型后缀混淆，e 后面数字前的正号与零可省略。例：

```
q)type 111.222
-9h
q)type 1.2 2.3 3.4    /float 数组
9h
q)2.0
2f
q)type 1e9
-9h
q)1e-9
1e-009
```

（二） real

单精度浮点数（real）大小为 4 字节，类型后缀为 e，类型代码为 8h。real 的精度至少可以保留 6 位小数， float 的精度至少可以保留 15 位小数，因此在金融计算中，推荐使用精度较高的 float。

（三） 浮点数显示

需要注意得是，q 控制台默认显示 7 位小数精度的数字。因此，我们可能无法显示太长的浮点数。要想显示全部，可以利用\P（P 为大写字母）来控制显示精度。\P 后面的数字表示希望显示多少位数字，不包括小数点，但包括小数点前的数字。例：

```
q) 3.1415926535
3.141593
```

q) \P 11

q) 3.1415926535

3.1415926535

第三节 文本数据

本节主要介绍两种文本数据类型。q 语言有字符（char）和符号（symbol）两种原子文本类型，分别类似 SQL 中的 CHAR 和 VARCHAR。

一、char

字符类型（char）用 1 字节来储存一个单独的 ASCII 码，对应于 SQL 中的 CHAR，用双引号表示。在 q 语言中，双引号（double-quote）、反斜线（back-slash）等特殊字符不能直接放在双引号中成为字符，需要在其前使用转义符（\）进行转义，尽管在 q 控制台中依旧将\显示了出来，但是实际上仍为单字符。此外，由字符组成的数组称为字符串（string）。例：

```
q)type "x"
-10h
q)type "xyz""x"
-10h
q)"\"" / 双引号
"\""
q)"\\" / 反斜杠
"\\"
q)"\n" / 换行
"\n"
q)"\r" / 回车
"\r"
q)"\t" / tab 制表符
"\t"
q)"\142" /显示 ASCII 码对应的字母，142 为字母 b 的 ASCII 码八进制表示
"b"
```

二、symbol

符号（symbol）数据类型用反引号（`）表示，我们在第一节提到的类型符号就是 symbol 类型，所以都带有反引号（如`long，`char)。q 中所有名称均为符号，但并非所有符号均为名称。符号类似于 SQL 中的 VARCHAR，可以容纳任意数量的字符，不同之处在于 symbol 与 char 一样，是不可再拆分的原子类型，这意味着不可直接访问组成该符号的各个字符。需要注意的是，symbol 不同于 string，string 不是原子类型，而是字符构成的数组。并且 symbol 数据`a 与 char 数据"a"不相等。例：

```
q)type `z
-11h
q)type `x`y`z
11h
```

第四节 二进制数据

本节主要介绍布尔值（boolean）、字节（byte）与全局唯一标识符（GUID）三种二进制数据。

一、boolean

布尔值类型（boolean）用 1 字节来存储一个二进制位（bit，0 或 1），注意需要加上类型后缀 b。布尔值可参与计算并自动进行类型提升，即运算结果的类型为与其进行运算的数据类型相同，这个特性使得条件可以参与运算。

例：

```
q) type 0b
-1h
q) type 1b
-1h
q)type 10b      /boolean 数组，如 11b,101b,111b 等
1h
```

q) a: 1+0b

q) type a

-7h

q) flag:1b

q) base:10

q) base+flag*6

16

二、byte

字节（byte）数据类型大小为 1 字节，以 0x 为前缀，后面是两个 16 进制数字，可以使用大写或小写字母作为字母的十六进制数字，但通常使用小写字母。类型代码为 4h。与布尔值数据一样，byte 也可参与计算并自动进行类型提升。例：

q) type 0x2a

-4h

q) type 0x2A

-4h

q) type 0x2A3B4C /字节数组

4h

q) 1+0x2a

42

三、GUID

全局唯一标识符（GUID）是在 q 语言 3.0 版本中引入的。GUID 的全称是 globally unique identifier，是一个 16 字节的二进制值，该值在时间和空间上几乎是唯一的。GUID 特别适合在无需借助中央控制机制（例如交易 ID）的情况下，在本地生成全局唯一标识符。GUID 可以用作表的主键或在表连接中使用，在这种情况下，优于使用字符串或符号。

我们可以对 0Ng（null guid）使用?来生成一个 guid 数组,其中正号使用同一个初始随机种子

生产数组，而负号使用的种子全部是随机的。

例：

q)1?0Ng

q)-1?0Ng

q)3?0Ng

q)-3? 0ng

q)0x0 sv 16?0xff　/使用命令 sv 从一个 16 个 byte 的数组构造 GUID

第五节 日期与时间数据

本节将重点介绍 kdb+最具有特色的日期与时间数据类型：date、time、datetime、timestamp、timespan、month、minute、second 等。

一、日期与时间数据类型

q 语言最大的优势是可以有效地处理时间序列数据，q 语言中主要日期与时间数据类型的格式和总结如下表，我们可以注意到这些数据都以不同形式转换为数字。接下来，我们将对每种类型进行一一介绍。

表 2.3 q 的时间数据类型

类型	格式（红色字符表示固定字符）	代表内涵
date	yyyy.mm.dd	从 2000 年 1 月 1 日以来的第几天
time	hh:mm:ss.uuu	从凌晨计数的毫秒数
timespan	hh:mm:ss.nnnnnnnnn	从凌晨计数的纳秒数
datetime	yyyy.mm.dd**T**hh:mm:ss.uuu	整数部分为 2000 年 1 月 1 日以来天数，小数部分为凌晨计数的毫秒数占比
timestamp	yyyy.mm.dd**D**hh:mm:ss.nnnnnnnnn	从 2000 年 1 月 1 日以来的纳秒数
month	yyyy.mm**m**	从 2000 年 1 月以来的月份数
minute	hh:mm	从凌晨计数的分钟数，小时数会转换成分钟数
second	hh:mm:ss	从凌晨计数的秒数，时分都会转换成秒数

（一） date

日期（date）大小为 4 字节，必须写成 yyyy.mm.dd 的形式。例如，2019 年 7 月 4 日要用 2019.07.04 来表示，而 2019.7.4 则是错误的写法。日期（date）代表从 2000.01.01 计数的日期数，之后的为正值，之前的则为负值。

例：

q)2000.01.01=0 /判断 2000.01.01 是否等于 0

1b /结果为真

q)`int$2000.02.01 /表示的累积日数可以通过强制转换得到

（二） time 与 timespan

时间数据类型分为 time 和 timespan 两种。其中 time 精确到毫秒，表示为 hh:mm:ss.uuu 的形式（hh 代表小时，mm 代表分钟，ss 代表秒，uuu 代表毫秒），代表从凌晨（00:00）计算的毫秒数。与 date 一样，time 表示值同样可以通过强制转换来获取。如果毫秒级够用的话，我们就使用 time 数据类型。例：

q)12:34:56.789

12:34:56.789

q)12:00:00.000=12*60*60*1000

1b

若毫秒级不够用，可使用 timespan 类型。该类型将凌晨（00:00）后的纳秒数(nanoseconds)存储为长整数，表示为 0Dhh:mm:ss.nnnnnnnnn。其中，开头的 0D 是可选的，可以不写。例：

q)12:34:56.123456789

0D12:34:56.123456789

q)12:34:56.123456 / 微秒自动转化为纳秒

0D12:34:56.123456000

（三） datetime 与 timestamp

日期与时间数据类型包含 datetime 和 timestamp 两种。其中，datetime 类型已过时，不推荐使用，该类型用大写字母 T 来分隔日期与时间，表示为 yyyy.mm.ddThh:mm:ss.uuu。例：

q)2000.01.01T12:00:00.000

_

q)2000.01.02T12:00:00.000=1.5

1b

推荐使用的日期与时间类型是 timestamp 类型（时间戳）,也就是 date 类型和 timespan 类型的组合，通过大写字母 D 分隔日期与时间。timestamp 表示从 2000.01.01 以来的纳秒数，与 date 数据类型一样，2000.01.01 之后的为正值，之前的则为负值。例：

q)2014.11.22D17:43:40.123456789

q)`long$2014.11.22D17:43:40.123456789

q)`date$2014.11.22D17:43:40.123456789

q)`timespan$2014.11.22D17:43:40.123456789

q)2014.11.22+17:43:40.123　　/date + time 将得到 timestamp 类型

2014.11.22D17:43:40.123000000

（四） month

月份（month）数据类型存储为 32 位带符号的整型，表示为 yyyy.mm，必须加上类型后缀 m, 表示从 2000.01.01 计数的月份数。例：

q)2000.01m=0 /判断 2000.01m 是否等于 0

1b /结果为真

（五） minute

分钟（minute）数据类型存储为 32 位带符号的整型， 表示为 hh:mm， 代表从凌晨（00:00）开始计数的分钟数。例：

q)00:00=0

1b

q)12:00=12*60

1b

（六） second

秒（second）数据类型也是 32 位带符号整型， 表示为 hh:mm:ss，代表从凌晨（00:00）计数的秒数。例：

q)00:00:00=0

1b

q)23:59:59=-1+24*60*60

1b

二、点操作符与强制转换

我们可以通过点操作符提取年份、月份、日期等。但是更推荐使用强制转换符$，因为它适用于所有有意义的时间数据的提取和转换，且点运算符在函数内部不起作用。例：

q)d:2014.01.01

q)d.year

2014i

q)d.mm

q)d.dd

q)ti:12:34:56.789

q)ti.hh

12i

q)ti.mm

q)ti.ss

q)`dd$d　/强制转换

1i

q)`mm$d

q)`dd$d

q)`month$d

2014.01m

第六节 空值与最值

本节将介绍 q 中的空值（Nulls）与最值（Infinities）。

一、空值

空值（nulls）一般表示缺失数据。在 C，java 等语言中，空值是未分配内存空间的指针，在 q 中，空值与正常值同样占用内存，这是与其他编程语言不同的地方。

（一） 空值的类型

表 2.4 不同类型空值

类型	空值
boolean	0b
guid	0Ng (00000000-0000-0000-0000-000000000000)
byte	0x00
short	0Nh
Int	0Ni
long	0N 或 0Nj
real	0Ne
float	0n 或 0Nf
char	" "
sym	`
timestamp	0Np
month	0Nm
date	0Nd
datetime	0Nz
timespan	0Nn

minute	0Nu
second	0Nv
time	0Nt

（二） 空值的判断

q 使用 null 函数来测试一个值是不是 null 值，而 null 函数支持各种数据类型的。

例：

q)null 20

0b

q)null `

1b

q)null " "

1b

q)null ""

二、无穷

表 2.5 不同类型无穷值

记号	含义	无穷代表数值
0w	浮点数正无穷 Positive float infinity	
-0w	浮点数负无穷 Negative float infinity	
0W	长整型正无穷 Positive long infinity	+9223372036854775807
-0W	长整型负无穷 Negative long infinity	-9223372036854775807
0N	整型空值	-9223372036854775808
0n	浮点数空值、NaN、不是一个数字	

无穷值比一般的值都大。表 2.5 分别列出了长整型空值、正无穷和负无穷对应的实际数值，我们可以看出：0N < -0W < 正常整数< 0W。需要注意的是，小写的 w 代表 float, 大写的 W 代表长整数。整型无穷可以参与比较并返回正确的结果。q 的哲学是任何合法的数学表达式都会得到结果而非返回错误。在数学中，0 除以 0 是未定义的，但在 q 语言中，数字的除法结果总是 float，

正数除以 0 的结果为正无穷，负数除以 0 的结果为负无穷。

第七节 数据类型转换

本节介绍兼容类型转换、字符串解析等数据类型转换方法。

一、数据类型转换

类型转换（Casting）是将数据由一种类型转换为另一种兼容的数据类型，转换过程中信息可能保留也可能有所丢失。使用二元运算符“$”进行转换。操作符右边的参数是原始值，左边的参数是目标类型。我们有三种方式来指定目标类型：类型符号（type symbol）、类型字符（type char）、类型代码（type number），具体可见本章第一节。例：

```
q)`long$2i /通过类型符号转换
2
q)"j"$2i /通过类型简称转换，注意数值数据之间的类型转换要用小写字母
2
q)7h$2i /通过类型代码转换
2
```

（一）短字节转长字节

由第一节我们可以知道，不同数据类型所占字节大小不同，如果由短字节（窄字节）数据类型转换为长字节（宽字节）数据类型，一般不会造成信息丢失。例：

```
q)7h$2i /int 转换为 long
2
q)9h$2 /long 转换为 float
2f
q)“j”$2i /int 转换为 long
2
q)“f”$2 /long 转换为 float
```

2f

q)`long$2i /int 转换为 long

2

q)`float$2 /long 转换为 float

2f

（二）长字节转短字节

与之对应，如果由长字节数据类型转换为短字节数据类型，则可能造成信息丢失。

1、长字节数值转化为短字节数值

例：

q)`long$3.14159 /由 float 转换为 long 造成信息丢失

3

q)`short$123456789 /由 long 转换为 short 造成信息丢失

32767h

2、数值数据转换为 boolean

将数值类型转化为布尔值时，为 0 的数值转化为 0b，其他数值转化为 1b，例：

q)`boolean$0

0b

q)`boolean$123

1b

q)`boolean$-12.345

1b

3、复杂类型中提取成分

例：

q)`date$2015.01.02D10:20:30.123456789

2015.01.02

q)`year$2015.01.02

2015i

q)`month$2015.01.02

2015.01m

q)`mm$2015.01.02

1i

q)`dd$2015.01.02

2i

q)`hh$10:20:30.123456789

10i

q)`minute$10:20:30.123456789

10:20

q)`uu$10:20:30.123456789

20i

q)`second$10:20:30.123456789

10:20:30

q)`ss$10:20:30.123456789

30i

（三）跨数据类型间转换

1、字符串与整数互转

在前面我们提到字符串（char）类型存储的是 ASCII 码，所以只要整数不超过 256，我们就可以将 char 与整数互相转换。

例：

q)`char$42 /ASCII 码 42 代表的符号是*号

"*"

q)`long$"\n" /符号\n 代表的 ASCII 是 10

10

2、日期与整数互转

例：

q)`date$0

2000.01.01

q)`long$2001.01.01

366

3、timespan 与长整数互转

例：

q)`long$12:34:56.123456789

45296123456789

q)`timespan$150000000000

0D00:02:30.000000000

（四）整数极值转换

例：

q)`int$0Wh

32767i

q)`int$-0Wh

-32767i

q)`long$0Wi

2147483647

q)`long$-0Wi

-2147483647

（五）强制类型转换

我们对简单数组赋值时需要严格匹配数组的数据类型，这种时候可以通过强制类型转换来实现。例：

q)L:10 20 30 40

q)L[1]:42h

'type

q)L,:43h

'type

q)L[1]:(type L)$42h

q)L,:(type L)$43h

（六）转换为原子型操作

cast 是原子型函数,可以应用到左右两侧的每一个元素。对于 q 语言中的基础数据类型，我们可以同时对多个数据进行类型转换。例：

q)"i"$10 30 50

10 30 50i

q)`float$(12j; 12i; 12j)

12 12 12f

q)`short`int`long$12

12h

12i

12

q)"ijf"$98.6

99i

99

98.6

q)"ijf"$10 30 50

10i

30

50f

二、数据和文本的转换

（一）数据转换为 string

我们之前在介绍 char 时提到过，q 语言的 string 其实是由 char 组成的数组。我们还可以使用 string 函数将任何 q 实体转换为适合控制台展示或储存的文本形式。需要注意的是，string 函数不是原子型函数，存在如下特点：（1）转换结果是一个 char 数组，不是单个的 char。（2）转换结果不会包含任何 q 类型标识，可能无法解析回原始值。（3）对 string 数据使用 string 函数可能不会得到我们想要的结果。

例：

```
q)string 12 /转换后的"12"不再是 12，而是包含"1"和"2"两个 char 元素的数组，即字符串
"12"
q)string 1 /转换后的 1 变成单元素数组 "1"，即元素为一个"1"字符的字符数组
,"1"
q)string 12i
"12"
q)a:3.0
q)string a
,"3"
q)string 1 3 5
,"1"
,"3"
,"5"
q)string"test"    /“test”是一个 char 类型组成的数组
,"t"
,"e"
,"s"
,"t"
q)string (1 3 5; 20 40 60)
,"1" ,"3" ,"5"
```

"20" "40" "60"

q)string `Life`Is`Beautiful　/将 symbol 转换为 string

"Life"

"Is"

"Beautiful"

（二）string 转换为 symbol

我们可以使用 `$将 string 转换为 symbol，这也是创建带有空格或者其它特殊符号 symbol 的唯一方法。

例：

q)`$"xyz" /作为 string 类型，“ xyz”三个元素组成的数组

`xyz /作为 symbol 类型，xyz 是一个元素

q)`$"Hello World"

`Hello World

q)`$("Life";"Is";" Beautiful")

`Life`Is`Beautiful

q)`$"Zaphod \"Z\"" /注意带有转义符的转换

`Zaphod "Z"

q)string `$" xyz " /左右去空格

"xyz"

q)`$"中文"

`中文

q)`$"\326\320\316\304"　/中文 GBK 编码、8 进制

`中文

（三）　字符串解析

我们可以使用$将 string 转换为其他类型的数据，$右边是要解析的字符串，$左边是类型字符，代表希望转换成的目标数据类型。例：

q)"J"$"12"

12

q)"F"$"12"

12f

q)"F"$"12.0"

12f

q)"I"$"12.0"

0Ni

q)"I"$" "

0Ni

我们可以用这个方法将 string 日期信息转换为日期数据类型。例：

q)"D"$"12.31.2018"

2018.12.31

q)"D"$"12-31-2018"

2018.12.31

q)"D"$"12/31/2018"

2018.12.31

q)"D"$"12/31/2018"

2018.12.31

q)"D"$"2018/12/31"

2018.12.31

第三章至第十章内容见:

参考文献

[1]https://kx.com , https://code.kx.com

[2]http://itfin.f3322.org/opt/cgi/wiki.pl/KdbPlus

[3]Q for Mortals 3,Jeffry A. Borror , https://code.kx.com/q4m3/index.html

[4]q tips - Fast, Scalable, and Maintainable Kdb+, 2015,Nick Psaris,Vector Sigma

[5]Machine Learning and Big Data with kdb+,2020, Jan Novotny, Paul A. Bilokon, Aris Galiotos and Frédéric Déloze. Wiley Finance

[6]https://zhuanlan.zhihu.com/kdb2019

[7]http://www.aquaq.co.uk/

[8]http://www.timestored.com/

[9]http://www.devnet.de/

[10]http://www.tutorialspoint.com/kdbplus/index.htm

[11]https://www.meiwen.com.cn/c/evlpqttx.html

[12]http://www.vue5.com/kdbplus/kdbplus_overview.html