Kdb+ Cheat Sheet

Data Types

In v3.0+ the default number type is *long* and not *int*. The data type, *timestamp*, was introdcued as it is more accurate than the deprecated *datetime* type.

Type	Char	Num	Example	Null	•
boolean	b	1h	T|F → 1b|0b		
byte	x	4h	0x26	0x00	
short	h	5h	42h	0Nh	0Wh
int	i	6h	42i	0Ni	0W
long	j	7h	42j	0Nj	0Wj
real	e	8h	4.2e	0Ne	
float	f	9h	4.2	0n	0w
char	c	10h	“c”	“ “	
symbol	s	11h	`symbol	`	
timestamp	p	12h	2013.02.06D… 12:34:56.123 456789	0Np	
month	m	13h	2013.02m	0Nm	
date	d	14h	2013.02.06	0Nd	0Wd
datetime	z	15h	2013.02.06T… 12:34:56.123	0Nz	0Wz
minute	u	17h	12:34	0Nu	
second	v	18h	12:34:56	0Nv	
time	t	19h	12:34:56:123	0	0Wt
enumeration		*	`u$v		
dictionary		99h	`a`b`c!1 2 3		
table		98h	([]c1:`a`b`c ;c2:1 2 3)		

Basics

Alias – ::	a:1; b::a; c:a; a:2; b→2 c→1
Assign - :	a:42; a→42
Casting - **$**	5h$42.24 ⇔ ”h”$42.24 ⇔ `int$42.24 → 42i
Comment - /	/slash must be first char in line
Creating a **list**	(`a`b`c; 10 20; “qwe”)
Cross – **cross**	1 2 cross `a`b`c → ((1;`a); (1;`b); (1;`c);… (2;`a); (2;`b); (3;`c))
Cut - _	1 2_1 2 3!`a`b`c→3| c
Elide Indices - list[row;col;pos]	((1 2;3); (`a;`z`x))[1;1;0] → `z
Fill - **^**	12^1 2 0N 0N 3 4 → 1 2 12 12 3 4
Find - **?**	12 34 32?34 → 1
Function* – **f:{…}**	func:{[param$_1$; param$_n$] exp$_1$; exp$_n$}
Function call – **f[]**	func[param$_1$; param$_n$]
Indexing – **[]**	(`a`b`c;10 20)[1][0] → 10
Join - ,	1 2,3 4 → 1 2 3 4
Match - **~** (**tilde**)	(2*2; 1+1) ~ 4 2 → 1b
Max - **|**	123|321 → 321
Min - **&**	“asd”&”fgh” → ”agd”
Negate – **neg**	neg 1 2 3 → -1 -2 -3
Not equal - **<>**	1 2 3<>2 → 101b
Null – **null**	null 1 0N 2 3 0N → 01001b
Take - #	-2#1 2 3 → 2 3 5#1 2 3 → 1 2 3 1 2 (repeats)

* functions implicitly take parameters x,y,z

Common Functions

Unique – **distinct**	distinct `a`b`c`b`b`a`c → `a`b`c
Modulo – **mod**	4 mod 3 → 1
Power of – **xexp**	2 xexp 5 → 32f
Signature – **signum**	signum 42 -42 0 → 1 -1 0i

Advanced

Enumeration	v:`z`x`c`x`z`c; u:distinct v; k:u?v; u→`z`x`c k→0 1 2 1 0 2
Multi Projection	f:{x*y+z}; f[3;;2][5] → 21
IPC - `:server:port	h:hopen `:192.168.0.42:5001 h:hopen `::5001 (local) h“1+1” → 2 (synchronous) (neg h)”a:1” → (asynchronous) hclose h

Q-SQL

Define	t:([k1:`int$()] c1:`symbol$(); c2:`int$())
Insert	insert[`t;(`Sym;123;”a”)]

Foreign K	t1:([id:1 2] name:`Ab`Ti; height:180 185)
	t2:([] id:`t1$1 1 2 1 2; sc: 8 9 4 9 3)

Select	select CM:height from t where name=`Ab
By	select sc by id from t2
Exec	exec name from t1 → `Ab`Ti
Each	t:([]c1:1 2 3; c2:(“ab”; “cde”; “qw”)) select from t where (“cde”~) each c2
Update	update c1:val1,c2:val2 from t where c1=...
Select[n]	select[-2] from t (selects last two)

Adverbs

Adverbs are entities that modify a verb of a function to produce a new verb or function whose behavior is derived from the original. For the examples, f is a diadic function, x is an atom or list and y is a list

each both – ‘ (single quote)	Good for joining columns or tables of the same count 1 2 3,’3 → (1 3;2 3;3 3)
each monadic - **each**	reverse each (1 2; `a`b`c; ”24”) → (2 1; `c`b`a; ”42”) note: different to reverse
each left - \:	f f\: y 3 4 5 6{x*y}\:5 → 15 20 25 30
each right - /:	x f/: y 5{x*y}/:3 4 5 6 → 15 20 25 30
over - /	2{2*x+y}/1 2 → 16
scan - \	2{2*x+y}\1 2 → 6 16
each previous – ‘:	0+’:1 2 3 4 → 1 3 5 7
cartesian product - ,/:\:	1 2 ,/:\: `a`b`c→ 1 `a 1 `b 1 `c 2 `a 2 `b 2 `c
,\:/:	1 2 ,\:/: `a`b`c→ 1 `a 2 `a 1 `b 2 `b 1 `c 2 `c (transpose of above) note: raze ,\:/: ⇔ cross
verb ‘at’ - @	Monadic protected evaluation. @[MonadicFunc;arg;ExprOnFail]
verb ‘dot’ - .	Multivalent protected evaluation. .[MultValantFunc;argsExprOnFail]

Joins

Joins combine data from two tables (or table and dict).
Keyed joins: equi, inner, left, plus, union and upsert
ASOF joins: asof and window

equi – **ej**	ej[`col1`col2; t1; t2]
inner – **ij**	t ij kt
left – **lj**	t1 lj t2
plus – **pj**	t1 pj t2 Adds the values of the common columns
union – **uj**	t1 uj t2 If t1 and t2 have matching key columns, then records in t2 update matching records in t1. Otherwise, t2 records are inserted.
upsert	table upsert newrecords If keyed, new matched records update, otherwise inserted.
asof	table asof table/dict Last key or column on the right must correspong to a time column on left.
aj	aj[`col1`col2; t1; t2] t1 need not be keyed, t2 must not have a key. Common columns must be the same type.
aj0	If the resulting time is actual time instead of boundary time, use aj0.
wj	wj[w; c; t; (q;(f0;c0); (f1;c1))] Window join over time.
wj1	If the join considers quotes arriving from the beginning of the interval, use wj1.

File Handling

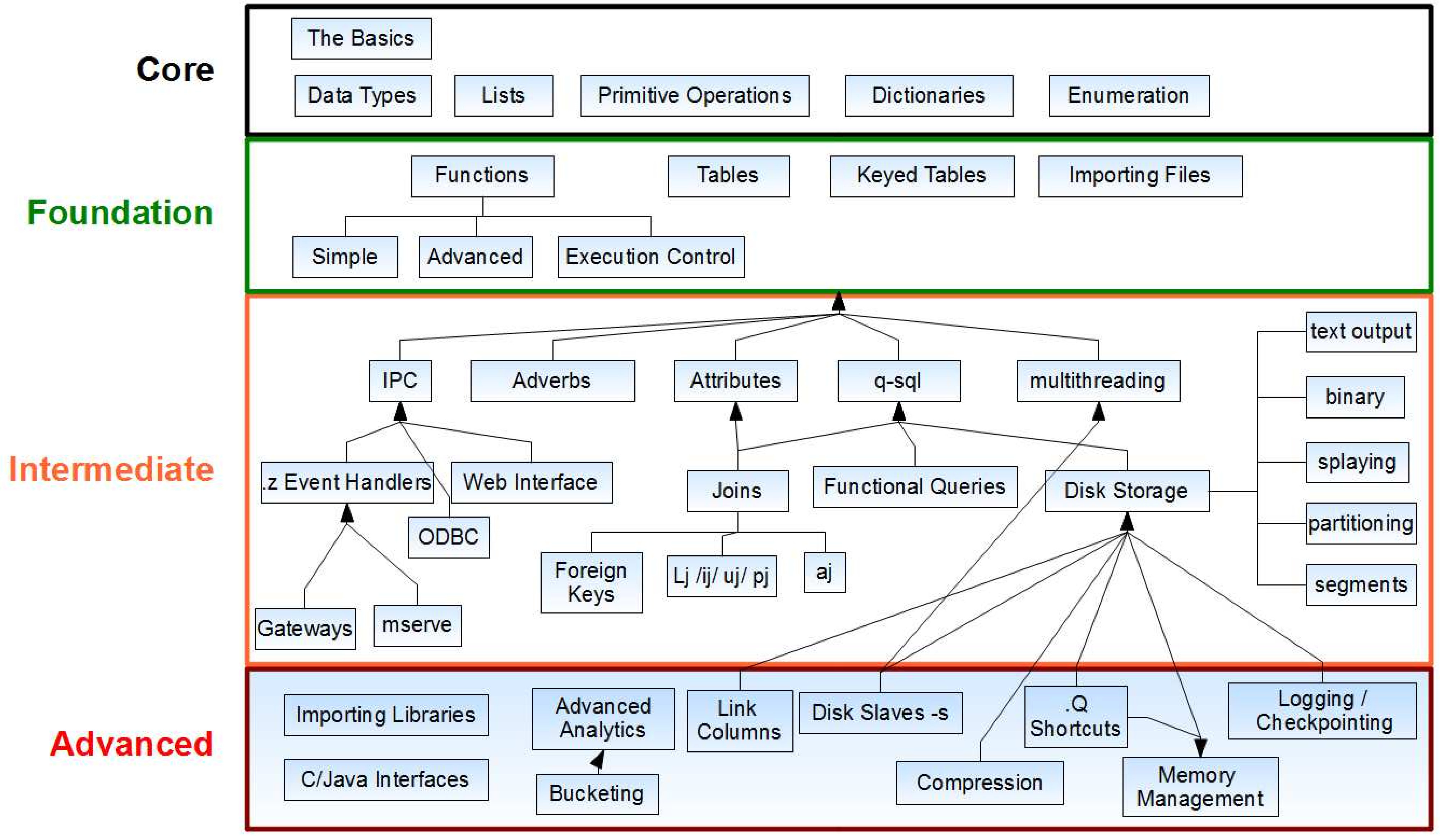
Saving a table	Saves the table t in the data file name.dat `:/q/folder/t.dat set t
Splaying a table	Saves the table t, splaying over a directory with each column as a columnName.dat file `:/q/folder/t/ set t
Saving a table in delimted formats^	save `:path/tname.(txt|csv|xml)
Load a table	load `:/path/table
Load splayed table	load `:/path/table/
Load a CSV to table	Without column names in CSV t:(“ISF”; “,”) 0: `:*.csv With column names t:(“ISF”; **enlist** “,”) 0: `:*.csv

System Variables

.z.d	date	**.z.P**	localtime	**.z.p**	GMT
.z.ts	timer	**\a**	tables	**\b**	views
\c	console size	**\d**	namespace	**\f**	functions
\P	precision	**\p**	port	**\v**	variables

TimeStored
http://www.timestored.com

Path to q God



We offer q/Kdb+ training, go to
http://www.timestored.com/kdbTraining

TimeStored