Tabularray	Typeset Tabulars and Arrays with LaTeX3
Author	Jianrui Lyu (tolvjr@163.com)
Version	2025C (2025-11-27)
Code	https://github.com/lvjr/tabularray
Code	https://bitbucket.org/lvjr/tabularray
Issue	https://github.com/lvjr/tabularray/issues
Discussion	https://github.com/lvjr/tabularray/discussions

```
\begin{tblr}{
  colspec = {rX}, colsep = 8mm, hlines = {2pt, white},
  row{odd} = {azure8}, row{even} = {gray8},
  row{1} = {6em,azure2,fg=white,font=\LARGE\bfseries\sffamily},
  row{2-Z} = {3em,font=\Large},
}
  Tabularray & Typeset Tabulars and Arrays with \LaTeX3 \\
  Author     & Jianrui Lyu (tolvjr@163.com) \\
  Version    & \myversion\ (\the\year-\mylpad\month-\mylpad\day) \\
  Code       & \url{https://github.com/lvjr/tabularray} \\
  Code       & \url{https://bitbucket.org/lvjr/tabularray} \\
  Issue      & \url{https://github.com/lvjr/tabularray/issues} \\
  Discussion & \url{https://github.com/lvjr/tabularray/discussions} \\
\end{tblr}
```

Contents

Chapter 1

Overview of Features

Before using `tabularray` package, it is better to know how to typeset simple text and math tables with traditional `tabular`, `tabularx` and `array` environments, because we will compare `tblr` environment from `tabularray` package with these environments. You may read web pages on LaTeX tables on LearnLaTeX and Overleaf first.

1.1 Vertical space

After loading `tabularray` package in the preamble, we can use `tblr` environments to typeset tabulars and arrays. The name `tblr` is short for `tabularray` or `top-bottom-left-right`. The following is our first example:

```
\begin{tabular}{lccr}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline
 Epsilon & Zeta  & Eta    & Theta \\
\hline
 Iota    & Kappa & Lambda & Mu    \\
\hline
\end{tabular}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

```
\begin{tblr}{lccr}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline
 Epsilon & Zeta  & Eta    & Theta \\
\hline
 Iota    & Kappa & Lambda & Mu    \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

You may notice that there is extra space above and below the table rows with `tblr` environment. This space makes the table look better. If you don't like it, you could use `\SetTblrInner` command:

```
\SetTblrInner{rowsep=0pt}
\begin{tblr}{lccr}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline
 Epsilon & Zeta  & Eta    & Theta \\
\hline
 Iota    & Kappa & Lambda & Mu    \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

But in many cases, this `rowsep` is useful:

```
$\begin{array}{rrr}
\hline
 \dfrac{2}{3} &  \dfrac{2}{3} &  \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} &  \dfrac{2}{3} \\
\hline
\end{array}$
```

$$\begin{array}{rrr}
\hline
 \dfrac{2}{3} & \dfrac{2}{3} & \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} & \dfrac{2}{3} \\
\hline
\end{array}$$

```
$\begin{tblr}{rrr}
\hline
 \dfrac{2}{3} &  \dfrac{2}{3} &  \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} &  \dfrac{2}{3} \\
\hline
\end{tblr}$
```

$$\begin{tblr}{rrr}
\hline
 \dfrac{2}{3} & \dfrac{2}{3} & \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} & \dfrac{2}{3} \\
\hline
\end{tblr}$$

Note that you can use `tblr` in both text and math modes.

1.2 Multiline cells

It's quite easy to write multiline cells without fixing the column width in `tblr` environments: just enclose the cell text with braces and use `\\` to break lines:

```
\begin{tblr}{|l|c|r|}
\hline
 Left & {Center \\ Cent \\ C} & {Right \\ R} \\
\hline
 {L \\ Left} & {C \\ Cent \\ Center} & R \\
\hline
\end{tblr}
```

Left	Center Cent C	Right R
L Left	C Cent Center	R

1.3 Cell alignment

From time to time, you may want to specify the horizontal and vertical alignment of cells at the same time. `Tabularray` package provides a `Q` column for this (In fact, `Q` column is the only primitive column, other columns are defined as `Q` columns with some options):

```
\begin{tblr}{|Q[l,t]|Q[c,m]|Q[r,b]|}
\hline
 {Top Baseline \\ Left Left} & Middle Center & {Right Right \\ Bottom Baseline} \\
\hline
\end{tblr}
```

Top Baseline Left Left	Middle Center	Right Right Bottom Baseline

Note that you can use more meaningful `t` instead of `p` for top baseline alignment. For some users who are familiar with word processors, these `t` and `b` columns are counter-intuitive. In `tabularray` package, there are another two column types `h` and `f`, which will align cell text at the head and the foot, respectively:

```
\begin{tblr}{Q[h,4em]Q[t,4em]Q[m,4em]Q[b,4em]Q[f,4em]}
\hline
 {row\\head} & {top\\line} & {middle} & {line\\bottom} & {row\\foot} \\
\hline
 {row\\head} & {top\\line} & {11\\22\\mid\\44\\55} & {line\\bottom} & {row\\foot} \\
\hline
\end{tblr}
```

row head	top line	middle	line bottom	row foot
row head	top line	11 22 mid 44 55	line bottom	row foot

1.4 Multirow cells

The above `h` and `f` alignments are necessary when we write multirow cells with `\SetCell` command in `tabularray`.

```
\begin{tabular}{|l|l|l|l|}
\hline
 \multirow[t]{4}{1.5cm}{Multirow Cell One} & Alpha &
 \multirow[b]{4}{1.5cm}{Multirow Cell Two} & Alpha \\
 & Beta  & & Beta \\
 & Gamma & & Gamma \\
 & Delta & & Delta \\
\hline
\end{tabular}
```

Multirow Cell One	Alpha Beta Gamma Delta	Multirow Cell Two	Alpha Beta Gamma Delta

```
\begin{tblr}{|l|l|l|l|}
\hline
 \SetCell[r=4]{h,1.5cm} Multirow Cell One & Alpha &
 \SetCell[r=4]{f,1.5cm} Multirow Cell Two & Alpha \\
 & Beta  & & Beta \\
 & Gamma & & Gamma \\
 & Delta & & Delta \\
\hline
\end{tblr}
```

Multirow Cell One	Alpha Beta Gamma Delta	Multirow Cell Two	Alpha Beta Gamma Delta

Note that you don't need to load `multirow` package first, since `tabularray` doesn't depend on it. Furthermore, `tabularray` will always typeset decent multirow cells. First, it will set correct vertical middle alignment, even though some rows have large height:

```
\begin{tabular}{|l|m{4em}|}
\hline
 \multirow[c]{4}{1.5cm}{Multirow} & Alpha  \\
 & Beta  \\
 & Gamma \\
 & Delta Delta Delta \\
\hline
\end{tabular}
```

Multirow	Alpha Beta Gamma Delta Delta Delta

```
\begin{tblr}{|l|m{4em}|}
\hline
 \SetCell[r=4]{m,1.5cm} Multirow & Alpha  \\
 & Beta  \\
 & Gamma \\
 & Delta Delta Delta \\
\hline
\end{tblr}
```

Multirow	Alpha Beta Gamma Delta Delta Delta

Second, it will enlarge row heights if the multirow cells have large height, therefore it always avoids vertical overflow:

```
\begin{tabular}{|l|m{4em}|}
\hline
 \multirow[c]{2}{1cm}{Line \\ Line \\ Line \\ Line} & Alpha \\
\cline{2-2}
 & Beta \\
\hline
\end{tabular}
```

Line Line	Alpha Beta
Line Line	

```
\begin{tblr}{|l|m{4em}|}
\hline
 \SetCell[r=2]{m,1cm} {Line \\ Line \\ Line \\ Line} & Alpha \\
\cline{2}
 & Beta \\
\hline
\end{tblr}
```

Line Line Line Line	Alpha
	Beta

If you want to distribute extra vertical space evenly to two rows, you may use `vspan` option described in Chapter 3.

1.5 Multi rows and columns

It was a hard job to typeset cells with multiple rows and multiple columns. For example:

```
\begin{tabular}{|c|c|c|c|c|}
\hline
 \multirow{2}{*}{2 Rows}
     & \multicolumn{2}{c|}{2 Columns}
                 & \multicolumn{2}{c|}{\multirow{2}{*}{2 Rows 2 Columns}} \\
\cline{2-3}
     & 2-2 & 2-3 & \multicolumn{2}{c|}{} \\
\hline
 3-1 & 3-2 & 3-3 & 3-4 & 3-5 \\
\hline
\end{tabular}
```

2 Rows	2 Columns		2 Rows 2 Columns	
	2-2	2-3		
3-1	3-2	3-3	3-4	3-5

With `tabularray` package, you can set spanned cells with `\SetCell` command: within the optional argument of `\SetCell` command, option `r` is for rowspan number, and `c` for colspan number; within the mandatory argument of it, horizontal and vertical alignment options are accepted. Therefore it's much simpler to typeset spanned cells:

```
\begin{tblr}{|c|c|c|c|c|}
\hline
 \SetCell[r=2]{c} 2 Rows
     & \SetCell[c=2]{c} 2 Columns
           &     & \SetCell[r=2,c=2]{c} 2 Rows 2 Columns & \\
\hline
     & 2-2 & 2-3 &     &     \\
\hline
 3-1 & 3-2 & 3-3 & 3-4 & 3-5 \\
\hline
\end{tblr}
```

2 Rows	2 Columns		2 Rows 2 Columns	
	2-2	2-3		
3-1	3-2	3-3	3-4	3-5

Using `\multicolumn` command, the omitted cells *must* be removed. On the contrary, using `\multirow` command, the omitted cells *must not* be removed. `\SetCell` command behaves the same as `\multirow` command in this aspect.

With `tblr` environment, any `\hline` segments inside a spanned cell will be ignored, therefore we're free to use `\hline` in the above example. Also, any omitted cell will definitely be ignored when typesetting, no matter it's empty or not. With this feature, we could put row and column numbers into the omitted cells, which will help us to locate cells when the tables are rather complex:

```
\begin{tblr}{|ll|c|rr|}
\hline
 \SetCell[r=3,c=2]{h} r=3 c=2 & 1-2 & \SetCell[r=2,c=3]{r} r=2 c=3 & 1-4 & 1-5 \\
 2-1 & 2-2 & 2-3 & 2-4 & 2-5 \\
\hline
 3-1 & 3-2 & MIDDLE & \SetCell[r=3,c=2]{f} r=3 c=2 & 3-5 \\
\hline
 \SetCell[r=2,c=3]{l} r=2 c=3 & 4-2 & 4-3 & 4-4 & 4-5 \\
 5-1 & 5-2 & 5-3 & 5-4 & 5-5 \\
\hline
\end{tblr}
```

r=3 c=2			r=2 c=3	
		MIDDLE		
r=2 c=3				r=3 c=2

1.6 Column types

`Tabularray` package supports all normal column types, as well as the extendable `X` column type, which first occurred in `tabularx` package and was largely improved by `tabu` package:

```
\begin{tblr}{|X[2,l]|X[3,l]|X[1,r]|X[r]|}
\hline
 Alpha & Beta & Gamma & Delta \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta

Also, `X` columns with negative coefficients are possible:

```
\begin{tblr}{|X[2,l]|X[3,l]|X[-1,r]|X[r]|}
\hline
 Alpha & Beta & Gamma & Delta \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta

We need the width to typeset a table with `X` columns. If unset, the default is `\linewidth`. To change the width, we have to first put all column specifications into `colspec={...}`:

```
\begin{tblr}{width=0.8\linewidth,colspec={|X[2,l]|X[3,l]|X[-1,r]|X[r]|}}
\hline
 Alpha & Beta & Gamma & Delta \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta

You can define new column types with `\NewTblrColumnType` command. For example, in `tabularray` package, `b` and `X` columns are defined as special `Q` columns:

```
\NewTblrColumnType{b}[1]{Q[b,wd=#1]}
\NewTblrColumnType{X}[1][]{Q[co=1,#1]}
```

1.7 Row types

Now that we have column types and `colspec` option, you may ask for row types and `rowspec` option. Yes, they are here:

```
\begin{tblr}{colspec={Q[l]Q[c]Q[r]},rowspec={|Q[t]|Q[m]|Q[b]|}}
 {Alpha \\ Alpha} & Beta              & Gamma \\
 Delta            & Epsilon           & {Zeta \\ Zeta}  \\
 Eta              & {Theta \\ Theta}  & Iota  \\
\end{tblr}
```

Alpha Alpha	Beta	Gamma
Delta	Epsilon	Zeta Zeta
Eta	Theta Theta	Iota

Same as column types, `Q` is the only primitive row type, and other row types are defined as `Q` types with different options. It's better to specify horizontal alignment in `colspec`, and vertical alignment in `rowspec`, respectively.

Inside `rowspec`, `|` is the hline type. Therefore we need not to write `\hline` command, which makes table code cleaner.

1.8 Hlines and vlines

Hlines and vlines have been improved too. You can specify the widths and styles of them:

```
\begin{tblr}{|l|[dotted]|[2pt]c|r|[solid]|[dashed]|}
\hline
One   &  Two  & Three \\
\hline\hline[dotted]\hline
Four  & Five  &   Six \\
\hline[dashed]\hline[1pt]
Seven & Eight &  Nine \\
\hline
\end{tblr}
```

One	Two	Three
Four	Five	Six
Seven	Eight	Nine

1.9 Colorful tables

To add colors to your tables, you need to load `xcolor` package first. `Tabularray` package will also load `ninecolors` package for proper color contrast. First you can specify background option for `Q` rows/columns inside `rowspec`/`colspec`:

```
\begin{tblr}{colspec={lcr},rowspec={|Q[cyan7]|Q[azure7]|Q[blue7]|}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{colspec={Q[l,brown7]Q[c,yellow7]Q[r,olive7]},rowspec={|Q|Q|Q|}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Also you can use `\SetRow` or `\SetColumn` command to specify row or column colors:

```
\begin{tblr}{colspec={lcr},rowspec={|Q|Q|Q|}}
 \SetRow{cyan7}  Alpha   & Beta  & Gamma  \\
 \SetRow{azure7} Epsilon & Zeta  & Eta    \\
 \SetRow{blue7}  Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{colspec={lcr},rowspec={|Q|Q|Q|}}
 \SetColumn{brown7}
 Alpha           & \SetColumn{yellow7}
                   Beta            & \SetColumn{olive7}
                                     Gamma  \\
 Epsilon         & Zeta            & Eta    \\
 Iota            & Kappa           & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Hlines and vlines can also have colors:

```
\begin{tblr}{colspec={lcr},rowspec={|[2pt,green7]Q|[teal7]Q|[green7]Q|[3pt,teal7]}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{colspec={|[2pt,violet5]l|[2pt,magenta5]c|[2pt,purple5]r|[2pt,red5]}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Chapter 2

Basic Interfaces

2.1 Old and new interfaces

With `tabularray` package, you can change the styles of tables via old interfaces or new interfaces.

The old interfaces consist of some table commands inside the table contents. Same as `tabular` and `array` environments, all table commands *must* be put at the beginning of the cell text. Also, new table commands *must* be defined with `\NewTblrTableCommand`.

The new interfaces consist of some options inside the mandatory argument, hence totally separating the styles and the contents of tables.

Table 2.1: Old Interfaces and New Interfaces

Old Interfaces	New Interfaces
`\SetHlines`	`hlines`
`\SetHline`, `\hline`, `\hborder`, `\cline`	`hline`, `hborder`, `rowspec`
`\SetVlines`	`vlines`
`\SetVline`, `\vline`, `\vborder`, `\rline`	`vline`, `vborder`, `colspec`
`\SetCells`	`cells`
`\SetCell`	`cell`
`\SetRows`	`rows`
`\SetRow`	`row`, `rowspec`
`\SetColumns`	`columns`
`\SetColumn`	`column`, `colspec`

2.2 Hlines and vlines

All available keys for hlines and vlines are described in Table 2.2 and Table 2.3.

Table 2.2: Keys for Hlines

Key	Description and Values	Initial Value
`dash`	dash style: `solid`, `dashed` or `dotted`	`solid`
`text`	replace hline with text (like `!` specifier in `rowspec`)	×
`wd`	rule width dimension	`0.4pt`

Continued on next page

Table 2.2: Keys for Hlines (Continued)

Key	Description and Values	Initial Value
fg	rule color name	×
leftpos	crossing or trimming position at the left side	1
rightpos	crossing or trimming position at the right side	1
l	same as leftpos, default -0.8	1
r	same as rightpos, default -0.8	1
lr	crossing or trimming positions at both sides, default -0.8	1
endpos	adjust leftpos/rightpos for only the leftmost/rightmost column	false

Note: In most cases, you can omit the underlined key names and write only their values.

Table 2.3: Keys for Vlines

Key	Description and Values	Initial Value
dash	dash style: solid, dashed or dotted	solid
text	replace vline with text (like ! specifier in colspec)	×
wd	rule width dimension	0.4pt
fg	rule color name	×
abovepos	crossing or trimming position at the above side	0
belowpos	crossing or trimming position at the below side	0

Note: In most cases, you can omit the underlined key names and write only their values.

2.2.1 Hlines and vlines in new interfaces

Options hlines and vlines are for setting all hlines and vlines, respectively. With empty value, all hlines/vlines will be solid.

```
\begin{tblr}{hlines,vlines}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

With values inside one pair of braces, all hlines/vlines will be styled.

```
\begin{tblr}{
 hlines = {1pt,solid}, vlines = {red3,dashed},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Another pair of braces before will select segments in all hlines/vlines.

```
\begin{tblr}{
 vlines = {1,3,5}{dashed},
 vlines = {2,4}{solid},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon

The above example can be simplified with `odd` and `even` values.

```
\begin{tblr}{
 vlines = {odd}{dashed},
 vlines = {even}{solid},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon

Another pair of braces before will draw more hlines/vlines (in which `-` stands for all line segments).

```
\begin{tblr}{
 hlines = {1}{-}{dashed}, hlines = {2}{-}{solid},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Note that you *must* use indexes in order: first 1, then 2, etc.

Options `hline{i}` and `vline{j}` are for setting some hlines and vlines, respectively. Their values are the same as options `hlines` and `vlines`:

```
\begin{tblr}{
 hline{1,7} = {1pt,solid},
 hline{3-5} = {blue3,dashed},
 vline{1,5} = {3-4}{dotted},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

You can use `U`, `V`, `W`, `X`, `Y`, `Z` to denote the last six children, respectively. It is especially useful when you are writing long tables:

```
\begin{tblr}{
 hline{1,Z} = {2pt},
 hline{2,Y} = {1pt},
 hline{3-X} = {dashed},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

Now we show the usage of `text` key by the following example[1]:

```
\begin{tblr}{
  vlines, hlines,
  colspec = {lX[c]X[c]X[c]X[c]},
  vline{2} = {1}{text=\clap{:}},
  vline{3} = {1}{text=\clap{$+$}},
  vline{4} = {1}{text=\clap{$\longrightarrow$}},
  vline{5} = {1}{text=\clap{$+$}},
}
  Equation & $CH_4$  & $2O_2$   & $CO_2$ & $2H_2O$ \\
  Initial  & $n_1$   & $n_2$    & $0$    & $0$ \\
  Final    & $n_1-x$ & $n_2-2x$ & $x$    & $2x$
\end{tblr}
```

Equation :	CH_4	+	$2O_2$	$\longrightarrow$	CO_2	+	$2H_2O$
Initial	n_1		n_2		0		0
Final	$n_1 - x$		$n_2 - 2x$		x		$2x$

The `leftpos` and `rightpos` keys specify crossing or trimming positions for hlines. The possible values for them are decimal numbers between `-1` and `1`. Their initial values are `1`.

-1	the hline is trimmed by `colsep`
0	the hline only touches the first vline
1	the hline touches all the vlines

The `abovepos` and `belowpos` keys for vlines have similar meanings. But their initial values are `0`.

-1	the vline is trimmed by `rowsep`
0	the vline only touches the first hline
1	the vline touches all the hlines

Here is an example for these four keys:

[1] Code from https://tex.stackexchange.com/questions/603023/tabularray-and-tabularx-column-separator.

```
\begin{tblr}{
  hline{1,4} = {1}{-}{},
  hline{1,4} = {2}{-}{},
  hline{2,3} = {1}{-}{leftpos = -1, rightpos = -1},
  hline{2,3} = {2}{-}{leftpos = -1, rightpos = -1},
  vline{1,4} = {abovepos = 1, belowpos = 1},
}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

There is also an `endpos` option for adjusting `leftpos`/`rightpos` for only the leftmost/rightmost column:

```
\begin{tblr}{
 hline{1,4} = {1}{-}{},
 hline{1,4} = {2}{-}{},
 hline{2,3} = {leftpos = -1, rightpos = -1, endpos},
 vline{1,4} = {abovepos = 1, belowpos = 1},
}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

2.2.2 Hlines and vlines in old interfaces

The `\hline` command has an optional argument which accepts key-value options. The available keys are described in Table 2.2.

```
\begin{tblr}{llll}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline[dashed]
 Epsilon & Zeta  & Eta    & Theta \\
\hline[dotted]
 Iota    & Kappa & Lambda & Mu    \\
\hline[2pt,blue5]
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

The `\cline` command also has an optional argument which is the same as `\hline`.

```
\begin{tblr}{llll}
\cline{1-4}
 Alpha   & Beta  & Gamma  & Delta \\
\cline[dashed]{1,3}
 Epsilon & Zeta  & Eta    & Theta \\
\cline[dashed]{2,4}
 Iota    & Kappa & Lambda & Mu    \\
\cline[2pt,blue5]{-}
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

You can use child selectors in the mandatory argument of `\cline`.

```
\begin{tblr}{llll}
\cline{1-4}
 Alpha   & Beta  & Gamma  & Delta \\
\cline[dashed]{odd}
 Epsilon & Zeta  & Eta    & Theta \\
\cline[dashed]{even}
 Iota    & Kappa & Lambda & Mu    \\
\cline[2pt,blue5]{-}
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Commands `\SetHline` combines the usages of `\hline` and `\cline`:

```
\begin{tblr}{llll}
\SetHline{1-3}{blue5,1pt}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\SetHline{2-4}{teal5,1pt}
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

```
\begin{tblr}{llll}
\SetHline[1]{1-3}{blue5,1pt}
\SetHline[2]{1-3}{azure5,1pt}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\SetHline[1]{2-4}{teal5,1pt}
\SetHline[2]{2-4}{green5,1pt}
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

In fact, table command `\SetHline[<index>]{<columns>}{<styles>}` at the beginning of row `i` is the same as table option `hline{i}={<index>}{<columns>}{<styles>}`.

Also, table command `\SetHlines[<index>]{<columns>}{<styles>}` at the beginning of some row is the same as table option `hlines={<index>}{<columns>}{<styles>}`.

The usages of table commands `\vline`, `\rline`, `\SetVline`, `\SetVlines` are similar to those of `\hline`, `\cline`, `\SetHline`, `\SetHlines`, respectively. But normally you don't need to use them.

2.3 Hborders and vborders

Options `hborder{i}` and `vborder{j}` are similar to `hline{i}` and `vline{j}`, respectively, but they hold border specifications not related to one specific hline and vline. All available keys for `hborder{i}` and `vborder{j}` are described in Table 2.4 and Table 2.5.

Table 2.4: Keys for Hborders

Key	Description and Values	Initial Value
`pagebreak`	pagebreak at this position: `yes`, `no` or `auto` (See Chapter 4)	`auto`
`abovespace`	set `belowsep` of previous row (see Table 2.8)	`2pt`
`belowspace`	set `abovesep` of current row (see Table 2.8)	`2pt`
`abovespace+`	increase `belowsep` of previous row	×
`belowspace+`	increase `abovesep` of current row	×

Table 2.5: Keys for Vborders

Key	Description and Values	Initial Value
leftspace	set rightsep of previous column (see Table 2.9)	6pt
rightspace	set leftsep of current column (see Table 2.9)	6pt
leftspace+	increase rightsep of previous column	×
rightspace+	increase leftsep of current column	×

Furthermore, table command \hborder{<specs>} at the beginning of row i is the same as table option hborder{i}={<specs>}, and table command \vborder{<specs>} at the beginning of column j is the same as table option vborder{j}={<specs>}.

2.4 Cells and spancells

All available keys for cells are described in Table 2.6 and Table 2.7.

Table 2.6: Keys for the Content of Cells

Key	Description and Values	Initial Value
halign	horizontal alignment: l (left), c (center), r (right) or j (justify)	j
valign	vertical alignment: t (top), m (middle), b (bottom), h (head) or f (foot)	t
wd	width dimension	×
bg	background color name	×
fg	foreground color name	×
font	font commands	×
mode	set cell mode: math, imath, dmath or text	×
$	same as mode=math	×
$$	same as mode=dmath	×
cmd	execute command for the cell text	×
preto	prepend text to the cell	×
appto	append text to the cell	×

Note: In most cases, you can omit the underlined key names and write only their values.

Table 2.7: Keys for Multispan of Cells

Key	Description and Values	Initial Value
r	number of rows the cell spans	1
c	number of columns the cell spans	1

2.4.1 Cells and spancells in new interfaces

Option cells is for setting all cells.

```
\begin{tblr}{hlines={white},cells={c,blue7}}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi

Option `cell{i}{j}` is for setting some cells, where `i` stands for the row numbers and `j` stands for the column numbers.

```
\begin{tblr}{
  cell{1}{2-4} = {cmd=\fbox}
}
  Alpha & Beta & Gamma & Delta
\end{tblr}
```

Alpha	Beta	Gamma	Delta

```
\begin{tblr}{
 hlines = {white},
 vlines = {white},
 cell{1,6}{odd} = {teal7},
 cell{1,6}{even} = {green7},
 cell{2,4}{1,4} = {red7},
 cell{3,5}{1,4} = {purple7},
 cell{2}{2} = {r=4,c=2}{c,azure7},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta		Theta
Iota			Mu
Nu			Pi
Rho			Upsilon
Phi	Chi	Psi	Omega

From version 2025A, you can select cells with a list of two dimensional indexes:

```
\begin{tblr}{
  cell{{2}{6},{7}{3}} = {bg=blue7},
  cell{{1}{1}-{4}{4},{5}{8}-{8}{5}} = {bg=red7}
}
  1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  3 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  4 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  5 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  6 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  7 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  8 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{tblr}
```

1	2	3	4	5	6	7	8
2	2	3	4	5	6	7	8
3	2	3	4	5	6	7	8
4	2	3	4	5	6	7	8
5	2	3	4	5	6	7	8
6	2	3	4	5	6	7	8
7	2	3	4	5	6	7	8
8	2	3	4	5	6	7	8

In this example, `-` characters are used for diagonal selection.

2.4.2 Cells and spancells in old interfaces

The `\SetCell` command has a mandatory argument for setting the styles of current cell. The available keys are described in Table 2.6.

```
\begin{tblr}{llll}
\hline[1pt]
 Alpha   & \SetCell{bg=teal2,fg=white} Beta & Gamma \\
\hline
 Epsilon & Zeta & \SetCell{r,font=\scshape} Eta \\
\hline
 Iota    & Kappa & Lambda \\
\hline[1pt]
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

The \SetCell command also has an optional argument for setting the multispan of current cell. The available keys are described in Table 2.7.

```
\begin{tblr}{|X|X|X|X|X|X|}
\hline
 Alpha & Beta & Gamma & Delta & Epsilon & Zeta \\
\hline
 \SetCell[c=2]{c} Eta & 2-2
              & \SetCell[c=2]{c} Iota & 2-4
                            & \SetCell[c=2]{c} Lambda  & 2-6 \\
\hline
 \SetCell[c=3]{c} Nu & 3-2 & 3-3
                      & \SetCell[c=3]{c} Pi & 3-5 & 3-6   \\
\hline
 \SetCell[c=6]{c} Tau & 4-2 & 4-3 & 4-4 & 4-5 & 4-6 \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta	Epsilon	Zeta
Eta		Iota		Lambda	
Nu			Pi		
Tau					

```
\begin{tblr}{|X|X|X|X|X|X|}
\hline
 Alpha & Beta    & Gamma   & Delta & Epsilon & Zeta \\
\hline
 \SetCell[r=2]{m} Eta
       & Theta   & Iota    & Kappa & Lambda  & \SetCell[r=2]{m} Mu  \\
\hline
 Nu    & Xi      & Omicron & Pi    & Rho     & Sigma \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta	Epsilon	Zeta
Eta	Theta	Iota	Kappa	Lambda	Mu
	Xi	Omicron	Pi	Rho	

In fact, table command \SetCell[]{<styles>} at the beginning of cell at row i and column j is the same as table option cell{i}{j}={}{<styles>}.

Also, table command \SetCells[]{<styles>} at the beginning of some cell is the same as table option cells={}{<styles>}.

2.5 Rows and columns

All available keys for rows and columns are described in Table 2.8 and Table 2.9.

Table 2.8: Keys for Rows

Key	Description and Values	Initial Value
halign	horizontal alignment: l (left), c (center), r (right) or j (justify)	j

Continued on next page

Table 2.8: Keys for Rows (Continued)

Key	Description and Values	Initial Value
valign	vertical alignment: `t` (top), `m` (middle), `b` (bottom), `h` (head) or `f` (foot)	`t`
ht	height dimension	×
bg	background color name	×
fg	foreground color name	×
font	font commands	×
mode	set mode for row cells: `math`, `imath`, `dmath` or `text`	×
$	same as `mode=math`	×
$$	same as `mode=dmath`	×
cmd	execute command for every cell text	×
abovesep	set vertical space above the row	2pt
abovesep+	increase vertical space above the row	×
belowsep	set vertical space below the row	2pt
belowsep+	increase vertical space below the row	×
rowsep	set vertical space above and below the row	2pt
rowsep+	increase vertical space above and below the row	×
preto	prepend text to every cell (like `>` specifier in `rowspec`)	×
appto	append text to every cell (like `<` specifier in `rowspec`)	×

Note: In most cases, you can omit the underlined key names and write only their values.

Table 2.9: Keys for Columns

Key	Description and Values	Initial Value
halign	horizontal alignment: `l` (left), `c` (center), `r` (right) or `j` (justify)	`j`
valign	vertical alignment: `t` (top), `m` (middle), `b` (bottom), `h` (head) or `f` (foot)	`t`
wd	width dimension	×
co	coefficient for the extendable column (`X` column)	×
bg	background color name	×
fg	foreground color name	×
font	font commands	×
mode	set mode for column cells: `math`, `imath`, `dmath` or `text`	×
$	same as `mode=math`	×
$$	same as `mode=dmath`	×
cmd	execute command for every cell text	×
leftsep	set horizontal space to the left of the column	6pt
leftsep+	increase horizontal space to the left of the column	×
rightsep	set horizontal space to the right of the column	6pt
rightsep+	increase horizontal space to the right of the column	×
colsep	set horizontal space to both sides of the column	6pt
colsep+	increase horizontal space to both sides of the column	×
preto	prepend text to every cell (like `>` specifier in `colspec`)	×

Continued on next page

Table 2.9: Keys for Columns (Continued)

Key	Description and Values	Initial Value
appto	append text to every cell (like < specifier in colspec)	×

Note: In most cases, you can omit the underlined key names and write only their values.

2.5.1 Rows and columns in new interfaces

Options rows and columns are for setting all rows and columns, respectively.

```
\begin{tblr}{
 hlines, vlines,
 rows = {7mm}, columns = {15mm,c},
}
 Alpha   & Beta  & Gamma   & Delta \\
 Epsilon & Zeta  & Eta     & Theta \\
 Iota    & Kappa & Lambda  & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Options row{i} and column{j} are for setting some rows and columns, respectively.

```
\begin{tblr}{
 hlines = {1pt,white},
 row{odd} = {blue7},
 row{even} = {azure7},
 column{1} = {purple7,c},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

The following example demonstrates the usages of bg, fg and font keys:

```
\begin{tblr}{
 row{odd} = {bg=azure8},
 row{1}   = {bg=azure3, fg=white, font=\sffamily},
}
 Alpha & Beta    & Gamma \\
 Delta & Epsilon & Zeta  \\
 Eta   & Theta   & Iota  \\
 Kappa & Lambda  & Mu    \\
 Nu Xi Omicron & Pi Rho Sigma & Tau Upsilon Phi \\
\end{tblr}
```

Alpha	Beta	Gamma
Delta	Epsilon	Zeta
Eta	Theta	Iota
Kappa	Lambda	Mu
Nu Xi Omicron	Pi Rho Sigma	Tau Upsilon Phi

The following example demonstrates the usages of `mode` key:

```
$\begin{tblr}{
  column{1} = {mode=text},
  column{3} = {mode=dmath},
}
\hline
  Alpha   & \frac12 & \frac12 \\
  Epsilon & \frac34 & \frac34 \\
  Iota    & \frac56 & \frac56 \\
\hline
\end{tblr}$
```

Alpha	$\frac{1}{2}$	$\frac{1}{2}$
Epsilon	$\frac{3}{4}$	$\frac{3}{4}$
Iota	$\frac{5}{6}$	$\frac{5}{6}$

Note that you *can not* write multiline math directly (such as `\alpha \\ \beta`) in any math-mode cell.

The following example demonstrates the usages of `abovesep`, `belowsep`, `leftsep`, `rightsep` keys:

```
\begin{tblr}{
 hlines, vlines,
 rows = {abovesep=1pt,belowsep=5pt},
 columns = {leftsep=1pt,rightsep=5pt},
}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

The following example shows that we can replace `\\[dimen]` with `belowsep+` key.

```
\begin{tblr}{
 hlines, row{2} = {belowsep+=5pt},
}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

2.5.2 Rows and columns in old interfaces

The `\SetRow` command has a mandatory argument for setting the styles of current row. The available keys are described in Table 2.8.

```
\begin{tblr}{llll}
\hline[1pt]
 \SetRow{azure8} Alpha & Beta & Gamma & Delta \\
\hline
 \SetRow{blue8,c} Epsilon & Zeta & Eta & Theta \\
\hline
 \SetRow{violet8} Iota & Kappa & Lambda & Mu \\
\hline[1pt]
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

In fact, table command `\SetRow{<styles>}` at the beginning of row `i` is the same as table option `row{i}={<styles>}`.

Also, table command `\SetRows{<styles>}` at the beginning of some row is the same as table option `rows={<styles>}`.

The usages of table commands `\SetColumn` and `\SetColumns` are similar to those of `\SetRow` and `\SetRows`, respectively. But normally you don't need to use them.

2.6 Colspec and rowspec

Options `colspec`/`rowspec` are for setting column/row specifications with column/row type specifiers.

2.6.1 Colspec and width

Option `width` is for setting the width of the table with extendable columns. The following example demonstrates the usage of `width` option.

```
\begin{tblr}{width=0.8\textwidth, colspec={|l|X[2]|X[3]|X[-1]|}}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

You can omit `colspec` name if it is the only key you use inside the mandatory argument. The following example demonstrates the usages of `$` and `$$` keys:

```
\begin{tblr}{Q[l]Q[r,$]Q[r,$$]}
\hline
  Alpha   & \frac12 & \frac12 \\
  Epsilon & \frac34 & \frac34 \\
  Iota    & \frac56 & \frac56 \\
\hline
\end{tblr}
```

Alpha	$\frac{1}{2}$	$\frac{1}{2}$
Epsilon	$\frac{3}{4}$	$\frac{3}{4}$
Iota	$\frac{5}{6}$	$\frac{5}{6}$

2.6.2 Column types

The `tabularray` package has only one type of primitive column: the `Q` column. Other types of columns are defined as `Q` columns with some keys.

```
\NewTblrColumnType{l}{Q[l]}
\NewTblrColumnType{c}{Q[c]}
\NewTblrColumnType{r}{Q[r]}
\NewTblrColumnType{t}[1]{Q[t,wd=#1]}
\NewTblrColumnType{m}[1]{Q[m,wd=#1]}
\NewTblrColumnType{b}[1]{Q[b,wd=#1]}
\NewTblrColumnType{h}[1]{Q[h,wd=#1]}
\NewTblrColumnType{f}[1]{Q[f,wd=#1]}
\NewTblrColumnType{X}[1][]{Q[co=1,#1]}
```

```
\begin{tblr}{|t{15mm}|m{15mm}|b{20mm}|}
 Alpha   & Beta  & {Gamma\\Gamma} \\
 Epsilon & Zeta  & {Eta\\Eta} \\
 Iota    & Kappa & {Lambda\\Lambda} \\
\end{tblr}
```

Alpha	Beta	Gamma Gamma
Epsilon	Zeta	Eta Eta
Iota	Kappa	Lambda Lambda

Any new column type must be defined with `\NewTblrColumnType` command. It can have an optional argument when it's defined.

2.6.3 Row types

The `tabularray` package has only one type of primitive row: the `Q` row. Other types of rows are defined as `Q` rows with some keys.

```
\NewTblrRowType{l}{Q[l]}
\NewTblrRowType{c}{Q[c]}
\NewTblrRowType{r}{Q[r]}
\NewTblrRowType{t}[1]{Q[t,ht=#1]}
\NewTblrRowType{m}[1]{Q[m,ht=#1]}
\NewTblrRowType{b}[1]{Q[b,ht=#1]}
\NewTblrRowType{h}[1]{Q[h,ht=#1]}
\NewTblrRowType{f}[1]{Q[f,ht=#1]}
```

```
\begin{tblr}{rowspec={|t{12mm}|m{10mm}|b{10mm}|}}
 Alpha   & Beta  & {Gamma\\Gamma} \\
 Epsilon & Zeta  & {Eta\\Eta} \\
 Iota    & Kappa & {Lambda\\Lambda} \\
\end{tblr}
```

Alpha	Beta	Gamma Gamma
Epsilon	Zeta	Eta Eta
Iota	Kappa	Lambda Lambda

Any new row type must be defined with `\NewTblrRowType` command. It can have an optional argument when it's defined.

Chapter 3

Extra Interfaces

In general, `tblr` environment accepts both inner and outer specifications:

```
\begin{tblr}[<outer specs>]{<inner specs>}
  <table body>
\end{tblr}
```

Inner specifications are all specifications written in the mandatory argument of `tblr` environment, which include new interfaces described in Chapter 2.

Outer specifications are all specifications written in the optional argument of `tblr` environment, most of which are used for long tables (see Chapter 4).

You can use `\SetTblrInner` and `\SetTblrOuter` commands to set default inner and outer specifications of tables, respectively (see Section 3.3).

3.1 Inner specifications

In addition to new interfaces in Chapter 2, there are several inner specifications which are described in Table 3.1.

Table 3.1: Keys for Inner Specifications

Key	Description and Values	Initial Value
`rulesep`	space between two hlines or vlines	`2pt`
`stretch`	stretch ratio for struts added to cell text	`1`
`abovesep`	set vertical space above every row	`2pt`
`belowsep`	set vertical space below every row	`2pt`
`rowsep`	set vertical space above and below every row	`2pt`
`leftsep`	set horizontal space to the left of every column	`6pt`
`rightsep`	set horizontal space to the right of every column	`6pt`
`colsep`	set horizontal space to both sides of every column	`6pt`
`hspan`	horizontal span algorithm: `default`, `even`, or `minimal`	`default`
`vspan`	vertical span algorithm: `default` or `even`	`default`
`baseline`	set the baseline of the table	`m`

3.1.1 Space between double rules

The following example shows that we can replace `\doublerulesep` parameter with `rulesep` key.

```
\begin{tblr}{
 colspec={||llll||},rowspec={|QQQ|},rulesep=4pt,
}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

3.1.2 Minimal strut for cell text

The following example shows that we can replace `\arraystretch` parameter with `stretch` key.

```
\begin{tblr}{hlines,stretch=1.5}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

By replacing stretch with row heights, we can get perfect vertical centering for your numerical tables.

```
\begin{tblr}{hlines, stretch=0, rows={ht=\baselineskip}}
  2021 & 2022 & 2023 \\
  0.4  & 0.5  & 0.6  \\
  1.1  & 2.2  & 3.3  \\
\end{tblr}
```

2021	2022	2023
0.4	0.5	0.6
1.1	2.2	3.3

3.1.3 Rowseps and colseps for all

The following example uses `rowsep` and `colsep` keys to set padding for all rows and columns.

```
\SetTblrInner{rowsep=2pt,colsep=2pt}
\begin{tblr}{hlines,vlines}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

3.1.4 Hspan and vspan algorithms

With `hspan=default` or `hspan=even`, `tabularray` package will compute column widths from span widths. But with `hspan=minimal`, it will compute span widths from column widths. The following examples show the results from different `hspan` values.

```
\SetTblrInner{hlines, vlines, hspan=default}
\begin{tblr}{cell{2}{1}={c=2}{l},cell{3}{1}={c=3}{l},cell{4}{2}={c=2}{l}}
 111 111 & 222 222 & 333 333 \\
 12 Multi Columns Multi Columns 12 & & 333 \\
 13 Multi Columns Multi Columns Multi Columns 13 & & \\
 111 & 23 Multi Columns Multi Columns 23 & \\
\end{tblr}
```

111 111	222 222	333 333
12 Multi Columns Multi Columns 12		333
13 Multi Columns Multi Columns Multi Columns 13		
111	23 Multi Columns Multi Columns 23	

```
\SetTblrInner{hlines, vlines, hspan=even}
\begin{tblr}{cell{2}{1}={c=2}{l},cell{3}{1}={c=3}{l},cell{4}{2}={c=2}{l}}
 111 111 & 222 222 & 333 333 \\
 12 Multi Columns Multi Columns 12 & & 333 \\
 13 Multi Columns Multi Columns Multi Columns 13 & & \\
 111 & 23 Multi Columns Multi Columns 23 & \\
\end{tblr}
```

111 111	222 222	333 333
12 Multi Columns Multi Columns 12		333
13 Multi Columns Multi Columns Multi Columns 13		
111	23 Multi Columns Multi Columns 23	

```
\SetTblrInner{hlines, vlines, hspan=minimal}
\begin{tblr}{cell{2}{1}={c=2}{l},cell{3}{1}={c=3}{l},cell{4}{2}={c=2}{l}}
 111 111 & 222 222 & 333 333 \\
 12 Multi Columns Multi Columns 12 & & 333 \\
 13 Multi Columns Multi Columns Multi Columns 13 & & \\
 111 & 23 Multi Columns Multi Columns 23 & \\
\end{tblr}
```

111 111	222 222	333 333
12 Multi Columns Multi Columns 12		333
13 Multi Columns Multi Columns Multi Columns 13		
111	23 Multi Columns Multi Columns 23	

The following examples show the results from different `vspan` values.

```
\SetTblrInner{hlines, vlines, vspan=default}
\begin{tblr}{column{2}={3.25cm}, cell{2}{2}={r=3}{l}}
  Column1 & Column2 \\
  Row1 & Long text that needs multiple lines.
         Long text that needs multiple lines.
         Long text that needs multiple lines. \\
  Row2 & \\
  Row3 & \\
  Row4 & Short text \\
\end{tblr}
```

Column1	Column2
Row1	Long text that needs multiple lines. Long text that needs multiple lines. Long text that needs multiple lines.
Row2	
Row3	
Row4	Short text

```
\SetTblrInner{hlines, vlines, vspan=even}
\begin{tblr}{column{2}={3.25cm}, cell{2}{2}={r=3}{l}}
  Column1 & Column2 \\
  Row1 & Long text that needs multiple lines.
         Long text that needs multiple lines.
         Long text that needs multiple lines. \\
  Row2 & \\
  Row3 & \\
  Row4 & Short text \\
\end{tblr}
```

Column1	Column2
Row1	Long text that needs multiple lines. Long text that needs multiple lines. Long text that needs multiple lines.
Row2	
Row3	
Row4	Short text

3.1.5 Set baseline for the table

With `baseline` key, you can set baseline for the table. All possible values for `baseline` are as follows:

`t`	align the table at the top
`T`	align the table at the first row
`m`	align the table at the middle, initial value
`b`	align the table at the bottom
`B`	align the table at the last row
`<n>`	align the table at row `<n>` (a positive integer)

If there is no hline above the first row, you get the same result with either `t` or `T`. But you get different results if there are one or more hlines above the row:

```
Baseline\begin{tblr}{hlines,baseline=t}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}Baseline
```

Baseline

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Baseline

```
Baseline\begin{tblr}{hlines,baseline=T}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}Baseline
```

Baseline

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Baseline

The differences between `b` and `B` are similar to `t` and `T`. In fact, these two values `T` and `B` are better replacements for currently obsolete `\firsthline` and `\lasthline` commands.

3.2 Outer specifications

Except for specifications to be introduced in Chapter 4, there are several other outer specifications which are described in Table 3.2.

Table 3.2: Keys for Outer Specifications

Key	Description and Values	Initial Value
`baseline`	set the baseline of the table	`m`
`long`	change the table to a long table	×
`tall`	change the table to a tall table	×
`expand`	you need this key to use verb commands	×
`expand+`	like `expand` but appends to previous values	×

3.2.1 Set baseline in another way

You may notice that you can write `baseline` option as either an inner or an outer specification. It is true that either way would do the job. But there is a small difference: when `baseline=t/T/m/b/B` is an outer specification, you can omit the key name and write the value only.

```
Baseline\begin{tblr}[m]{hlines}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}Baseline
```

Baseline

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Baseline

3.2.2 Long and tall tables

You can change a table to long table by passing outer specification `long`, or change it to tall table by passing outer specification `tall` (see Chapter 4). Therefore the following two tables are the same:

```
\begin{longtblr}{lcr}
  Alpha & Beta & Gamma
\end{longtblr}
\begin{tblr}[long]{lcr}
  Alpha & Beta & Gamma
\end{tblr}
```

3.2.3 Expand macros first

In contrast to traditional `tabular` environment, `tabularray` environments need to see every `&` and `\\` when splitting the table body. And you can not put cell text inside any table command defined with `\NewTblrTableCommand`. But you could use outer key `expand` to make `tabularray` expand every occurrence of any of the specified macros *once* and in the given order before splitting the table body. Note that you *can not* expand a command defined with `\NewDocumentCommand`. You can also use `expand+` if you still want to keep the macros in the current `expand` setting.

To expand a command without optional argument, you can define it with `\newcommand`.

```
\newcommand*\tblrrowa{
  20 & 30 & 40 \\
}
\newcommand*\tblrrowb{
  50 & 60 & 70 \\
}
\newcommand*\tblrbody{
 \hline
  \tblrrowa
  \tblrrowb
 \hline
}
\SetTblrOuter{expand=\tblrbody\tblrrowa}
\begin{tblr}[expand+=\tblrrowb]{ccc}
 \hline
  AA & BB & CC \\
  \tblrbody
  DD & EE & FF \\
  \tblrbody
  GG & HH & II \\
 \hline
\end{tblr}
```

AA	BB	CC
20	30	40
50	60	70
DD	EE	FF
20	30	40
50	60	70
GG	HH	II

To expand commands with optional arguments, you *can not* define them with `\newcommand`. But you can define them with `\NewExpandableDocumentCommand`, and use option `expand=\expanded` to do exhaustive expansions.

```
\NewExpandableDocumentCommand\yes{O{Yes}m}{\SetCell{bg=green9}#1}
\NewExpandableDocumentCommand\no{O{No}m}{\SetCell{bg=red9}#1}
\begin{tblr}[expand=\expanded]{hlines}
  What I get                & is below              \\
  \expanded{\yes{}}         & \expanded{\no{}}      \\
  \expanded{\yes[Great]{}} & \expanded{\no[Bad]{}}
\end{tblr}
```

What I get	is below
Yes	No
Great	Bad

Note that you need to protect fragile commands (if any) inside them with `\unexpanded` command.

3.3 Default specifications

`Tabularray` package provides `\SetTblrInner` and `\SetTblrOuter` commands for you to change the default inner and outer specifications of tables.

In general different `tabularray` environments (`tblr`, `talltblr`, `longtblr`, etc) could have different default specifications. You can list the environments in the optional arguments of these two commands, and they only apply to `tblr` environment when the optional arguments are omitted.

In the following example, the first line draws all hlines and vlines for all `tblr` tables created afterwards, while the second line makes all `tblr` tables created afterwards vertically align at the last row.

```
\SetTblrInner{hlines,vlines}
\SetTblrOuter{baseline=B}
```

And the following example sets zero `rowsep` for all `tblr` and `longtblr` tables created afterwards.

```
\SetTblrInner[tblr,longtblr]{rowsep=0pt}
```

3.4 New tabularray environments

You can define new `tabularray` environments using `\NewTblrEnviron` command:

```
\NewTblrEnviron{mytblr}
\SetTblrInner[mytblr]{hlines,vlines}
\SetTblrOuter[mytblr]{baseline=B}
Text \begin{mytblr}{cccc}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{mytblr} Text
```

Text

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Text

3.5 New general environments

With `+b` argument type of `\NewDocumentEnvironment` command, you can also define a new general environment based on `tblr` environment (note that there is an extra pair of curly braces at the end):

```
\NewDocumentEnvironment{fancytblr}{+b}{
  Before Text
  \begin{tblr}{hlines}
    #1
  \end{tblr}
  After Text
}{}
```

```
\begin{fancytblr}
  One   & Two   & Three \\
  Four  & Five  & Six   \\
  Seven & Eight & Nine  \\
\end{fancytblr}
```

Before Text

One	Two	Three
Four	Five	Six
Seven	Eight	Nine

After Text

3.6 New table commands

All commands which change the specifications of tables *must* be defined with `\NewTblrTableCommand`. The following example demonstrates how to define a new table command:

```
\NewTblrTableCommand\myhline{\hline[0.1em,red5]}
\begin{tblr}{llll}
\myhline
 Alpha   & Beta  & Gamma   & Delta \\
 Epsilon & Zeta  & Eta     & Theta \\
 Iota    & Kappa & Lambda  & Mu    \\
\myhline
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

3.7 Child indexers and selectors

From version 2025A, child indexer `Z` accepts an optional argument for making a negative index.

```
\begin{tblr}{
  cell{1}{2-Z[2]} = {red9},
  cell{2}{3-Z[3]} = {green9},
  cell{3}{Z[6]-Z[4]} = {blue9}
}
  A & B & C & D & E & F & G & H & I \\
  A & B & C & D & E & F & G & H & I \\
  A & B & C & D & E & F & G & H & I
\end{tblr}
```

A	B	C	D	E	F	G	H	I
A	B	C	D	E	F	G	H	I
A	B	C	D	E	F	G	H	I

From version 2022A, child selectors `odd` and `even` accept an optional argument, in which you can specify the start index of the children.

```
\begin{tblr}{
  cell{1}{odd} = {yellow9},
  cell{2}{odd[5]} = {purple9},
  cell{3}{odd[4]} = {cyan9}
}
  A & B & C & D & E & F & G & H & I & J \\
  A & B & C & D & E & F & G & H & I & J \\
  A & B & C & D & E & F & G & H & I & J
\end{tblr}
```

A	B	C	D	E	F	G	H	I	J
A	B	C	D	E	F	G	H	I	J
A	B	C	D	E	F	G	H	I	J

```
\begin{tblr}{
  cell{1}{even} = {red9},
  cell{2}{even[4]} = {green9},
  cell{3}{even[3]} = {blue9}
}
  A & B & C & D & E & F & G & H & I & J \\
  A & B & C & D & E & F & G & H & I & J \\
  A & B & C & D & E & F & G & H & I & J \\
\end{tblr}
```

A	B	C	D	E	F	G	H	I	J
A	B	C	D	E	F	G	H	I	J
A	B	C	D	E	F	G	H	I	J

From version 2025A, there is a new child selector `every` for selecting indexes in an arithmetic sequence.

```
\begin{tblr}{
  cell{1}{every{2}{-2}} = {yellow9},
  cell{2}{every[2]{2}{-2}} = {purple9},
  cell{3}{every[3]{-9}{-2}} = {cyan9}
}
  A & B & C & D & E & F & G & H & I & J \\
  A & B & C & D & E & F & G & H & I & J \\
  A & B & C & D & E & F & G & H & I & J
\end{tblr}
```

A	B	C	D	E	F	G	H	I	J
A	B	C	D	E	F	G	H	I	J
A	B	C	D	E	F	G	H	I	J

The interface of `every` selector is `every[<step>]{<start>}{<end>}`, where `<start>` and `<end>` are positive or negative indexes. and they can not be child indexers such as `U` or `Z`.

More child indexers and selectors can be defined by users (see Section 7.4).

3.8 Counters and lengths

Counters `rownum`, `colnum`, `rowcount`, `colcount` can be used in cell text:

```
\begin{tblr}{hlines}
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] &
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] \\
 Row=\arabic{rowcount}, Col=\arabic{colcount} &
 Row=\arabic{rowcount}, Col=\arabic{colcount} &
 Row=\arabic{rowcount}, Col=\arabic{colcount} &
 Row=\arabic{rowcount}, Col=\arabic{colcount} \\
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] &
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] \\
\end{tblr}
```

Cell[1][1]	Cell[1][2]	Cell[1][3]	Cell[1][4]
Row=3, Col=4	Row=3, Col=4	Row=3, Col=4	Row=3, Col=4
Cell[3][1]	Cell[3][2]	Cell[3][3]	Cell[3][4]

Also, lengths `\leftsep`, `\rightsep`, `\abovesep`, `\belowsep` can be used in cell text.

3.9 Tracing tabularray

To trace internal data behind `tblr` environment, you can use `\SetTblrTracing` command. For example, `\SetTblrTracing{all}` will turn on all tracings, and `\SetTblrTracing{none}` will turn off all tracings. `\SetTblrTracing{+row,+column}` will only tracing row and column data. All tracing messages will be written to the log files.

Chapter 4

Use Long Tables

4.1 A simple example

To make a decent long table with header and footer, it is better to separate header/footer as table head/foot (which includes caption, footnotes, continuation text) and row head/foot (which includes some rows of the table that should appear in every page). By this approach, alternating row colors work as expected.

Table 4.1: A Long Long Long Long Long Long Long Table

Head	Head	Head
Head	Head	Head
Alpha	Beta	Gamma
Epsilon	Zeta[a]	Eta
Iota	Kappa[†]	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Foot	Foot	Foot

Continued on next page

Table 4.1: A Long Long Long Long Long Long Long Table (Continued)

Head	Head	Head
Head	Head	Head
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Foot	Foot	Foot

Continued on next page

Table 4.1: A Long Long Long Long Long Long Long Table (Continued)

Head	Head	Head
Head	Head	Head
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda
Nu	Xi	Omicron
Rho	Sigma	Tau
Phi	Chi	Psi
Foot	Foot	Foot

[a] It is the first footnote.

[†] It is the second long long long long long long footnote.

Note: Some general note. Some general note. Some general note.

Source: Made up by myself. Made up by myself. Made up by myself.

As you can see in the above example, the appearance of long tables of `tabularray` package is similar to that of `threeparttablex` packages. It supports table footnotes, but not page footnotes.

The source code for the above long table is shown below. It is mainly self-explanatory.

```
\NewTblrTheme{fancy}{
  \SetTblrStyle{firsthead}{font=\bfseries}
  \SetTblrStyle{firstfoot}{fg=blue2}
  \SetTblrStyle{middlefoot}{\itshape}
  \SetTblrStyle{caption-tag}{red2}
}
\begin{longtblr}[
  theme = fancy,
  caption = {A Long Long Long Long Long Long Long Table},
  entry = {Short Caption},
  label = {tblr:test},
  note{a} = {It is the first footnote.},
  note{$\dag$} = {It is the second long long long long long long footnote.},
  remark{Note} = {Some general note. Some general note. Some general note.},
  remark{Source} = {Made up by myself. Made up by myself. Made up by myself.},
]{
  colspec = {XXX}, width = 0.85\linewidth,
  rowhead = 2, rowfoot = 1,
  row{odd} = {gray9}, row{even} = {brown9},
  row{1-2} = {purple7}, row{Z} = {blue7},
}
\hline
 Head    & Head  & Head    \\
\hline
 Head    & Head  & Head    \\
\hline
 Alpha   & Beta  & Gamma   \\
\hline
 Epsilon & Zeta\TblrNote{a}       & Eta    \\
\hline
 Iota    & Kappa\TblrNote{$\dag$} & Lambda \\
\hline
 Nu      & Xi    & Omicron \\
\hline
 Rho     & Sigma & Tau     \\
\hline
 Phi     & Chi   & Psi     \\
\hline
......
\hline
 Alpha   & Beta  & Gamma   \\
\hline
 Epsilon & Zeta  & Eta     \\
\hline
 Iota    & Kappa & Lambda  \\
\hline
 Nu      & Xi    & Omicron \\
\hline
 Rho     & Sigma & Tau     \\
\hline
 Phi     & Chi   & Psi     \\
\hline
 Foot    & Foot  & Foot    \\
\hline
\end{longtblr}
```

As you can see in the above code, we typeset long tables with `longtblr` environment. And we can totally separate contents and styles of long tables with `tabularray` package.

Row head and row foot consist of some lines of the table and should appear in every page. Their options are inner specifications and should be put in the mandatory argument of the `longtblr` environment. In the above example, We set `rowhead=2` and `rowfoot=1`.

Table 4.2: Inner Specifications for Row Heads and Row Foots

Key Name	Key Description	Initial Value
`rowhead`	number of the first rows of the table appear in every page	0
`rowfoot`	number of the last rows of the table appear in every page	0

Table head and table foot consist of the caption, continuation text, footnotes and remarks. Their options are outer specifications and should be put in the optional argument of the `longtblr` environment.

Table 4.3: Outer Specifications for Table Heads and Table Foots

Key Name	Key Description	Initial Value
`headsep`	vertical space between table head and table body	`6pt`
`footsep`	vertical space between table foot and table body	`6pt`
`presep`	vertical space between table head and the above text	`1.5\bigskipamount`
`postsep`	vertical space between table foot and the below text	`1.5\bigskipamount`
`theme`	table theme (including settings for templates and styles)	×
`caption`	table caption	×
`entry`	short table caption to be put in List of Tables	×
`label`	table label	×
`note{<name>}`	table note with `<name>` as tag	×
`remark{<name>}`	table remark with `<name>` as tag	×

If you write `entry=none`, `tabularray` package will not add an entry in List of Tables. Therefore `caption=text,entry=none` is similar to `\caption[]{text}` in `longtable`.

If you write `label=none`, `tabularray` package will not step `table` counter, and set the `caption-tag` and `caption-sep` elements (see below) to empty. Therefore `caption=text,entry=none,label=none` is similar to `\caption*{text}` in `longtable`, except for the counter.

4.2 Customize templates

4.2.1 Overview of templates

The template system for table heads and table foots in `tabularray` is largely inspired by `beamer`, `caption` and `longtable` packages. For elements in Table 4.4, you can use `\DeclareTblrTemplate` to define and modify templates, and use `\SetTblrTemplate` to choose default templates. In defining templates, you can include other templates with `\UseTblrTemplate` and `\ExpTblrTemplate` commands.

Table 4.4: Elements for Table Heads and Table Foots

Element Name	Element Description and Default Template
`contfoot-text`	continuation text in the foot, normally “Continued on next page”

Continued on next page

Table 4.4: Elements for Table Heads and Table Foots (Continued)

Element Name	Element Description and Default Template
contfoot	continuation paragraph in the foot, normally including contfoot-text template
conthead-text	continuation text in the head, normally "(Continued)"
conthead	continuation paragraph in the head, normally including conthead-text template
caption-tag	caption tag, normally like "Table 4.2"
caption-sep	caption separator, normally like ": "
caption-text	caption text, normally using user provided value
caption	including caption-tag + caption-sep + caption-text
note-tag	note tag, normally using user provided value
note-sep	note separator, normally like " "
note-text	note tag, normally using user provided value
note	including note-tag + note-sep + note-text
remark-tag	remark tag, normally using user provided value
remark-sep	remark separator, normally like ": "
remark-text	remark text, normally using user provided value
remark	including remark-tag + remark-sep + remark-text
firsthead	table head on the first page, normally including caption template
middlehead	table head on middle pages, normally including caption and conthead templates
lasthead	table head on the last page, normally including caption and conthead templates
head	setting all of firsthead, middlehead and lasthead
firstfoot	table foot on the first page, normally including contfoot template
middlefoot	table foot on middle pages, normally including contfoot template
lastfoot	table foot on the last page, normally including note and remark templates
foot	setting all of firstfoot, middlefoot and lastfoot

An element which only includes short text is called a sub element. Normally there is one - in the name of a sub element. An element which includes one or more paragraphs is called a main element. Normally there isn't any - in the name of a main element.

For each of the above elements, two templates normal and empty are always defined. You can select one of them with \SetTblrTemplate command.

4.2.2 Continuation templates

Let us have a look at the code for defining templates of continuation text first:[1]

```
\DeclareTblrTemplate{contfoot-text}{normal}{Continued on next page}
\SetTblrTemplate{contfoot-text}{normal}
\DeclareTblrTemplate{conthead-text}{normal}{(Continued)}
\SetTblrTemplate{conthead-text}{normal}
```

In the above code, command \DeclareTblrTemplate defines the templates with name normal, and then command \SetTblrTemplate sets the templates with name normal as default. The normal template is always defined and set as default for any element in tabularray. Therefore you had better use another name when defining new templates.

[1] To tell the truth, the default conthead-text and contfoot-text are actually stored in commands \tblrcontheadname and \tblrcontfootname respectively. And you may contribute your translations of them to **babel** package.

If you use `default` as template name in `\DeclareTblrTemplate`, you define and set it as default at the same time. Therefore the above code can be written in another way:

```
\DeclareTblrTemplate{contfoot-text}{default}{Continued on next page}
\DeclareTblrTemplate{conthead-text}{default}{(Continued)}
```

You may modify the code to customize continuation text to fit your needs.

The templates for `contfoot` and `conthead` normally include the templates of their sub elements with `\UseTblrTemplate` commands. But you can also handle user settings such as horizontal alignment here.

```
\DeclareTblrTemplate{contfoot}{default}{\UseTblrTemplate{contfoot-text}{default}}
\DeclareTblrTemplate{conthead}{default}{\UseTblrTemplate{conthead-text}{default}}
```

4.2.3 Caption templates

Normally a caption consists of three parts, and their templates are defined with the follow code:

```
\DeclareTblrTemplate{caption-tag}{default}{Table\hspace{0.25em}\thetable}
\DeclareTblrTemplate{caption-sep}{default}{:\enskip}
\DeclareTblrTemplate{caption-text}{default}{\InsertTblrText{caption}}
```

The command `\InsertTblrText{caption}` inserts the value of `caption` key, which you could write in the optional argument of `longtblr` environment.

The `caption` template normally includes three sub templates with `\UseTblrTemplate` commands: The `caption` template will be used in `firsthead` template.

```
\DeclareTblrTemplate{caption}{default}{
  \UseTblrTemplate{caption-tag}{default}
  \UseTblrTemplate{caption-sep}{default}
  \UseTblrTemplate{caption-text}{default}
}
```

Furthermore `capcont` template includes `conthead` template as well. The `capcont` template will be used in `middlehead` and `lasthead` templates.

```
\DeclareTblrTemplate{capcont}{default}{
  \UseTblrTemplate{caption-tag}{default}
  \UseTblrTemplate{caption-sep}{default}
  \UseTblrTemplate{caption-text}{default}
  \UseTblrTemplate{conthead-text}{default}
}
```

4.2.4 Note and remark templates

The templates for table notes can be defined like this:

```
\DeclareTblrTemplate{note-tag}{default}{\textsuperscript{\InsertTblrNoteTag}}
\DeclareTblrTemplate{note-sep}{default}{\space}
\DeclareTblrTemplate{note-text}{default}{\InsertTblrNoteText}
```

```
\DeclareTblrTemplate{note}{default}{
  \MapTblrNotes{
    \noindent
    \UseTblrTemplate{note-tag}{default}
    \UseTblrTemplate{note-sep}{default}
    \UseTblrTemplate{note-text}{default}
    \par
  }
}
```

The `\MapTblrNotes` command loops for all table notes, which are written in the optional argument of `longtblr` environment. Inside the loop, you can use `\InsertTblrNoteTag` and `\InsertTblrNoteText` commands to insert current note tag and note text, respectively.

The definition of remark templates are similar to note templates.

```
\DeclareTblrTemplate{remark-tag}{default}{\InsertTblrRemarkTag}
\DeclareTblrTemplate{remark-sep}{default}{:\space}
\DeclareTblrTemplate{remark-text}{default}{\InsertTblrRemarkText}
```

```
\DeclareTblrTemplate{remark}{default}{
  \MapTblrRemarks{
    \noindent
    \UseTblrTemplate{remark-tag}{default}
    \UseTblrTemplate{remark-sep}{default}
    \UseTblrTemplate{remark-text}{default}
    \par
  }
}
```

4.2.5 Head and foot templates

The templates for table heads and foots are defined as including other templates:

```
\DeclareTblrTemplate{firsthead}{default}{
  \UseTblrTemplate{caption}{default}
}
\DeclareTblrTemplate{middlehead,lasthead}{default}{
  \UseTblrTemplate{capcont}{default}
}
\DeclareTblrTemplate{firstfoot,middlefoot}{default}{
  \UseTblrTemplate{contfoot}{default}
}
\DeclareTblrTemplate{lastfoot}{default}{
  \UseTblrTemplate{note}{default}
  \UseTblrTemplate{remark}{default}
}
```

Note that you can define the same template for multiple elements in `\DeclareTblrTemplate` command. If you only want to show table caption in the first page, you may change the definitions of `middlehead` and `lasthead` elements:

```
\DeclareTblrTemplate{middlehead,lasthead}{default}{
  \UseTblrTemplate{conthead}{default}
}
```

4.3 Change styles

All available keys for template elements are described in Table 4.5.

Table 4.5: Keys for the Styles of Elements

Key Name	Key Description	Initial Value
fg	foreground color	×
font	font commands	×
halign	horizontal alignment: `l` (left), `c` (center), `r` (right) or `j` (justify)	`j`
indent	parindent value	`0pt`
hang	hangindent value	`0pt` or `0.7em`

Note: In most cases, you can omit the underlined key names and write only their values. The keys `halign`, `indent` and `hang` are only for main templates.

You may change the styles of elements with `\SetTblrStyle` command:

```
\SetTblrStyle{firsthead}{font=\bfseries}
\SetTblrStyle{firstfoot}{fg=blue2}
\SetTblrStyle{middlefoot}{\itshape}
\SetTblrStyle{caption-tag}{red2}
```

When you write `\UseTblrTemplate{element}{default}` in defining a template, beside including template code of the `element`, the foreground color and font commands of the `element` will be set up automatically. In contrast, `\ExpTblrTemplate{element}{default}` will only include template code.

4.4 Define themes

You may define your own themes for table heads and foots with `\NewTblrTheme` command. a theme consists of some template and style settings. For example:

```
\NewTblrTheme{fancy}{
  \DeclareTblrTemplate{conthead}{default}{[Continued]}
  \SetTblrStyle{firsthead}{font=\bfseries}
  \SetTblrStyle{firstfoot}{fg=blue2}
  \SetTblrStyle{middlefoot}{\itshape}
  \SetTblrStyle{caption-tag}{red2}
}
```

After defining the theme `fancy`, you can use it by writing `theme=fancy` in the optional argument of `longtblr` environment.

4.5 Control page breaks

Just like `longtable` package, inside `longtblr` environment, you can use `\\*` or `\nopagebreak` to prohibit a page break, and use `\pagebreak` to force a page break.

4.6 Floatable tall tables

There is also a `talltblr` environment as an alternative to `threeparttable` environment. It can not cross multiple pages, but it can be put inside `table` environment.

```
TEXT\begin{talltblr}[
  caption = {Long Long Long Long Tabular},
  entry = {Short Caption},
  label = {tblr:tall},
  note{a} = {It is the first footnote.},
  note{$\dag$} = {It is the second long long long long long long footnote.},
]{
  colspec = {XXX}, width = 0.5\linewidth, hlines,
}
  Alpha   & Beta  & Gamma \\
  Epsilon & Zeta  & Eta\TblrNote{a} \\
  Iota    & Kappa & Lambda\TblrNote{$\dag$} \\
\end{talltblr}TEXT
```

Table 4.6: Long Long Long Long Tabular

Alpha	Beta	Gamma
Epsilon	Zeta	Eta[a]
Iota	Kappa	Lambda[†]

TEXT TEXT

[a] It is the first footnote.
[†] It is the second long long long long long long footnote.

Chapter 5

Use Some Libraries

A `tabularray` library could be loaded by `\UseTblrLibrary` command. From version 2025A, an external library `foo` could also be loaded if its filename is `tblrlibfoo.sty`.

5.1 Library `amsmath`

With `\UseTblrLibrary{amsmath}` in the preamble of the document, `tabularray` will load `amsmath` package, and define `+array`, `+matrix`, `+bmatrix`, `+Bmatrix`, `+pmatrix`, `+vmatrix`, `+Vmatrix` and `+cases` environments. Each of the environments is similar to the environment without `+` prefix in its name, but has default `rowsep=2pt` just as `tblr` environment. Every environment except `+array` accepts an optional argument, where you can write inner specifications.

```
$\begin{pmatrix}
 \dfrac{2}{3} &  \dfrac{2}{3} &  \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} &  \dfrac{2}{3} \\
\end{pmatrix}$
```

$$\begin{pmatrix} \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \end{pmatrix}$$

```
$\begin{+pmatrix}[cells={r},row{2}={purple8}]
 \dfrac{2}{3} &  \dfrac{2}{3} &  \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} &  \dfrac{2}{3} \\
\end{+pmatrix}$
```

$$\begin{+pmatrix}[cells={r},row{2}={purple8}] \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \end{+pmatrix}$$

```
$f(x)=\begin{cases}
 0,            & x=1; \\
 \dfrac{1}{3}, & x=2; \\
 \dfrac{2}{3}, & x=3; \\
 1,            & x=4. \\
\end{cases}$
```

$$f(x)=\begin{cases} 0, & x=1; \\ \frac{1}{3}, & x=2; \\ \frac{2}{3}, & x=3; \\ 1, & x=4. \end{cases}$$

```
$f(x)=\begin{+cases}
 0,            & x=1; \\
 \dfrac{1}{3}, & x=2; \\
 \dfrac{2}{3}, & x=3; \\
 1,            & x=4. \\
\end{+cases}$
```

$$f(x)=\begin{+cases} 0, & x=1; \\ \frac{1}{3}, & x=2; \\ \frac{2}{3}, & x=3; \\ 1, & x=4. \end{+cases}$$

5.2 Library booktabs

With `\UseTblrLibrary{booktabs}` in the preamble of the document, `tabularray` will load `booktabs` package, and define `\toprule`, `\midrule`, `\bottomrule`, `\cmidrule`, `\cmidrulemore`, `\morecmidrules`, `\specialrule`, `\addrowspace`, and `\addlinespace` as table commands.

```
\begin{tblr}{llll}
\toprule
 Alpha   & Beta  & Gamma   & Delta \\
\midrule
 Epsilon & Zeta  & Eta     & Theta \\
\cmidrule{1-3}
 Iota    & Kappa & Lambda  & Mu    \\
\cmidrule{2-4}
 Nu      & Xi    & Omicron & Pi    \\
\bottomrule
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi

Just like `\hline` and `\cline` commands, you can also specify rule width and color by using hline keys in the optional argument of any of these commands.

Like in `booktabs`, by default width of `\toprule` and `\bottomrule` are determined by `\heavyrulewidth`, width of `\midrule` is determined by `\lightrulewidth`, and width of `\cmidrule` and `\cmidrulemore` are determined by `\cmidrulewidth`, respectively. All three `\...rulewidth` are dimensions.

```
\begin{tblr}{llll}
\toprule[2pt,purple3]
 Alpha   & Beta  & Gamma  & Delta \\
\midrule[blue3]
 Epsilon & Zeta  & Eta    & Theta \\
\cmidrule[azure3]{2-3}
 Iota    & Kappa & Lambda & Mu    \\
\bottomrule[2pt,purple3]
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

If you need more than one `\cmidrules`, you can use `\cmidrulemore` command, which is simpler than the `booktabs` usage `\morecmidrules\cmidrule`. `\cmidrulemore` can receive hline keys in an optional argument too.

```
\begin{tblr}{llll}
\toprule
 Alpha   & Beta  & Gamma   & Delta \\
\cmidrule{1-3} \cmidrulemore{2-4}
 Epsilon & Zeta  & Eta     & Theta \\
\cmidrule{1-3} \morecmidrules \cmidrule{2-4}
 Iota    & Kappa & Lambda  & Mu    \\
\bottomrule
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

From version 2021N, you can set trimming positions of `\cmidrule` and `\cmidrulemore`, using newly introduced trimming options (`leftpos`, `rightpos`, `endpos`, `l`, `r`, and `lr`) (see Section 2.2). Option `endpos` is already applied to these two commands.

```
\begin{tblr}{llll}
\toprule
 Alpha   & Beta  & Gamma   & Delta \\
\cmidrule[lr]{1-2} \cmidrule[lr=-0.4]{3-4}
 Epsilon & Zeta  & Eta     & Theta \\
\cmidrule[r]{1-2} \cmidrule[l]{3-4}
 Iota    & Kappa & Lambda  & Mu    \\
\bottomrule
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Since `booktabs` tables usually don't have vlines, the meaningful values here are decimal numbers between `-1` and `0`. The default value `-0.8` for `l`, `r`, and `lr` is chosen to make similar result as `booktabs` package does.

There is also a `booktabs` environment for you. With this environment, the default `rowsep=0pt`, but extra vertical space will be added by `\toprule`, `\midrule`, `\bottomrule` and `\cmidrule` commands. The sizes of vertical space are determined by `\aboverulesep` and `\belowrulesep` dimensions.

```
\begin{booktabs}{
  colspec = lcccc,
  cell{1}{1} = {r=2}{}, cell{1}{2,4} = {c=2}{},
}
\toprule
  Sample & I &   & II &   \\
\cmidrule[lr]{2-3} \cmidrule[lr]{4-5}
         & A & B & C & D \\
\midrule
  S1     & 5 & 6 & 7 & 8 \\
  S2     & 6 & 7 & 8 & 5 \\
  S3     & 7 & 8 & 5 & 6 \\
\bottomrule
\end{booktabs}
```

Sample	I		II	
	A	B	C	D
S1	5	6	7	8
S2	6	7	8	5
S3	7	8	5	6

You can also use `\specialrule` command. The second argument sets `belowsep` of previous row, and the third argument sets `abovesep` of current row,

```
\begin{booktabs}{row{2}={olive9}}
\toprule
 Alpha   & Beta  & Gamma   & Delta \\
\specialrule{0.5pt}{4pt}{6pt}
 Epsilon & Zeta  & Eta     & Theta \\
\specialrule{0.8pt,blue3}{3pt}{2pt}
 Iota    & Kappa & Lambda  & Mu    \\
\bottomrule
\end{booktabs}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

At last, there is also an `\addlinespace` command, with an alternative name `\addrowspace`. You can specify the size of vertical space to be added in its optional argument, and the default size is determinted by `\defaultaddspace` dimension, initially `0.5em`. This command adds one half of the space to `belowsep` of previous row, and the other half to `abovesep` of current row.

```
\begin{booktabs}{row{2}={olive9}}
\toprule
 Alpha   & Beta  & Gamma   & Delta \\
\addlinespace
 Epsilon & Zeta  & Eta     & Theta \\
\addlinespace[1em]
 Iota    & Kappa & Lambda  & Mu    \\
\bottomrule
\end{booktabs}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

From version 2022A, there is a `longtabs` environment for writing long `booktabs` tables, and a `talltabs` environment for writing tall `booktabs` tables.

5.3 Library `counter`

You need to load `counter` library with `\UseTblrLibrary{counter}`, if you want to modify some LaTeX counters inside `tabularray` tables.

```
\newcounter{mycnta}
\newcommand{\mycnta}{\stepcounter{mycnta}\arabic{mycnta}}
\begin{tblr}{hlines}
  \mycnta & \mycnta & \mycnta \\
  \mycnta & \mycnta & \mycnta \\
  \mycnta & \mycnta & \mycnta \\
\end{tblr}
```

1	2	3
4	5	6
7	8	9

5.4 Library `diagbox`

When writing `\UseTblrLibrary{diagbox}` in the preamble of the document, `tabularray` package loads `diagbox` package, and you can use `\diagbox` and `\diagboxthree` commands inside `tblr` environment.

```
\begin{tblr}{hlines,vlines}
 \diagbox{Aa}{Pp} & Beta & Gamma \\
 Epsilon & Zeta  & Eta \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Aa \ Pp	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{hlines,vlines}
 \diagboxthree{Aa}{Pp}{Hh} & Beta & Gamma \\
 Epsilon & Zeta  & Eta \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Aa \ Pp \ Hh	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

You can also use `\diagbox` and `\diagboxthree` commands in math mode.

```
$\begin{tblr}{|c|cc|}
\hline
 \diagbox{X_1}{X_2} & 0 & 1 \\
\hline
  0 & 0.1 & 0.2 \\
  1 & 0.3 & 0.4 \\
\hline
\end{tblr}$
```

X_1 \ X_2	0	1
0	0.1	0.2
1	0.3	0.4

5.5 Library functional

With \UseTblrLibrary{functional} in the preamble of the document, tabularray will load functional package, and define outer key evaluate and inner key process. This library brings intuitive functional programming into tabularray tables.

5.5.1 Evaluate inner specifications

With this library, tabularray will evaluate every function (defined with \prgNewFunction) within inner specifications, replacing it with its return value, before parsing the key-value pairs. Here is an example:

```
\begin{tblr}{
  hlines,
  row{2} = {bg=\funColor{RGB}{180,180,255}}
}
  Alpha   & Beta  & Gamma  \\
  Epsilon & Zeta  & Eta    \\
  Iota    & Kappa & Lambda
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

The \funColor function is provided by functional package. And now let's see another example:

```
\begin{tblr}{
  row{2} = {bg=\intIfOddTF{\value{page}}{\prgReturn{red7}}{\prgReturn{blue7}}}
}
  Alpha   & Beta  & Gamma  \\
  Epsilon & Zeta  & Eta    \\
  Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

You may like to define a new function for it if you need to use it several times:

```
\IgnoreSpacesOn
\prgNewFunction \colorMagic {mm} {
  \intIfOddTF{\value{page}}{\prgReturn{#1}}{\prgReturn{#2}}
}
\IgnoreSpacesOff
\begin{tblr}{
  row{1} = {bg=\colorMagic{yellow7}{brown7}},
  row{3} = {bg=\colorMagic{green7}{teal7}}
}
  Alpha   & Beta  & Gamma  \\
  Epsilon & Zeta  & Eta    \\
  Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

5.5.2 Evaluate table body

With outer key `evaluate`, you can evaluate every occurrence of a specified protected function (defined with `\prgNewFunction`) and replace it with the return value before splitting the table body.

The first application of `evaluate` key is for inputting files inside tables. Assume you have two files `test1.tmp` and `test2.tmp` with the following contents:

```
\begin{filecontents*}[overwrite]{test1.tmp}
Some & Some \\
\end{filecontents*}
```

```
\begin{filecontents*}[overwrite]{test2.tmp}
Other & Other \\
\end{filecontents*}
```

Then you can input them with outer specification `evaluate=\fileInput`. The `\fileInput` function is provided by `functional` package.

```
\begin{tblr}[evaluate=\fileInput]{hlines}
  Row1 & 1 \\
  \fileInput{test1.tmp}
  Row3 & 3 \\
  \fileInput{test2.tmp}
  Row5 & 5 \\
\end{tblr}
```

Row1	1
Some	Some
Row3	3
Other	Other
Row5	5

In general, you can define your functions which return parts of table contents, and use `evaluate` key to evaluate them inside tables.

```
\IgnoreSpacesOn
\prgNewFunction \myFunOne {m} {
  \prgReturn {#1 & #1 \\}
}
\IgnoreSpacesOff
\begin{tblr}[evaluate=\myFunOne]{hlines}
  Row1 & 1 \\
  \myFunOne{Text}
  Row3 & 3 \\
  \myFunOne{Text}
  Row5 & 5 \\
\end{tblr}
```

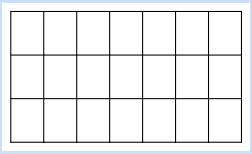
Row1	1
Text	Text
Row3	3
Text	Text
Row5	5

```
\IgnoreSpacesOn
\prgNewFunction \myFunTwo {} {
  \prgReturn {Other & Other \\}
}
\IgnoreSpacesOff
\begin{tblr}[evaluate=\myFunTwo]{hlines}
  Row1 & 1 \\
  \myFunTwo
  Row3 & 3 \\
  \myFunTwo
  Row5 & 5 \\
\end{tblr}
```

Row1	1
Other	Other
Row3	3
Other	Other
Row5	5

You can even generate the whole table with some function.

```
\IgnoreSpacesOn
\prgNewFunction \makeEmptyTable {mm}  {
  \tlSet \lTmpaTl {\intReplicate {\intEval{#2-1}} {&}}
  \tlPutRight \lTmpaTl {\\}
  \intReplicate {#1} {\tlUse \lTmpaTl}
}
\IgnoreSpacesOff
\begin{tblr}[evaluate=\makeEmptyTable]{hlines,vlines}
  \makeEmptyTable{3}{7}
\end{tblr}
```



From version 2023A, you can evaluate all functions in the table body with option `evaluate=all`.

5.5.3 Process table elements

With inner key `process`, you can modify the contents and styles before the table is built. Several public functions defined with `\prgNewFuncton` are provided for you:

- `\cellGetText{<rownum>}{<colnum>}`
- `\cellSetText{<rownum>}{<colnum>}{<text>}`
- `\cellSetStyle{<rownum>}{<colnum>}{<style>}`
- `\rowSetStyle{<rownum>}{<style>}`
- `\columnSetStyle{<colnum>}{<style>}`

As the first example, let's calculate the sums of cells column by column:

```
\IgnoreSpacesOn
\prgNewFunction \calcSum {} {
  \intStepOneInline {1} {\arabic{colcount}} {
    \intZero \lTmpaInt
    \intStepOneInline {1} {\arabic{rowcount}-1} {
      \intAdd \lTmpaInt {\cellGetText {####1} {##1}}
    }
    \cellSetText {\expWhole{\arabic{rowcount}}} {##1} {\intUse\lTmpaInt}
  }
}
\IgnoreSpacesOff
```

```
\begin{tblr}{colspec={rrr},process=\calcSum}
\hline
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\hline
    &   &   \\
\hline
\end{tblr}
```

1	2	3
4	5	6
7	8	9
12	15	18

Now, let's set background colors of cells depending on their contents:

```
\IgnoreSpacesOn
\prgNewFunction \colorBack {} {
  \intStepOneInline {1} {\arabic{rowcount}} {
    \intStepOneInline {1} {\arabic{colcount}} {
      \intSet \lTmpaInt {\cellGetText {##1} {####1}}
      \intCompareTF {\lTmpaInt} > {0}
        {\cellSetStyle {##1} {####1} {bg=purple8}}
        {\cellSetStyle {##1} {####1} {bg=olive8}}
    }
  }
}
\IgnoreSpacesOff
```

```
\begin{tblr}{hlines,vlines,cells={r,$},process=\colorBack}
  -1 &  2 &  3 \\
   4 &  5 & -6 \\
   7 & -8 &  9 \\
\end{tblr}
```

−1	2	3
4	5	−6
7	−8	9

We can also use color series of `xcolor` package to color table rows:

```
\definecolor{lightb}{RGB}{217,224,250}
\definecolorseries{tblrow}{rgb}{last}{lightb}{white}
\resetcolorseries[3]{tblrow}
\IgnoreSpacesOn
\prgNewFunction \colorSeries {} {
  \intStepOneInline {1} {\arabic{rowcount}} {
    \tlSet \lTmpaTl {\intMathMod {##1-1} {3}}
    \rowSetStyle {##1} {\expWhole{bg=tblrow!![\lTmpaTl]}}
  }
}
\IgnoreSpacesOff
```

```
\begin{tblr}{hlines,process=\colorSeries}
  Row1 & 1 \\
  Row2 & 2 \\
  Row3 & 3 \\
  Row4 & 4 \\
  Row5 & 5 \\
  Row6 & 6 \\
\end{tblr}
```

Row1	1
Row2	2
Row3	3
Row4	4
Row5	5
Row6	6

5.6 Library `hook`

This library is *experimental*. It will also load `varwidth` library and set `measure=vstore` as default. See Section 7.2 for more details of the library.

5.7 Library `html`

This library is *experimental*. See Section 7.3 for more details of the library.

5.8 Library nameref

From version 2022D, you can load `nameref` library to make `\nameref` and `longtblr` work together.

5.9 Library siunitx

When writing `\UseTblrLibrary{siunitx}` in the preamble of the document, `tabularray` package loads `siunitx` package, and defines `S` column as `Q` column with `si` key.

```
\begin{tblr}{
  hlines, vlines,
  colspec={S[table-format=3.2]S[table-format=3.2]}
}
   {Head}  & {Head} \\
  111      & 111    \\
    2.1    &   2.2  \\
   33.11   &  33.22 \\
\end{tblr}
```

Head	Head
111	111
2.1	2.2
33.11	33.22

```
\begin{tblr}{
  hlines, vlines,
  colspec={Q[si={table-format=3.2},c]Q[si={table-format=3.2},c]}
}
   {Head}  & {Head} \\
  111      & 111    \\
    2.1    &   2.2  \\
   33.11   &  33.22 \\
\end{tblr}
```

Head	Head
111	111
2.1	2.2
33.11	33.22

Note that you need to use one pairs of curly braces to guard non-numeric cells[1]. But it is cumbersome to enclose each cell with braces. From version 2022B a new key `guard` is provided for cells and rows. With `guard` key the previous example can be largely simplified.

```
\begin{tblr}{
  hlines, vlines,
  colspec={Q[si={table-format=3.2},c]Q[si={table-format=3.2},c]},
  row{1} = {guard}
}
   Head  & Head   \\
  111    & 111    \\
    2.1  &   2.2  \\
   33.11 &  33.22 \\
\end{tblr}
```

Head	Head
111	111
2.1	2.2
33.11	33.22

[1] Before version 2025A, three pairs of braces are needed.

Also you must use `l`, `c` or `r` to set horizontal alignment for non-numeric cells:

```
\begin{tblr}{
  hlines, vlines, columns={6em},
  colspec={
    Q[si={table-format=3.2,table-number-alignment=left},l,blue7]
    Q[si={table-format=3.2,table-number-alignment=center},c,teal7]
    Q[si={table-format=3.2,table-number-alignment=right},r,purple7]
  },
  row{1} = {guard}
}
  Head  & Head   & Head   \\
 111    & 111    & 111    \\
   2.1  &   2.2  &   2.3  \\
  33.11 &  33.22 &  33.33 \\
\end{tblr}
```

Head	Head	Head
111	111	111
2.1	2.2	2.3
33.11	33.22	33.33

Both `S` and `s` columns are supported. In fact, These two columns have been defined as follows:

```
\NewTblrColumnType{S}[1][]{Q[si={#1},c]}
\NewTblrColumnType{s}[1][]{Q[si={#1},c,cmd=\TblrUnit]}
```

You don't need to and are not allowed to define them again.

5.10 Library `tikz`

With this *experimental* `tikz` library,[2] you can draw `tikz` pictures below or above (short or tall) tables. This library depends on and loads `tabularray` library `hook` and `tikz` library `calc`.[3]

To draw below/above a table, write some `tikz` code inside `tblrtikzbelow`/`tblrtikzabove` environment. Both of them should be put before the table, and two compilations are needed to get desired result.

Inside `tblrtikzbelow`/`tblrtikzabove` environment, you can use these predefined nodes:

Table 5.1: Nodes created by `tikz` library

Node Name	Node Description
table node `table`	rectangle node for the whole table
cell node `<i>-<j>`	rectangle node for `cell{<i>}{<j>}`
corner node `h<i>`	coordinate node at the intersection point of `hborder{<i>}` and `vborder{1}`
corner node `v<j>`	coordinate node at the intersection point of `vborder{<j>}` and `hborder{1}`

The first example below demonstrates the table node and cell nodes:

[2] The author thanks Jasper Habicht for his contributions to this library.
[3] Please have a look at this `tikz` issue first if you need to write `\UseTblrLibrary{tikz}` in your LaTeX3 package.

```
\begin{tblrtikzbelow}
  \path[pattern color=gray9,pattern=checkerboard,
        draw=blue3, ultra thick, rounded corners]
    (table.north west) rectangle (table.south east);
\end{tblrtikzbelow}%
\begin{tblrtikzabove}
  \draw[red3, thick]
    (2-2.north west) -- (2-3.south east)
    (2-2.south west) -- (2-3.north east);
\end{tblrtikzabove}%
\begin{tblr}{hline{2-3},vline{2-4}}
  1-1 & 1-2 & 1-3 & 1-4 \\
  2-1 & 2-2 & 2-3 & 2-4 \\
  3-1 & 3-2 & 3-3 & 3-4
\end{tblr}
```

1-1	1-2	1-3	1-4
2-1	2-2	2-3	2-4
3-1	3-2	3-3	3-4

The second example below demonstrates corner nodes:

```
\begin{tblrtikzabove}
  \draw[color=white,thick]
      (h1-|v1) -- (h1-|v2) -- (h2-|v2)
    -- (h2-|v3) -- (h3-|v3) -- (h3-|v4)
    -- (h4-|v4) -- (h4-|v5) -- (h2-|v5)
    -- (h2-|v6) -- (h1-|v6);
\end{tblrtikzabove}%
\begin{tblr}{hlines={wd=4pt},vlines={wd=3pt}}
  1-1 & 1-2 & 1-3 & 1-4 & 1-5 \\
  2-1 & 2-2 & 2-3 & 2-4 & 2-5 \\
  3-1 & 3-2 & 3-3 & 3-4 & 3-5
\end{tblr}
```

1-1	1-2	1-3	1-4	1-5
2-1	2-2	2-3	2-4	2-5
3-1	3-2	3-3	3-4	3-5

By `tikz` intersection syntax, `h<i>-|v<j>` is the intersection point of `hborder{<i>}` and `vborder{<j>}`.

It is rather complicated to add full support for drawing `tikz` pictures on long tables. At present, the support is limited: only cell nodes are available for multi-page long tables. In writing drawing code, users are responsible for making sure the elements to draw are actually in current page table. These public variables might come in handy: `\lTblrRowHeadInt`, `\lTblrRowFootInt`, `\lTblrTablePageInt`, `\lTblrRowFirstInt`, `\lTblrRowLastInt` (they are described in Section 7.3). Here is an example:

Table 5.2: Long Table Tikz

Head1	Head2	Head3	Head4	Head5
2-1	2-2	2-3	2-4	2-5
3-1	3-2	3-3	3-4	3-5
4-1	4-2	4-3	4-4	4-5
5-1	5-2	5-3	5-4	5-5
6-1	6-2	6-3	6-4	6-5
7-1	7-2	7-3	7-4	7-5
8-1	8-2	8-3	8-4	8-5
9-1	9-2	9-3	9-4	9-5
0-1	0-2	0-3	0-4	0-5
1-1	1-2	1-3	1-4	1-5
Foot1	Foot2	Foot3	Foot4	Foot5

Continued on next page

Table 5.2: Long Table Tikz (Continued)

Head1	Head2	Head3	Head4	Head5
2-1	2-2	2-3	2-4	2-5
3-1	3-2	3-3	3-4	3-5
4-1	4-2	4-3	4-4	4-5
5-1	5-2	5-3	5-4	5-5
Foot1	Foot2	Foot3	Foot4	Foot5

```
\ExplSyntaxOn
\cs_generate_variant:Nn \clist_map_inline:nn {e}
\cs_new_protected:Npn \mymagic #1
  {
    \clist_map_inline:en {\ExpTblrChildClass {#1}}
      {
        \bool_lazy_and:nnT
          { \int_compare_p:n {\lTblrRowFirstInt <= \use_i:nn ##1} }
          { \int_compare_p:n {\lTblrRowLastInt >= \use_i:nn ##1} }
          { \exp_args:Noo \mymagicfill {\use_i:nn ##1} {\use_ii:nn ##1} }
      }
  }
\ExplSyntaxOff
\newcommand\mymagicfill[2]{
  \fill[teal7,rounded corners=8pt] (#1-#2.north west) rectangle (#1-#2.south east);
}
\begin{tblrtikzbelow}
  \mymagic{magic}
\end{tblrtikzbelow}%
\begin{longtblr}[
  caption = Long Table Tikz
]{
  rowhead=1, rowfoot=1, hlines, vlines, colspec={*{5}{X[r]}}
}
  Head1 & Head2 & Head3 & Head4 & Head5 \\
    2-1 &   2-2 & \SetChild{class=magic}2-3 & 2-4 & 2-5 \\
    3-1 &   3-2 &   3-3 &   3-4 &   3-5 \\
    4-1 &   4-2 &   4-3 & \SetChild{class=magic}4-4 &4-5 \\
    5-1 &   5-2 &   5-3 &   5-4 &   5-5 \\
    6-1 &   6-2 &   6-3 &   6-4 &   6-5 \\
    7-1 &   \SetChild{class=magic}7-2 & 7-3 & 7-4 & 7-5 \\
    8-1 &   8-2 &   8-3 &   8-4 & \SetChild{class=magic}8-5 \\
    9-1 &   9-2 &   9-3 &   9-4 &   9-5 \\
    0-1 &   0-2 & \SetChild{class=magic}0-3 & 0-4 & 0-5 \\
    1-1 &   1-2 &   1-3 &   1-4 &   1-5 \\
   \SetChild{class=magic}2-1 & 2-2 & 2-3 & 2-4 & 2-5 \\
    3-1 &   3-2 &   3-3 &   3-4 &   3-5 \\
    4-1 &   4-2 &   4-3 & \SetChild{class=magic}4-4 &4-5 \\
    5-1 &   5-2 &   5-3 &   5-4 &   5-5 \\
  Foot1 & Foot2 & Foot3 & Foot4 & Foot5 \\
\end{longtblr}
```

5.11 Library varwidth

To build a nice table, tabularray need to measure the widths of cells. By default, it uses \hbox to measure the sizes. This causes an error if a cell contains some vertical material, such as lists or display maths.

With varwidth library, tabularray will load varwidth package, add a new inner specification measure, and set measure=vbox so that it will use \vbox to measure cell widths.

```
\begin{tblr}{hlines,measure=vbox}
  Text Text Text Text Text Text Text
  \begin{itemize}
    \item List List List List List List
    \item List List List List List List List
  \end{itemize}
  Text Text Text Text Text Text Text \\
\end{tblr}
```

Text Text Text Text Text Text Text

- List List List List List List
- List List List List List List List

Text Text Text Text Text Text Text

From version 2022A, you can remove extra space above and below lists, by adding option stretch=-1. The following example also needs enumitem package and its nosep option:

• List List List List List • List List List List List List	oooo
• List List List List List • List List List List List List	gggg

```
\begin{tblr}{
  hlines,vlines,rowspec={Q[l,t]Q[l,b]},
  measure=vbox,stretch=-1,
}
  \begin{itemize}[nosep]
    \item List List List List List
    \item List List List List List List
  \end{itemize} & oooo \\
  \begin{itemize}[nosep]
    \item List List List List List
    \item List List List List List List
  \end{itemize} & gggg \\
\end{tblr}
```

Note that option stretch=-1 also removes struts from cells, therefore it may not work well in tabularray environments with rowsep=0pt, such as booktabs/longtabs/talltabs environments from booktabs library.

From version 2025A, measure key also accepts an *experimental* vstore value. With measure=vstore, tabularray also measures cells with \vbox, but it will store the boxes for later use, which is necessary to make \lTblrMeasuringBool status correct.

From version 2025A, the setting of measure key also applies to subtables.

5.12 Library zref

From version 2022D, you can load zref library to make \zref and longtblr work together.

Chapter 6

Tips and Tricks

6.1 Default rule widths and colors

From version 2025A, default hrule and vrule widths are stored in variables `\lTblrDefaultHruleWidthDim` and `\lTblrDefaultVruleWidthDim` respectively, and default hrule and vrule colors are stored in variables `\lTblrDefaultHruleColorTl` and `\lTblrDefaultVruleColorTl` respectively. Here is an example:

```
\setlength\lTblrDefaultHruleWidthDim{1pt}%
\setlength\lTblrDefaultVruleWidthDim{2pt}%
\renewcommand\lTblrDefaultHruleColorTl{blue5}%
\renewcommand\lTblrDefaultVruleColorTl{red5}%
\begin{tblr}{
  hlines, hline{2} = {wd=2pt, fg=cyan5},
  vlines, vline{2} = {wd=1pt, fg=green5}
}
  Alpha   & Beta  & Gamma  \\
  Epsilon & Zeta  & Eta    \\
  Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

6.2 Control horizontal alignment

You can control horizontal alignment of cells in `tabularray` with `ragged2e` package, by redefining some of the following commands:

```
\RenewDocumentCommand\TblrAlignBoth{}{\justifying}
\RenewDocumentCommand\TblrAlignLeft{}{\RaggedRight}
\RenewDocumentCommand\TblrAlignCenter{}{\Centering}
\RenewDocumentCommand\TblrAlignRight{}{\RaggedLeft}
```

Please read the documentation of `ragged2e` package for more details of their alignment commands.

6.3 Use safe verbatim commands

Due to the limitations of TeX, we are not able to make `\verb` command behave well inside `tabularray` tables. As a replacement, you may use `\fakeverb` command from `codehigh` package.

The `\fakeverb` command will remove the backslashes in the following control symbols before typesetting its content: `\\`, `\{`, `\}`, `\#`, `\^` and `\␣`, `\%`. Also the argument of `\fakeverb` command need to be enclosed with curly braces. Therefore it could be safely used inside `tabularray` tables and other LaTeX commands.

Here is an example of using `\fakeverb` commands inside a `tblr` environment:

```
\begin{tblr}{hlines}
  Special & \fakeverb{\abc{}$&^_^uvw 123} \\
  Spacing & \fakeverb{\bfseries\ \#\%}    \\
  Nesting & \fbox{\fakeverb{$\left\\\{A\right.$\#}}
\end{tblr}
```

Special	`\abc{}$&^_^uvw 123`
Spacing	`\bfseries #%`
Nesting	`$\left\{A\right.$#`

In the above example, balanced curly braces and control words (such as `\bfseries`) need not to be escaped—only several special characters need to be escaped. Please read the documentation of `codehigh` package for more details of `\fakeverb` commands.[1]

6.4 Blank lines around cells

In `tabularray` tables, there could be a blank line before a cell, after a cell, or between table commands and cell text. Here is an example:

```
\begin{tblr}{rl}

\hline

  One

  &

  Two

  \\

\hline

  Three

  &

  Four

  \\

\hline

\end{tblr}
```

One	Two
Three	Four

But more blank lines are not supported. Therefore putting more than one blank line at any of these positions may cause wrong result.

[1] By the way, `\EscVerb` command from `fvextra` package is similar to `\fakeverb` command, but with `\EscVerb` you need to escape every control word.

Chapter 7

Experimental Interfaces

The interfaces in this chapter (and other undocumented public interfaces even if mentioned in the changelog) should be seen as *experimental* and are likely to change in future releases, if necessary. Don't use them in important documents.

7.1 Experimental public key paths

In version 2025A, all `tabularray` key paths were cleaned up as follows:

- `tabularray/table/inner` (from `tblr`): for inner specifications.
- `tabularray/table/outer` (from `tblr-outer`): for outer specifications.
- `tabularray/column/inner` (from `tblr-column`): for column specifications.
- `tabularray/row/inner` (from `tblr-row`): for row specifications.
- `tabularray/cell/inner` (from `tblr-cell-spec`): for cell specifications.
- `tabularray/cell/outer` (from `tblr-cell-span`): for cell spanning specifications.
- `tabularray/hline/inner` (from `tblr-hline`): for hline specifications.
- `tabularray/vline/inner` (from `tblr-vline`): for vline specifications.
- `tabularray/hborder/inner` (from `tblr-hborder`) for hborder specifications.
- `tabularray/vborder/inner` (from `tblr-vborder`) for vborder specifications.

An advanced user or package writer can use `\DeclareKeys` and `\SetKeys` commands (provided by LaTeX format) to declare new keys and apply key-value lists, respectively.

The key paths are quite long, therefore `tabularray` provides two shortcut commands `\DeclareTblrKeys` and `\SetTblrKeys`:

```
\DeclareTblrKeys{<path>}{<keyvals>} = \DeclareKeys[tabularray/<path>]{<keyvals>}
\SetTblrKeys{<path>}{<keyvals>} = \SetKeys[tabularray/<path>]{<keyvals>}
```

7.2 Experimental public hook names

All experimental public `tabularray` hook names provided by `hook` library are as follows:

- `tabularray/trial/before`: hook before trial typesetting.
- `tabularray/trial/after`: hook after trial typesetting.
- `tabularray/table/before`: hook before building the whole table.
- `tabularray/table/after`: hook after building the whole table.
- `tabularray/row/before`: hook before typesetting a table row.
- `tabularray/row/after`: hook after typesetting a table row.
- `tabularray/cell/before`: hook before typesetting a table cell.
- `tabularray/cell/after`: hook after typesetting a table cell.

An advanced user or package writer can use `\AddToHook` and `\AddToHookNext` commands (provided by LaTeX format) to inject code to `tabularray` tables.

The hook names are quite long, therefore `tabularray` provides two shortcut commands `\AddToTblrHook` and `\AddToTblrHookNext`:

```
\AddToTblrHook{<name>}{<code>} = \AddToHook{tabularray/<name>}{<code>}
\AddToTblrHookNext{<name>}{<code>} = \AddToHookNext{tabularray/<name>}{<code>}
```

7.3 Experimental public variables

This variable can be used to change page break settings for multirow cells:

- `\lTblrCellBreakBool`: whether to allow page breaks in the middle of multirow cells.

This variable is always available throughout the whole typesetting process of tables:

- `\lTblrMeasuringBool`: if `tabularray` is doing trial typesetting.

You need to make sure `measure=vstore` to make `\lTblrMeasuringBool` correct.

This variable is available before building every table:

- `\lTblrPortraitTypeTl`: table type (`short`, `tall` or `long`).

These variables are updated in building long tables:

- `\lTblrRowHeadInt`: total number of head rows.
- `\lTblrRowFootInt`: total number of foot rows.
- `\lTblrTablePageInt`: index number of current page table.
- `\lTblrRowFirstInt`: first row number in row body of current page table.
- `\lTblrRowLastInt`: last row number in row body of current page table.

These variables are updated by default before building every cell:

- `\lTblrCellRowSpanInt`: how many rows are spanned by current cell.
- `\lTblrCellColSpanInt`: how many columns are spanned by current cell.
- `\lTblrCellOmittedBool`: if current cell is spanned by another cell.
- `\lTblrCellBackgroundTl`: background color of current cell.

These variables are updated by `html` library before building every cell:

- `\lTblrCellAboveBorderStyleTl`
- `\lTblrCellAboveBorderWidthDim`
- `\lTblrCellAboveBorderColorTl`
- `\lTblrCellBelowBorderStyleTl`
- `\lTblrCellBelowBorderWidthDim`
- `\lTblrCellBelowBorderColorTl`
- `\lTblrCellLeftBorderStyleTl`
- `\lTblrCellLeftBorderWidthDim`
- `\lTblrCellLeftBorderColorTl`
- `\lTblrCellRightBorderStyleTl`
- `\lTblrCellRightBorderWidthDim`
- `\lTblrCellRightBorderColorTl`

In the above, `BorderStyle`, `BorderWidth`, `BorderColor` are similar to `border-style`, `border-width`, `border-color` in HTML/CSS, respectively. `BorderStyle` and `BorderColor` are empty by default.

7.4 New child indexers and selectors

7.4.1 One dimensional indexers and selectors

You can define new child indexers with `\NewTblrChildIndexer` command. As an example, the following is the simplified definition of `Z` indexer:

```
\ExplSyntaxOn
\NewTblrChildIndexer {Z} [1] [1]
  {
    \tl_set:Ne \lTblrChildIndexTl { \int_eval:n {\lTblrChildTotalInt + 1 - #1} {1} }
  }
\ExplSyntaxOff
```

In the definition, you can use `\lTblrChildTotalInt` which is the total number of children. And you only need to store the result index `<i>` in `\lTblrChildIndexTl`. The name of an indexer *must* consist of letters and start with an uppercase letter.

You can define new child selectors with `\NewTblrChildSelector` command. As an example, the following is the simplified definition of `odd` selector:

```
\ExplSyntaxOn
\NewTblrChildSelector {odd} [1] [1]
  {
    \clist_set:Ne \lTblrChildClist { {#1} {2} {\int_use:N \lTblrChildTotalInt} }
  }
\ExplSyntaxOff
```

In the definition, you can use `\lTblrChildTotalInt` which is the total number of children. And you only need to store the result indexes in `\lTblrChildClist`. When some indexes form an arithmetic sequence, you can simplify them as `{<start>}{<step>}{<end>}`. The name of a selector *must* consist of letters and start with a lowercase letter.

7.4.2 Two dimensional indexers and selectors

When selecting cells, you may need two dimensional indexers and selectors. You can also define new two dimensional child indexers with `\NewTblrChildIndexer` command, and two dimensional child selectors with `\NewTblrChildSelector` command.

In the definitions, you can use `\lTblrChildHtotalInt` which is the total number of horizontal children (rows), and `\lTblrChildVtotalInt` which is the total number of vertical children (columns).

You also need to store the result index `{<i>}{<j>}` in `\lTblrChildIndexTl` in defining two dimensional child indexers. Similarly you also need to store the result indexes in `\lTblrChildClist` in defining two dimensional child selectors.

7.4.3 Child ids and classes

When the table is long, it is clumsy to select children with indexes, positive or negative. In version 2025A, `tabularray` borrows ideas of ids and classes from HTML/CSS. With table command `\SetChild`, you can mark a hborder/vborder/row/column/cell with an id or class, and use it in inner specifications.

The `\SetChild` command accepts key-value input:

Table 7.1: Key-value input in `\SetChild` command

Input	Description
`id=Hello`	create a child indexer `Hello` which is an index `{<i>}{<j>}`
`idh=Hello`	create a child indexer `Helloh` which is a horizontal index `<i>`
`idv=Hello`	create a child indexer `Hellov` which is a vertical index `<j>`
`id*=Hello`	create all of the above three child indexers
`class=world`	create a child selector `world` which is a list of indexes `{<i>}{<j>}`

Continued on next page

Table 7.1: Key-value input in `\SetChild` command (Continued)

Input	Description
`classh=world`	create a child selector `worldh` which is a list of horizontal indexes `<i>`
`classv=world`	create a child selector `worldv` which is a list of vertical indexes `<j>`
`class*=world`	create all of the above three child selectors

The following is an example of child ids (every id name must start with uppercase letter since it creates a child indexer):

```
\begin{tblr}{
  hline{1,Z},
  row{Barh,Quxh} = {bg=azure7},
  column{Bazv,Quxv} = {fg=red3},
  cell{Foo,Qux} = {cmd=\fbox}
}
  1 & 2 & 3 & \SetChild{id=Foo}  4 & 5 & 6 & 7 & 8 \\
  2 & 2 & \SetChild{idh=Bar} 3 & 4 & 5 & 6 & 7 & 8 \\
  3 & \SetChild{idv=Baz} 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  \SetChild{id*=Qux} 4 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{tblr}
```

1	2	3	4	5	6	7	8
2	2	3	4	5	6	7	8
3	2	3	4	5	6	7	8
4	2	3	4	5	6	7	8

The following is an example of child classes (every class name must start with lowercase letter since it creates a child selector):

```
\begin{tblr}{
  hline{1,Z},
  row{fooh} = {bg=azure7},
  column{barv} = {fg=red3},
  cell{baz} = {cmd=\fbox}
}
  1 & 2 & \SetChild{classh=foo} 3 & 4 & 5 & 6 & 7 \\
  2 & \SetChild{classv=bar} 2 & 3 & 4 & 5 & 6 & 7 \\
  \SetChild{class=baz} 3  & 2 & 3 & 4 & 5 & 6 & 7 \\
  4 & 2 & 3 & 4 & \SetChild{class=baz}  5 & 6 & 7 \\
  5 & 2 & 3 & 4 & 5 & \SetChild{classh=foo} 6 & 7 \\
  6 & 2 & 3 & 4 & 5 & 6 & \SetChild{classv=bar} 7
\end{tblr}
```

1	2	3	4	5	6	7
2	2	3	4	5	6	7
3	2	3	4	5	6	7
4	2	3	4	5	6	7
5	2	3	4	5	6	7
6	2	3	4	5	6	7

Since `\SetChild` commands need to be extracted first before parsing inner specifications, they *must* be put at the beginning of cells, before other table commands such as `\hline`. Therefore it conflicts with syntaxes `\\[<dimen>]` and `\\*`. They can be replaced with `\\\SetRow{belowsep+=<dimen>}` and `\\\nopagebreak` respectively, so that `\SetChild` can be inserted in the middle:

```
\begin{tblr}{cell{Foo,Bar} = {fg=red3}}
\hline
  1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\SetChild{id=Foo}\SetRow{belowsep+=5pt}\hline
  2 & 2 & 3 & 4 & 5 & 6 & 7 \\
\SetChild{id=Bar}\nopagebreak\hline
  3 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
\end{tblr}
```

1	2	3	4	5	6	7
2	2	3	4	5	6	7
3	2	3	4	5	6	7

Only one `\SetChild` command in each cell is supported. But you can create multiple ids or classes with single `\SetChild` command.

In drawing `tikz` pictures on tables (see Section 5.10), you may want to get the value of a child id or class with `\ExpTblrChildId` or `\ExpTblrChildClass`. These two commands are fully expandable.

Chapter 8

History and Future

8.1 The future

As a policy, `tabularray` can support at most four TeX Live releases with latest updates. For example, `tabularray` releases published in 2025 could be used in TeX Live 2022–2025 with latest updates.

To make the upcoming releases more stable, you are very welcome to test the latest package file in the repository. To test it, you only need to download the following `tabularray.sty` and put it into the folder of your TeX documents:

https://github.com/lvjr/tabularray/raw/main/tabularray.sty

8.2 The history

The change log of `tabularray` package will be updated on the wiki page:

https://github.com/lvjr/tabularray/wiki/ChangeLog

When `tabularray` makes some breaking changes of *stable public* interfaces in a new release, you will be able to roll back to previous release to make your existing documents unaffected.

Normally you don't know when there will be a new breaking release. To keep your old documents as they are, you may add the date of current release to the last optional argument of `\usepackage` in loading `tabularray`, such as `\usepackage{tabularray}[=2024-02-16]`.

8.2.1 Important changes in version 2025A

In version 2025A, there were several important changes:

- Inner key `verb` (deprecated before) was removed; it is better to use `\fakeverb` command.
- Support for end index in `odd` and `even` selectors was removed; it is better to use `every` selector.
- Page breaks in the middle of multirow cells were disabled.
- `\DefTblrTemplate` was deprecated in favor of `\DeclareTblrTemplate`.
- `\NewColumnType` was deprecated in favor of `\NewTblrColumnType`.
- `\NewRowType` was deprecated in favor of `\NewTblrRowType`.
- `\NewColumnRowType` was deprecated in favor of `\NewTblrColumnRowType`.
- `\NewDashStyle` was deprecated in favor of `\NewTblrDashStyle`.
- `\NewChildSelector` was deprecated in favor of `\NewTblrChildSelector`.
- `\NewTableCommand` was deprecated in favor of `\NewTblrTableCommand`.
- `\tablewidth` was deprecated in favor of `\lTblrTableWidthDim`.

For your old documents, you can still rollback to version 2024 by `\usepackage{tabularray}[=v2024]`.

8.2.2 Important changes in version 2022A

In version 2022A, there were several breaking changes:

- `\multicolumn` command was removed; it is better to use `\SetCell` command.
- `\multirow` command was removed; it is better to use `\SetCell` command.
- `\firsthline` command was removed; it is better to use `baseline=T` option.
- `\lasthline` command was removed; it is better to use `baseline=B` option.

For your old documents, you can still rollback to version 2021 by `\usepackage{tabularray}[=v2021]`.

Chapter 9

The Source Code

9.1 Scratch variables and function variants

```
%% \DeclareRelease and \DeclareCurrentRelease are added in LaTeX 2018-04-01
\NeedsTeXFormat{LaTeX2e}[2018-04-01]
\providecommand\DeclareRelease[3]{}
\providecommand\DeclareCurrentRelease[2]{}
\DeclareRelease{v2024}{2022-01-01}{tabularray-2024.sty}
\DeclareRelease{v2021}{2021-01-01}{tabularray-2021.sty}
\DeclareCurrentRelease{}{2025-01-01}

\ProvidesExplPackage{tabularray}{2025-11-27}{2025C}
  {Typeset tabulars and arrays with LaTeX3}

%% \IfFormatAtLeastTF, xparse and lthooks are added in LaTeX 2020-10-01
%% Note that \@ifl@t@r or \@ifpackagelater means 'this date or later'
\msg_new:nnn { tabularray } { latex-too-old }
  {
    Your ~ LaTeX ~ release ~ is ~ too ~ old. \\
    Please ~ update ~ it ~ to ~ 2022-11-01 ~ first.
  }
\@ifl@t@r\fmtversion{2022-11-01}{}{
  \msg_error:nn { tabularray } { latex-too-old }
}

\AddToHook {package/xcolor/after} [tabularray] { \RequirePackage{ninecolors} }

%% There is a hyperref bug that NoHyper environment could not be nested.
%% See issue #620 and https://github.com/latex3/hyperref/issues/397
\newenvironment{tblrNoHyper}{}{}
\int_new:N \g__tblr_nohyper_int
\AddToHook {package/hyperref/after} [tabularray]
  {
    \renewenvironment{tblrNoHyper}
      {
        \int_compare:nNnT {\g__tblr_nohyper_int} = {0} {\begin{NoHyper}}
        \int_gincr:N \g__tblr_nohyper_int
      }
      {
        \int_gdecr:N \g__tblr_nohyper_int
        \int_compare:nNnT {\g__tblr_nohyper_int} = {0} {\end{NoHyper}}
      }
```

```
  }
\NewDocumentCommand \TblrParboxRestore { } { \@parboxrestore }

\NewDocumentCommand \TblrAlignBoth { }
  {
    \let \\ = \@normalcr
    \leftskip = \z@skip
    \@rightskip = \z@skip
    \rightskip = \@rightskip
    \parfillskip = \@flushglue
  }

\NewDocumentCommand \TblrAlignLeft { } { \raggedright }

\NewDocumentCommand \TblrAlignCenter { } { \centering }

\NewDocumentCommand \TblrAlignRight { } { \raggedleft }

\cs_set_eq:NN \TblrNewPage \newpage

%% Note that \cs_if_exist:NTF doesn't treat \relax as an existing command.
%% Therefore we define our \__tblr_cs_if_defined:NTF here.
\prg_set_conditional:Npnn \__tblr_cs_if_defined:N #1 { p, T, F, TF }
  {
    %% \if_cs_exist:N = \ifdefined in eTeX
    \if_cs_exist:N #1
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
\prg_set_conditional:Npnn \__tblr_cs_if_defined:c #1 { p, T, F, TF }
  {
    %% \if_cs_exist:w = \ifcsname in eTeX
    \if_cs_exist:w #1 \cs_end:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }

\cs_generate_variant:Nn \msg_error:nnnnn { nnnVV }
\cs_generate_variant:Nn \seq_map_indexed_inline:Nn { cn }
\cs_generate_variant:Nn \seq_set_split:Nnn { NVe }
\cs_generate_variant:Nn \seq_set_split_keep_spaces:Nnn { Ne }
\cs_generate_variant:Nn \seq_use:Nn { Ne }
\cs_generate_variant:Nn \tl_gput_right:Nn { Nf }
\cs_generate_variant:Nn \tl_rescan:nn { ne }
\prg_generate_conditional_variant:Nnn \clist_if_in:Nn { Ne } { TF }
\prg_generate_conditional_variant:Nnn \str_if_eq:nn { en } { TF }
\prg_generate_conditional_variant:Nnn \str_if_in:Nn { Ne } { TF }

%% Add missing function variants for texlive 2022. They can be removed in 2026.
\cs_generate_variant:Nn \clist_gput_right:Nn { ce }
\cs_generate_variant:Nn \clist_put_right:Nn { ce }
\cs_generate_variant:Nn \clist_set:Nn { Ne, ce }
```

```
\cs_generate_variant:Nn \keyval_parse:NNn { NNV }
\cs_generate_variant:Nn \msg_error:nnn { nnV }
\cs_generate_variant:Nn \msg_error:nnnn { nnV }
\cs_generate_variant:Nn \prop_item:Nn { Ne, NV }
\cs_generate_variant:Nn \prop_put:Nnn { Nne, Nen, Nee, NeV }
\cs_generate_variant:Nn \seq_put_right:Nn { Ne }
\cs_generate_variant:Nn \seq_set_split:Nnn { Nne, NVn }
\cs_generate_variant:Nn \str_gset:Nn { Ne }
\cs_generate_variant:Nn \tl_const:Nn { ce }
\cs_generate_variant:Nn \tl_gput_right:Nn { Ne }
\cs_generate_variant:Nn \tl_gset:Nn { Ne, ce }
\cs_generate_variant:Nn \tl_log:n { e }
\cs_generate_variant:Nn \tl_put_left:Nn { Ne, Nv }
\cs_generate_variant:Nn \tl_put_right:Nn { Ne }
\cs_generate_variant:Nn \tl_set:Nn { Ne, ce }
\cs_generate_variant:Nn \tl_set_rescan:Nnn { Nne, NnV }
\cs_generate_variant:Nn \tl_to_str:n { e }
\prg_generate_conditional_variant:Nnn \prop_if_in:Nn { c } { T }
\prg_generate_conditional_variant:Nnn \tl_if_eq:nn { en } { T, TF }
\prg_generate_conditional_variant:Nnn \tl_if_in:nn { nV } { TF }
\prg_generate_conditional_variant:Nnn \tl_if_in:Nn { Ne } { TF }
\prg_generate_conditional_variant:Nnn \tl_if_head_eq_catcode:nN { VN } { TF }
\prg_generate_conditional_variant:Nnn \tl_if_head_eq_meaning:nN { VN } { T, TF }

\tl_new:N  \l__tblr_a_tl
\tl_new:N  \l__tblr_b_tl
\tl_new:N  \l__tblr_c_tl
\tl_new:N  \l__tblr_d_tl
\tl_new:N  \l__tblr_e_tl
\tl_new:N  \l__tblr_f_tl
\tl_new:N  \l__tblr_h_tl
\tl_new:N  \l__tblr_i_tl  % for row index
\tl_new:N  \l__tblr_j_tl  % for column index
\tl_new:N  \l__tblr_k_tl
\tl_new:N  \l__tblr_n_tl
\tl_new:N  \l__tblr_o_tl
\tl_new:N  \l__tblr_r_tl
\tl_new:N  \l__tblr_s_tl
\tl_new:N  \l__tblr_t_tl
\tl_new:N  \l__tblr_u_tl
\tl_new:N  \l__tblr_v_tl
\tl_new:N  \l__tblr_w_tl
\tl_new:N  \l__tblr_x_tl
\tl_new:N  \l__tblr_y_tl
\int_new:N \l__tblr_a_int
\int_new:N \l__tblr_c_int % for column number
\int_new:N \l__tblr_r_int % for row number
\dim_new:N \l__tblr_d_dim % for depth
\dim_new:N \l__tblr_h_dim % for height
\dim_new:N \l__tblr_o_dim
\dim_new:N \l__tblr_p_dim
\dim_new:N \l__tblr_q_dim
\dim_new:N \l__tblr_r_dim
\dim_new:N \l__tblr_s_dim
\dim_new:N \l__tblr_t_dim
\dim_new:N \l__tblr_v_dim
\dim_new:N \l__tblr_w_dim % for width
\box_new:N \l__tblr_a_box
```

```
\box_new:N \l__tblr_b_box
\box_new:N \l__tblr_c_box % for cell box
\box_new:N \l__tblr_d_box

%% Some commands for horizontal alignment
\cs_new_eq:NN \__tblr_halign_command_j: \TblrAlignBoth
\cs_new_eq:NN \__tblr_halign_command_l: \TblrAlignLeft
\cs_new_eq:NN \__tblr_halign_command_c: \TblrAlignCenter
\cs_new_eq:NN \__tblr_halign_command_r: \TblrAlignRight

%% Total number of tblr tables, used for creating hyperref targets and tikz nodes.
%% We need to save and restore it before and after measuring stage respectively,
%% so we must define it with \newcounter command.
\newcounter { tblrcount }

%% Some counters for row and column numbering.
%% We may need to restore all LaTeX counters in measuring and building cells,
%% so we must not define these counters with \newcounter command.
\int_zero_new:N \c@rownum
\int_zero_new:N \c@colnum
\int_zero_new:N \c@rowcount
\int_zero_new:N \c@colcount

%% Add missing \therownum, \thecolnum, \therowcount, \thecolcount (issue #129)
\ProvideExpandableDocumentCommand \therownum {} { \@arabic \c@rownum }
\ProvideExpandableDocumentCommand \thecolnum {} { \@arabic \c@colnum }
\ProvideExpandableDocumentCommand \therowcount {} { \@arabic \c@rowcount }
\ProvideExpandableDocumentCommand \thecolcount {} { \@arabic \c@colcount }

%% Some dimensions for row and column spacing
\dim_new:N \abovesep
\dim_new:N \belowsep
\dim_new:N \leftsep
\dim_new:N \rightsep

%% Some functions for lwarp to remove rules and boxes
\cs_new:Npn \tblr_hrule_ht:n #1
  {
    \hrule height ~ #1 \scan_stop:
  }
\cs_new:Npn \tblr_vrule_wd_ht_dp:nnn #1 #2 #3
  {
    \vrule width ~ #1 ~ height ~ #2 ~ depth ~ #3 \scan_stop:
  }
\cs_new_protected:Npn \tblr_box_use:N #1
  {
    \box_use:N #1
  }
\cs_new_protected:Npn \tblr_vbox_set:Nn #1 #2
  {
    \vbox_set:Nn #1 {#2}
  }
```

9.2 Functions for splitting, extracting and matching

```
\str_const:Nn \c__tblr_left_bracket_str { [ }

\cs_new_protected:Npn \__tblr_split_before_brace:NNn #1 #2 #3
  {
    \__tblr_split_before:NNVn #1 #2 \c_left_brace_str {#3}
  }

\cs_new_protected:Npn \__tblr_split_before_bracket:NNn #1 #2 #3
  {
    \__tblr_split_before:NNVn #1 #2 \c__tblr_left_bracket_str {#3}
  }
\cs_generate_variant:Nn \__tblr_split_before_bracket:NNn { NNV }

%% Split tl #2 as two parts: sub tl before first #3 and sub tl from first #1.
%% And the results are stored in tl vars #1 and #2, respectively.
\cs_new_protected:Npn \__tblr_split_before:NNnn #1 #2 #3 #4
  {
    \cs_set_protected:Npn \__tblr_split_before_auxa:ww ##1 #3 ##2 \q_stop
      {
        \tl_set:Nn #1 { ##1 }
        \tl_if_empty:nTF { ##2 }
          { \tl_set:Nn #2 { ##2 } } { \__tblr_split_before_auxb:w ##2 \q_stop }
      }
    \cs_set_protected:Npn \__tblr_split_before_auxb:w ##1 #3 \q_stop
      {
        \tl_set:Nn #2 { #3 ##1 }
      }
    \__tblr_split_before_auxa:ww #4 #3 \q_stop
  }
\cs_generate_variant:Nn \__tblr_split_before:NNnn { NNV }

%% The name of a child indexer must start with uppercase letter.
%% The name of a child selector must start with lowercase letter.

\tl_const:Nn \c__tblr_upper_letter_tl { ABCDEFGHIJKLMNOPQRSTUVWXYZ }
\tl_const:Nn \c__tblr_lower_letter_tl { abcdefghijklmnopqrstuvwxyz }
\tl_const:Nn \c__tblr_digit_str { 0123456789 }

\prg_new_protected_conditional:Npnn \__tblr_if_head_upper:n #1 { TF }
  {
    \tl_if_in:NeTF \c__tblr_upper_letter_tl { \tl_head:n { #1 } }
      { \prg_return_true: } { \prg_return_false: }
  }

\prg_new_protected_conditional:Npnn \__tblr_if_head_lower:n #1 { TF }
  {
    \tl_if_in:NeTF \c__tblr_lower_letter_tl { \tl_head:n { #1 } }
      { \prg_return_true: } { \prg_return_false: }
  }

\prg_new_protected_conditional:Npnn \__tblr_tl_if_upper:n #1 { TF }
  {
    \tl_if_in:NnTF \c__tblr_upper_letter_tl { #1 }
      { \prg_return_true: } { \prg_return_false: }
```

```
  }
\prg_generate_conditional_variant:Nnn \__tblr_tl_if_upper:n { V } { TF }

\prg_new_protected_conditional:Npnn \__tblr_tl_if_lower:n #1 { TF }
  {
    \tl_if_in:NnTF \c__tblr_lower_letter_tl { #1 }
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__tblr_tl_if_lower:n { V } { TF }

\prg_new_protected_conditional:Npnn \__tblr_tl_if_digit:n #1 { F, TF }
  {
    \tl_if_in:NnTF \c__tblr_digit_tl { #1 }
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__tblr_tl_if_digit:n { e, V } { F, TF }

%% When the key name is omitted, we need to detect the name from its value.

\str_const:Nn \c__tblr_letter_str
  { ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz }

%% It is enough to check only the first item
\prg_new_protected_conditional:Npnn \__tblr_if_color_value:n #1 { T, F, TF }
  {
    \str_if_in:NeTF \c__tblr_letter_str { \tl_head:n { #1 } }
      { \prg_return_true: } { \prg_return_false: }
  }

\str_const:Nn \c__tblr_number_str { 0123456789+-.~ }
\bool_new:N \l__tblr_if_number_bool

%% The number value could be '25.6 - 3.14'.
\prg_new_protected_conditional:Npnn \__tblr_if_number_value:n #1 { T, F, TF }
  {
    \bool_set_true:N \l__tblr_if_number_bool
    \tl_map_inline:nn { #1 }
      {
        \str_if_in:NnF \c__tblr_number_str { ##1 }
          {
            \bool_set_false:N \l__tblr_if_number_bool
            \tl_map_break:
          }
      }
    \bool_if:NTF \l__tblr_if_number_bool
      { \prg_return_true: } { \prg_return_false: }
  }

\tl_const:Nn \c__tblr_digit_tl { 0123456789 }

%% It is enough to check only the first item
\prg_new_protected_conditional:Npnn \__tblr_if_positive_value:n #1 { T, F, TF }
  {
    \tl_if_in:NeTF \c__tblr_digit_tl { \tl_head:n { #1 } }
      { \prg_return_true: } { \prg_return_false: }
```

```
  }
\prg_generate_conditional_variant:Nnn \__tblr_if_positive_value:n { V } { TF }

%% It is enough to check only the first item
\prg_new_protected_conditional:Npnn \__tblr_if_negative_value:n #1 { T, F, TF }
  {
    \tl_if_eq:enTF { \tl_head:n { #1 } } { - }
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__tblr_if_negative_value:n { V } { TF }
```

9.3 Declare and set tabularray keys

```
\cs_new_protected:Npn \__tblr_keys_define:nn #1 #2
  {
    \keys_define:nn { tabularray/#1 } {#2}
  }

\cs_set_eq:NN \DeclareTblrKeys \__tblr_keys_define:nn

\cs_new_protected:Npn \__tblr_keys_set:nn #1 #2
  {
    \keys_set:nn { tabularray/#1 } {#2}
  }
\cs_generate_variant:Nn \__tblr_keys_set:nn { nV, nv, ne }

\cs_set_eq:NN \SetTblrKeys \__tblr_keys_set:nn

\cs_new_protected:Npn \__tblr_keys_set_groups:nnn #1 #2 #3
  {
    \keys_set_groups:nnn { tabularray/#1 } {#2} {#3}
  }

\prg_set_conditional:Npnn \__tblr_keys_if_exist:nn #1 #2 { p, T, F, TF }
  {
    \keys_if_exist:nnTF { tabularray/#1 } { #2 }
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__tblr_keys_if_exist:nn { nV } { TF }

%% A special key such as cell{i}{j} or note{a} consists of name and args.
%% These functions extracts two components of a special key as string #1.

\seq_new:N \l__tblr_key_split_seq
\str_new:N \l__tblr_key_split_name_str
\tl_new:N \l__tblr_key_split_args_tl

\cs_new_protected:Npn \__tblr_key_split_name:n #1
  {
    \seq_set_split_keep_spaces:Nen \l__tblr_key_split_seq \c_left_brace_str {#1}
    \seq_pop_left:NN \l__tblr_key_split_seq \l__tblr_key_split_name_str
    %\str_log:N \l__tblr_key_split_name_str
  }
\cs_generate_variant:Nn \__tblr_key_split_name:n { e }
```

```
\cs_new_protected:Npn \__tblr_key_split_name_args:n #1
  {
    \__tblr_key_split_name:n {#1}
    \seq_if_empty:NTF \l__tblr_key_split_seq
      { \tl_clear:N \l__tblr_key_split_args_tl }
      {
        \tl_set_rescan:Nne \l__tblr_key_split_args_tl {}
          {
            \c_left_brace_str
            \seq_use:Ne \l__tblr_key_split_seq { \c_left_brace_str }
          }
      }
  }

%% To distinguish between modern table specs and traditional column specs,
%% we need to extract first key name from tl #1 which is an argument of tblr.

\cs_new_protected:Npn \__tblr_keyval_extract_first_name:n #1
  {
    \seq_set_split:Nnn \l__tblr_key_split_seq {,} {#1}
    \seq_set_split:Nne \l__tblr_key_split_seq {=}
      { \seq_item:Nn \l__tblr_key_split_seq {1} }
    \__tblr_key_split_name:e
      { \tl_to_str:e { \seq_item:Nn \l__tblr_key_split_seq {1} } }
  }
```

9.4 Create and use tabularray hooks

```
\cs_new_protected:Npn \__tblr_hook_new:n #1
  {
    \hook_new:n { tabularray/#1 }
  }

\cs_new_protected:Npn \__tblr_hook_new_pair:nn #1 #2
  {
    \hook_new_pair:nn { tabularray/#1 } { tabularray/#2 }
  }

\cs_new_protected:Npn \__tblr_hook_use_false:n #1 {}

\cs_new_protected:Npn \__tblr_hook_use_true:n #1
  {
    \hook_use:n { tabularray/#1 }
  }

\cs_set_eq:NN \__tblr_hook_use:n \__tblr_hook_use_false:n

\NewDocumentCommand \AddToTblrHook { m o +m }
  {
    \hook_gput_code:nnn  { tabularray/#1 } { #2 } { #3 }
  }

\NewDocumentCommand \AddToTblrHookNext { m +m }
  {
```

```
    \hook_gput_next_code:nn  { tabularray/#1 } { #2 }
  }
```

9.5 Data structures based on property lists

```
\int_new:N \gTblrLevelInt % store table nesting level

%% \g_tblr_level_int is deprecated and will be removed later
\cs_set_eq:NN \g_tblr_level_int \gTblrLevelInt

\cs_new_protected:Npn \__tblr_clear_prop_lists:
  {
    \__tblr_prop_gclear_new:c
      { g__tblr_text_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_command_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_inner_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_outer_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_note_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_remark_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_more_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_row_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_column_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_cell_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_hline_ \int_use:N \gTblrLevelInt _prop }
    \__tblr_prop_gclear_new:c
      { g__tblr_vline_ \int_use:N \gTblrLevelInt _prop }
  }

\cs_new_protected:Npn \__tblr_prop_gput:nnn #1 #2 #3
  {
    \prop_gput:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop } { #2 } { #3 }
  }
\cs_generate_variant:Nn \__tblr_prop_gput:nnn { nne, nnV, nen, nee, neV }

\cs_new:Npn \__tblr_prop_item:nn #1 #2
  {
    \prop_item:cn { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop } { #2 }
  }
\cs_generate_variant:Nn \__tblr_prop_item:nn { ne }

\cs_new_protected:Npn \__tblr_prop_if_in:nnT #1
  {
    \prop_if_in:cnT { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop }
  }
```

```
\cs_new_protected:Npn \__tblr_prop_if_in:nnF #1
  {
    \prop_if_in:cnF { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop }
  }
\cs_new_protected:Npn \__tblr_prop_if_in:nnTF #1
  {
    \prop_if_in:cnTF { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop }
  }
\prg_generate_conditional_variant:Nnn \__tblr_prop_if_in:nn { ne } { T, F, TF }

\cs_new_protected:Npn \__tblr_prop_log:n #1
  {
    \prop_log:c { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop }
  }

\cs_new_protected:Npn \__tblr_prop_map_inline:nn #1 #2
  {
    \prop_map_inline:cn { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop } {#2}
  }

\cs_new_protected:Npn \__tblr_prop_gput_if_larger:nnn #1 #2 #3
  {
    \__tblr_gput_if_larger:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop } { #2 } { #3 }
  }
\cs_generate_variant:Nn \__tblr_prop_gput_if_larger:nnn { nnx, nnV, nxn, nxx, nxV }

\cs_new_protected:Npn \__tblr_prop_gadd_dimen_value:nnn #1 #2 #3
  {
    \__tblr_gadd_dimen_value:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _prop } { #2 } { #3 }
  }
\cs_generate_variant:Nn \__tblr_prop_gadd_dimen_value:nnn { nnx, nnV, nxn, nxx }

%% Put the dimension to the prop list only if it's larger than the old one

\tl_new:N \l__tblr_put_if_larger_tl

\cs_new_protected:Npn \__tblr_put_if_larger:Nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_put_if_larger_tl { \prop_item:Nn #1 { #2 } }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l__tblr_put_if_larger_tl }
      { \dim_compare_p:nNn { #3 } > { \l__tblr_put_if_larger_tl } }
      { \prop_put:Nnn #1 { #2 } { #3 }  }
  }
\cs_generate_variant:Nn \__tblr_put_if_larger:Nnn { Nee, NnV }

\cs_new_protected:Npn \__tblr_gput_if_larger:Nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_put_if_larger_tl { \prop_item:Nn #1 { #2 } }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l__tblr_put_if_larger_tl }
      { \dim_compare_p:nNn { #3 } > { \l__tblr_put_if_larger_tl } }
      { \prop_gput:Nnn #1 { #2 } { #3 }  }
  }
```

```
\cs_generate_variant:Nn \__tblr_gput_if_larger:Nnn { cnn }

%% Add the dimension to some key value of the prop list
%% #1: the prop list, #2: the key, #3: the dimen to add

\cs_new_protected:Npn \__tblr_add_dimen_value:Nnn #1 #2 #3
  {
    \prop_put:Nne #1 { #2 } { \dim_eval:n { \prop_item:Nn #1 { #2 } + #3 } }
  }
\cs_generate_variant:Nn \__tblr_add_dimen_value:Nnn { cnn }

\cs_new_protected:Npn \__tblr_gadd_dimen_value:Nnn #1 #2 #3
  {
    \prop_gput:Nne #1 { #2 } { \dim_eval:n { \prop_item:Nn #1 { #2 } + #3 } }
  }
\cs_generate_variant:Nn \__tblr_gadd_dimen_value:Nnn { cnn }
```

9.6 Data structures based on token lists

```
\cs_new_protected:Npn \__tblr_clear_spec_lists:
  {
    %\__tblr_clear_one_spec_lists:n { row }
    %\__tblr_clear_one_spec_lists:n { column }
    %\__tblr_clear_one_spec_lists:n { cell }
    \__tblr_clear_one_spec_lists:n { text }
    \__tblr_clear_one_spec_lists:n { hline }
    \__tblr_clear_one_spec_lists:n { vline }
    \__tblr_clear_one_spec_lists:n { outer }
  }

\cs_new_protected:Npn \__tblr_clear_one_spec_lists:n #1
  {
    \clist_if_exist:cTF { g__tblr_#1_ \int_use:N \gTblrLevelInt _clist }
      {
        \clist_map_inline:cn { g__tblr_#1_ \int_use:N \gTblrLevelInt _clist }
          {
            \tl_gclear_new:c { g__tblr_spec_ \int_use:N \gTblrLevelInt _#1_##1_tl }
          }
      }
      { \clist_new:c { g__tblr_#1_ \int_use:N \gTblrLevelInt _clist } }
  }

\cs_new_protected:Npn \__tblr_spec_gput:nnn #1 #2 #3
  {
    \tl_if_exist:cF { g__tblr_spec_ \int_use:N \gTblrLevelInt _#1_#2_tl }
      { \tl_new:c { g__tblr_spec_ \int_use:N \gTblrLevelInt _#1_#2_tl } }
    \tl_gset:cn
      { g__tblr_spec_ \int_use:N \gTblrLevelInt _#1_#2_tl } {#3}
    \clist_gput_right:ce { g__tblr_#1_ \int_use:N \gTblrLevelInt _clist } {#2}
  }
\cs_generate_variant:Nn \__tblr_spec_gput:nnn { nne, nnV, nen, nee, neV }

\cs_new:Npn \__tblr_spec_item:nn #1 #2
  {
    \tl_if_exist:cT { g__tblr_spec_ \int_use:N \gTblrLevelInt _#1_#2_tl }
```

```
      {
        \exp_args:Nv \exp_not:n
          { g__tblr_spec_ \int_use:N \gTblrLevelInt _#1_#2_tl }
      }
  }
\cs_generate_variant:Nn \__tblr_spec_item:nn { ne }

\cs_new_protected:Npn \__tblr_spec_gput_if_larger:nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_put_if_larger_tl { \__tblr_spec_item:nn {#1} {#2} }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l__tblr_put_if_larger_tl }
      { \dim_compare_p:nNn {#3} > { \l__tblr_put_if_larger_tl } }
      { \__tblr_spec_gput:nnn {#1} {#2} {#3} }
  }
\cs_generate_variant:Nn \__tblr_spec_gput_if_larger:nnn { nne, nnV, nen, nee, neV }

\cs_new_protected:Npn \__tblr_spec_gadd_dimen_value:nnn #1 #2 #3
  {
    \__tblr_spec_gput:nne {#1} {#2}
      { \dim_eval:n { \__tblr_spec_item:ne {#1} {#2} + #3 } }
  }
\cs_generate_variant:Nn \__tblr_spec_gadd_dimen_value:nnn { nne, nnV, nen, nee }

\cs_new_protected:Npn \__tblr_spec_log:n #1
  {
    \clist_gremove_duplicates:c
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _clist }
    \tl_log:e
      {
        The ~ spec ~ list ~ #1 _ \int_use:N \gTblrLevelInt
              \space contains ~ the ~ pairs:
      }
    \clist_map_inline:cn { g__tblr_#1_ \int_use:N \gTblrLevelInt _clist }
      {
        \tl_log:e
          {
            \space { ##1 } ~\space=>~\space { \__tblr_spec_item:nn {#1} {##1} }
          }
      }
  }
```

9.7 Data structures based on integer arrays

```
\msg_new:nnn { tabularray } { intarray-beyond-bound }
  { Position ~ #2 ~ is ~ beyond ~ the ~ bound ~ of ~ intarray ~ #1.}

\cs_new_protected:Npn \__tblr_intarray_gset:Nnn #1 #2 #3
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn {#2} < {0} }
      { \int_compare_p:nNn {#2} > {\intarray_count:N #1} }
      {
        \bool_if:NT \g__tblr_tracing_intarray_bool
          { \msg_warning:nnnn { tabularray } { intarray-beyond-bound } {#1} {#2} }
      }
```

```
      { \intarray_gset:Nnn #1 {#2} {#3} }
  }
\cs_generate_variant:Nn \__tblr_intarray_gset:Nnn { cnn }

%% #1: data name; #2: key name; #3: value type
\cs_new_protected:Npn \__tblr_data_new_key:nnn #1 #2 #3
  {
    \int_gincr:c { g__tblr_data_#1_key_count_int }
    \tl_const:ce
      {
        c__tblr_data_#1_key_name_
          \int_use:c { g__tblr_data_#1_key_count_int } _tl
      }
      { #2 }
    \tl_const:ce { c__tblr_data_#1_key_number_#2_tl }
      { \int_use:c { g__tblr_data_#1_key_count_int } }
    \tl_const:cn { c__tblr_data_#1_key_type_#2_tl } {#3}
  }

\int_new:N \g__tblr_data_row_key_count_int
\__tblr_data_new_key:nnn { row } { height }      { dim }
\__tblr_data_new_key:nnn { row } { coefficient } { dec }
\__tblr_data_new_key:nnn { row } { abovesep }    { dim }
\__tblr_data_new_key:nnn { row } { belowsep }    { dim }
\__tblr_data_new_key:nnn { row } { @row-height } { dim }
\__tblr_data_new_key:nnn { row } { @row-head }   { dim }
\__tblr_data_new_key:nnn { row } { @row-foot }   { dim }
\__tblr_data_new_key:nnn { row } { @row-upper }  { dim }
\__tblr_data_new_key:nnn { row } { @row-lower }  { dim }

\int_new:N \g__tblr_data_column_key_count_int
\__tblr_data_new_key:nnn { column } { width }       { dim }
\__tblr_data_new_key:nnn { column } { coefficient } { dec }
\__tblr_data_new_key:nnn { column } { leftsep }     { dim }
\__tblr_data_new_key:nnn { column } { rightsep }    { dim }
\__tblr_data_new_key:nnn { column } { @col-width }  { dim }

\int_new:N \g__tblr_data_cell_key_count_int
\__tblr_data_new_key:nnn { cell } { width }        { dim }
\__tblr_data_new_key:nnn { cell } { rowspan }      { int }
\__tblr_data_new_key:nnn { cell } { colspan }      { int }
\__tblr_data_new_key:nnn { cell } { halign }       { str }
\__tblr_data_new_key:nnn { cell } { valign }       { str }
\__tblr_data_new_key:nnn { cell } { background }   { str }
\__tblr_data_new_key:nnn { cell } { foreground }   { str }
\__tblr_data_new_key:nnn { cell } { font }         { str }
\__tblr_data_new_key:nnn { cell } { mode }         { str }
\__tblr_data_new_key:nnn { cell } { cmd }          { str }
\__tblr_data_new_key:nnn { cell } { omit }         { int }
\__tblr_data_new_key:nnn { cell } { @cell-width }  { dim }
\__tblr_data_new_key:nnn { cell } { @cell-height } { dim }
\__tblr_data_new_key:nnn { cell } { @cell-depth }  { dim }

\clist_const:Nn \c__tblr_data_clist { row, column, cell }
\tl_const:Nn \c__tblr_data_row_count_tl { \c@rowcount }
\tl_const:Nn \c__tblr_data_column_count_tl { \c@colcount }
\tl_const:Nn \c__tblr_data_cell_count_tl { \c@rowcount * \c@colcount }
```

```
\tl_const:Nn \c__tblr_data_row_index_number_tl {1}
\tl_const:Nn \c__tblr_data_column_index_number_tl {1}
\tl_const:Nn \c__tblr_data_cell_index_number_tl {2}
\int_new:N \g__tblr_array_int

\cs_new_protected:Npn \__tblr_init_table_data:
  {
    \clist_map_function:NN \c__tblr_data_clist \__tblr_init_one_data:n
  }

\cs_new_protected:Npn \__tblr_init_one_data:n #1
  {
    \int_gincr:N \g__tblr_array_int
    \intarray_new:cn { g__tblr_#1_ \int_use:N \g__tblr_array_int _intarray }
      {
        \int_use:c { g__tblr_data_#1_key_count_int }
          * \tl_use:c { c__tblr_data_#1_count_tl }
      }
    \cs_set_eq:cc { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
      { g__tblr_#1_ \int_use:N \g__tblr_array_int _intarray }
    %\intarray_log:c { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
  }

%% #1: data name; #2: data index; #3: key name
\cs_new:Npn \__tblr_data_key_to_int:nnn #1 #2 #3
  {
    ( #2 - 1 ) * \int_use:c { g__tblr_data_#1_key_count_int }
      + \tl_use:c { c__tblr_data_#1_key_number_#3_tl }
  }

%% #1: data name; #2: data index 1; #3: data index 2; #4: key name
\cs_new:Npn \__tblr_data_key_to_int:nnnn #1 #2 #3 #4
  {
    ( #2 - 1 ) * \c@colcount * \int_use:c { g__tblr_data_#1_key_count_int }
      + ( #3 - 1 ) * \int_use:c { g__tblr_data_#1_key_count_int }
      + \tl_use:c { c__tblr_data_#1_key_number_#4_tl }
  }

\int_new:N \l__tblr_key_count_int
\int_new:N \l__tblr_key_quotient_int
\int_new:N \l__tblr_key_quotient_two_int
\int_new:N \l__tblr_key_remainder_int

%% #1: data name; #2: array position;
%% #3: returning tl with index; #4: returning tl with key name
\cs_new:Npn \__tblr_data_int_to_key:nnNN #1 #2 #3 #4
  {
    \int_set_eq:Nc \l__tblr_key_count_int { g__tblr_data_#1_key_count_int }
    \int_set:Nn \l__tblr_key_quotient_int
      {
        \int_div_truncate:nn
          { #2 + \l__tblr_key_count_int - 1 } { \l__tblr_key_count_int }
      }
    \int_set:Nn \l__tblr_key_remainder_int
      {
        #2 + \l__tblr_key_count_int
```

```
          - \l__tblr_key_quotient_int * \l__tblr_key_count_int
      }
    \int_compare:nNnT { \l__tblr_key_remainder_int } = { 0 }
      { \int_set_eq:NN \l__tblr_key_remainder_int \l__tblr_key_count_int }
    \tl_set:Ne #3 { \int_use:N \l__tblr_key_quotient_int }
    \tl_set_eq:Nc #4
      { c__tblr_data_#1_key_name_ \int_use:N \l__tblr_key_remainder_int _tl }
  }

%% #1: data name; #2: array position;
%% #3: returning tl with index 1; #4: returning tl with index 2;
%% #5: returning tl with key name
\cs_new:Npn \__tblr_data_int_to_key:nnNNN #1 #2 #3 #4 #5
  {
    \int_set_eq:Nc \l__tblr_key_count_int { g__tblr_data_#1_key_count_int }
    \int_set:Nn \l__tblr_key_quotient_int
      {
        \int_div_truncate:nn
          { #2 + \l__tblr_key_count_int - 1 } { \l__tblr_key_count_int }
      }
    \int_set:Nn \l__tblr_key_remainder_int
      {
        #2 + \l__tblr_key_count_int
          - \l__tblr_key_quotient_int * \l__tblr_key_count_int
      }
    \int_compare:nNnT { \l__tblr_key_remainder_int } = { 0 }
      { \int_set_eq:NN \l__tblr_key_remainder_int \l__tblr_key_count_int }
    \tl_set_eq:Nc #5
      { c__tblr_data_#1_key_name_ \int_use:N \l__tblr_key_remainder_int _tl }
    \int_set:Nn \l__tblr_key_quotient_two_int
      {
        \int_div_truncate:nn
          { \l__tblr_key_quotient_int + \c@colcount - 1 } { \c@colcount }
      }
    \int_set:Nn \l__tblr_key_remainder_int
      {
        \l__tblr_key_quotient_int + \c@colcount
          - \l__tblr_key_quotient_two_int * \c@colcount
      }
    \int_compare:nNnT { \l__tblr_key_remainder_int } = { 0 }
      { \int_set_eq:NN \l__tblr_key_remainder_int \c@colcount }
    \tl_set:Ne #4 { \int_use:N \l__tblr_key_remainder_int }
    \tl_set:Ne #3 { \int_use:N \l__tblr_key_quotient_two_int }
  }

\tl_new:N \g__tblr_data_int_from_value_tl

%% #1: data name; #2: key name; #3: value
%% The result will be stored in \g__tblr_data_int_from_value_tl
\cs_new_protected:Npn \__tblr_data_int_from_value:nnn #1 #2 #3
  {
    \cs:w
      __tblr_data_int_from_ \tl_use:c { c__tblr_data_#1_key_type_#2_tl } :n
    \cs_end:
    {#3}
  }
```

```
%% #1: data name; #2: key name; #3: int
\cs_new:Npn \__tblr_data_int_to_value:nnn #1 #2 #3
  {
    \cs:w
      __tblr_data_int_to_ \tl_use:c { c__tblr_data_#1_key_type_#2_tl } :n
    \cs_end:
    {#3}
  }
\cs_generate_variant:Nn \__tblr_data_int_to_value:nnn { nne, nVe }

\cs_new_protected:Npn \__tblr_data_int_from_int:n #1
  {
    \tl_gset:Nn \g__tblr_data_int_from_value_tl {#1}
  }

\cs_new:Npn \__tblr_data_int_to_int:n #1
  {
    #1
  }

\cs_new_protected:Npn \__tblr_data_int_from_dim:n #1
  {
    \tl_gset:Ne \g__tblr_data_int_from_value_tl { \dim_to_decimal_in_sp:n {#1} }
  }

%% Return a dimension in pt so that it's easier to understand in tracing messages
\cs_new:Npn \__tblr_data_int_to_dim:n #1
  {
    %#1 sp
    %\dim_eval:n { #1 sp }
    \dim_to_decimal:n { #1 sp } pt
  }

\cs_new_protected:Npn \__tblr_data_int_from_dec:n #1
  {
    \tl_gset:Ne \g__tblr_data_int_from_value_tl
      { \dim_to_decimal_in_sp:n {#1 pt} }
  }

\cs_new:Npn \__tblr_data_int_to_dec:n #1
  {
    \dim_to_decimal:n {#1 sp}
  }

\int_new:N \g__tblr_data_str_value_count_int
\tl_gclear_new:c { g__tblr_data_0_to_str_tl }

\cs_new_protected:Npn \__tblr_data_int_from_str:n #1
  {
    \tl_if_exist:cTF { g__tblr_data_ \tl_to_str:n {#1} _to_int_tl }
      {
        \tl_gset_eq:Nc \g__tblr_data_int_from_value_tl
          { g__tblr_data_ \tl_to_str:n {#1} _to_int_tl }
      }
      {
        \int_gincr:N \g__tblr_data_str_value_count_int
```

```
        \tl_new:c { g__tblr_data_ \tl_to_str:n {#1} _to_int_tl }
        \tl_gset:ce { g__tblr_data_ \tl_to_str:n {#1} _to_int_tl }
          { \int_use:N \g__tblr_data_str_value_count_int }
        \tl_new:c
          { g__tblr_data_ \int_use:N \g__tblr_data_str_value_count_int _to_str_tl }
        \tl_gset:cn
          { g__tblr_data_ \int_use:N \g__tblr_data_str_value_count_int _to_str_tl }
          { \exp_not:n {#1} }
        \tl_gset:Ne \g__tblr_data_int_from_value_tl
          { \int_use:N \g__tblr_data_str_value_count_int }
      }
  }

\cs_new:Npn \__tblr_data_int_to_str:n #1
  {
    \tl_use:c { g__tblr_data_#1_to_str_tl }
  }

%% #1: data name; #2: data index; #3: key; #4: value
\cs_new_protected:Npn \__tblr_data_gput:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_from_value:nnn {#1} {#3} {#4}
    \__tblr_intarray_gset:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
      { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gput:nnnn
  { nnne, nnnV, nenn, nene, nenV, nVnn }

%% #1: data name; #2: data index 1; #3: data index 2; #4: key; #5: value
\cs_new_protected:Npn \__tblr_data_gput:nnnnn #1 #2 #3 #4 #5
  {
    \__tblr_data_int_from_value:nnn {#1} {#4} {#5}
    \__tblr_intarray_gset:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
      { \__tblr_data_key_to_int:nnnn {#1} {#2} {#3} {#4} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gput:nnnnn
  { nnnne, nnnnV, neenn, neene, neenV, neeen, nVVnn }

%% #1: data name; #2: data index; #3: key
\cs_new:Npn \__tblr_data_item:nnn #1 #2 #3
  {
    \__tblr_data_int_to_value:nne {#1} {#3}
      {
        \intarray_item:cn { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
          { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      }
  }
\cs_generate_variant:Nn \__tblr_data_item:nnn { nen }

%% #1: data name; #2: data index 1; #3: data index 2; #4: key
\cs_new:Npn \__tblr_data_item:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_to_value:nne {#1} {#4}
```

```
      {
        \intarray_item:cn { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
          { \__tblr_data_key_to_int:nnnn {#1} {#2} {#3} {#4} }
      }
  }
\cs_generate_variant:Nn \__tblr_data_item:nnnn { neen }

\tl_new:N \l__tblr_data_key_tl
\tl_new:N \l__tblr_data_index_tl
\tl_new:N \l__tblr_data_index_two_tl

\cs_new_protected:Npn \__tblr_data_log:n #1
  {
    \use:c { __tblr_data_log_ \use:c { c__tblr_data_#1_index_number_tl } :n } {#1}
    \__tblr_prop_log:n {#1}
  }

\cs_new_protected:cpn { __tblr_data_log_1:n } #1
  {
    %\intarray_log:c { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
    \tl_set:Ne \l_tmpa_tl { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
    \tl_log:n { ----------~----------~----------~----------~---------- }
    \int_step_inline:nn
      { \intarray_count:c { \l_tmpa_tl } }
      {
        \__tblr_data_int_to_key:nnNN {#1} {##1}
          \l__tblr_data_index_tl \l__tblr_data_key_tl
        \tl_log:e
          {
            \space
            { #1 [\l__tblr_data_index_tl] / \l__tblr_data_key_tl }
            ~\space => ~\space
            {
              \__tblr_data_int_to_value:nVe {#1} \l__tblr_data_key_tl
                { \intarray_item:cn { \l_tmpa_tl } {##1} }
            }
          }
      }
  }

\cs_new_protected:cpn { __tblr_data_log_2:n } #1
  {
    %\intarray_log:c { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
    \tl_set:Ne \l_tmpa_tl { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
    \tl_log:n { ----------~----------~----------~----------~---------- }
    \int_step_inline:nn
      { \intarray_count:c { \l_tmpa_tl } }
      {
        \__tblr_data_int_to_key:nnNNN {#1} {##1}
          \l__tblr_data_index_tl \l__tblr_data_index_two_tl \l__tblr_data_key_tl
        \tl_log:e
          {
            \space
            {
              #1 [\l__tblr_data_index_tl][\l__tblr_data_index_two_tl]
                 / \l__tblr_data_key_tl
            }
```

```
            ~\space => ~\space
            {
              \__tblr_data_int_to_value:nVe {#1} \l__tblr_data_key_tl
                { \intarray_item:cn { \l_tmpa_tl } {##1} }
            }
          }
      }
  }

%% #1: data name; #2: row index; #3: key; #4: value
\cs_new_protected:Npn \__tblr_data_gput_if_larger:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_from_value:nnn {#1} {#3} {#4}
    \__tblr_array_gput_if_larger:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
      { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gput_if_larger:nnnn { nnne, nnnV, nene, nenV }

\cs_new_protected:Npn \__tblr_array_gput_if_larger:Nnn #1 #2 #3
  {
    \int_compare:nNnT {#3} > { \intarray_item:Nn #1 {#2} }
      { \__tblr_intarray_gset:Nnn #1 {#2} {#3} }
  }
\cs_generate_variant:Nn \__tblr_array_gput_if_larger:Nnn { cnn }

%% #1: data name; #2: data index; #3: key; #4: value
\cs_new_protected:Npn \__tblr_data_gadd_dimen_value:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_from_value:nnn {#1} {#3} {#4}
    \__tblr_array_gadd_value:cnn
      { g__tblr_#1_ \int_use:N \gTblrLevelInt _intarray }
      { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gadd_dimen_value:nnnn
  { nnne, nnnV, nenn, nene }

\cs_new_protected:Npn \__tblr_array_gadd_value:Nnn #1 #2 #3
  {
    \__tblr_intarray_gset:Nnn #1 {#2} { \intarray_item:Nn #1 {#2} + #3 }
  }
\cs_generate_variant:Nn \__tblr_array_gadd_value:Nnn { cnn }
```

9.8 Switch between different data structures

```
\cs_set_eq:NN \__tblr_prop_new:N \prop_new:N
\cs_set_eq:NN \__tblr_prop_gclear_new:N \prop_gclear_new:N
\cs_generate_variant:Nn \__tblr_prop_gclear_new:N { c }

\bool_new:N \g__tblr_use_intarray_bool
\bool_gset_true:N \g__tblr_use_intarray_bool

\cs_new_protected:Npn \__tblr_map_data_to_spec:
```

```
  {
    \cs_set_protected:Npn \__tblr_data_gput:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_spec_gput:nnn {##1} { [##2] / ##3 } {##4}
      }
    \cs_set_protected:Npn \__tblr_data_gput:nnnnn ##1 ##2 ##3 ##4 ##5
      {
        \__tblr_spec_gput:nnn {##1} { [##2][##3] / ##4 } {##5}
      }
    \cs_set:Npn \__tblr_data_item:nnn ##1 ##2 ##3
      {
        \__tblr_spec_item:nn {##1} { [##2] / ##3 }
      }
    \cs_set:Npn \__tblr_data_item:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_spec_item:nn {##1} { [##2][##3] / ##4 }
      }
    \cs_set_protected:Npn \__tblr_data_log:n ##1
      {
        \__tblr_spec_log:n {##1}
      }
    \cs_set_protected:Npn \__tblr_data_gput_if_larger:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_spec_gput_if_larger:nnn {##1} { [##2] / ##3 } {##4}
      }
    \cs_set_protected:Npn \__tblr_data_gput_if_larger:nnnnn ##1 ##2 ##3 ##4 ##5
      {
        \__tblr_spec_gput_if_larger:nnn {##1} { [##2][##3] / ##4 } {##5}
      }
    \cs_set_protected:Npn \__tblr_data_gadd_dimen_value:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_spec_gadd_dimen_value:nnn {##1} { [##2] / ##3 } {##4}
      }
    \cs_set_protected:Npn \__tblr_data_gadd_dimen_value:nnnnn ##1 ##2 ##3 ##4 ##5
      {
        \__tblr_spec_gadd_dimen_value:nnn {##1} { [##2][##3] / ##4 } {##5}
      }
  }

\bool_new:N \g__tblr_use_linked_prop_bool

\cs_new_protected:Npn \__tblr_map_data_to_prop:
  {
    \cs_set_protected:Npn \__tblr_data_gput:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_prop_gput:nnn {##1} { [##2] / ##3 } {##4}
      }
    \cs_set_protected:Npn \__tblr_data_gput:nnnnn ##1 ##2 ##3 ##4 ##5
      {
        \__tblr_prop_gput:nnn {##1} { [##2][##3] / ##4 } {##5}
      }
    \cs_set:Npn \__tblr_data_item:nnn ##1 ##2 ##3
      {
        % Be careful not to add \tl_log in an expandable function
        \__tblr_prop_item:nn {##1} { [##2] / ##3 }
      }
    \cs_set:Npn \__tblr_data_item:nnnn ##1 ##2 ##3 ##4
      {
```

```
        % Be careful not to add \tl_log in an expandable function
        \__tblr_prop_item:nn {##1} { [##2][##3] / ##4 }
      }
    \cs_set_protected:Npn \__tblr_data_log:n ##1
      {
        \__tblr_prop_log:n {##1}
      }
    \cs_set_protected:Npn \__tblr_data_gput_if_larger:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_prop_gput_if_larger:nnn {##1} { [##2] / ##3 } {##4}
      }
    \cs_set_protected:Npn \__tblr_data_gput_if_larger:nnnnn ##1 ##2 ##3 ##4 ##5
      {
        \__tblr_prop_gput_if_larger:nnn {##1} { [##2][##3] / ##4 } {##5}
      }
    \cs_set_protected:Npn \__tblr_data_gadd_dimen_value:nnnn ##1 ##2 ##3 ##4
      {
        \__tblr_prop_gadd_dimen_value:nnn {##1} { [##2] / ##3 } {##4}
      }
    \cs_set_protected:Npn \__tblr_data_gadd_dimen_value:nnnnn ##1 ##2 ##3 ##4 ##5
      {
        \__tblr_prop_gadd_dimen_value:nnn {##1} { [##2][##3] / ##4 } {##5}
      }
  }

\cs_new_protected:Npn \__tblr_map_spec_to_prop:
  {
    \cs_set_eq:NN \__tblr_spec_gput:nnn \__tblr_prop_gput:nnn
    \cs_set_eq:NN \__tblr_spec_item:nn \__tblr_prop_item:nn
    \cs_set_eq:NN \__tblr_spec_log:n \__tblr_prop_log:n
    \cs_set_eq:NN
      \__tblr_spec_gput_if_larger:nnn
      \__tblr_prop_gput_if_larger:nnn
    \cs_set_eq:NN
      \__tblr_spec_gadd_dimen_value:nnn
      \__tblr_prop_gadd_dimen_value:nnn
  }

%% Backport fix for https://github.com/latex3/latex3/issues/1630
\cs_new_protected:Npn \__tblr_backport_prop_item_fix:
  {
    \cs_set:Npn \prop_item:Nn ##1 ##2
      {
        \__prop_if_flat:NTF ##1
          {
            \exp_args:NNo \prop_map_tokens:Nn ##1
              {
                \exp_after:wN \__prop_item:nnn
                \exp_after:wN { \tl_to_str:n {##2} }
              }
          }
          { \exp_after:wN \__prop_get_linked:w ##1 {##2} \exp_not:n { } { } }
      }
  }

%% We can't use \IfExplAtLeastTF since it was added in LaTeX release 2023-11-01.
\@ifl@t@r \ExplLoaderFileDate { 2024-02-18 }
```

```
  { \bool_gset_true:N \g__tblr_use_linked_prop_bool } { }

\AtBeginDocument
  {
    \bool_if:NTF \g__tblr_use_linked_prop_bool
      {
        \__tblr_map_spec_to_prop:
        \__tblr_map_data_to_prop:
        \cs_set_eq:NN \__tblr_prop_new:N \prop_new_linked:N
        \cs_set_eq:NN \__tblr_prop_gclear_new:N \prop_gclear_new_linked:N
        \@ifl@t@r \ExplLoaderFileDate { 2024-12-09 }
          { } { \__tblr_backport_prop_item_fix: }
        \bool_gset_false:N \g__tblr_use_intarray_bool
      }
      { \bool_if:NF \g__tblr_use_intarray_bool { \__tblr_map_data_to_spec: } }
  }
```

9.9 Child indexers and child selectors

```
\clist_new:N \lTblrUsedChildIndexerClist
\clist_new:N \lTblrUsedChildSelectorClist

\msg_new:nnn { tabularray } { invalid-child-indexer }
  { Child ~ indexer ~ name ~ '#1' ~ must ~ start ~ with ~ uppercase ~ letter. }

\msg_new:nnn { tabularray } { invalid-child-selector }
  { Child ~ selector ~ name ~ '#1' ~ must ~ start ~ with ~ lowercase ~ letter. }

\msg_new:nnn { tabularray } { used-child-indexer }
  { Child ~ indexer ~ name ~ '#1' ~ has ~ been ~ used. }

\msg_new:nnn { tabularray } { used-child-selector }
  { Child ~ selector ~ name ~ '#1' ~ has ~ been ~ used. }

\tl_new:N \l__tblr_child_arg_spec_tl

\NewDocumentCommand \NewTblrChildIndexer { m O{0} o m }
  {
    \__tblr_new_child_indexer_aux:ennn { \tl_trim_spaces:n {#1} } {#2} {#3} {#4}
  }

\cs_new_protected:Npn \__tblr_new_child_indexer_aux:nnnn #1 #2 #3 #4
  {
    \__tblr_if_head_upper:nTF { #1 }
      {
        \clist_if_in:NnTF \lTblrUsedChildIndexerClist { #1 }
          {
            \msg_error:nnn { tabularray } { used-child-indexer } { #1 }
            \clist_log:N \lTblrUsedChildIndexerClist
          }
          {
            \__tblr_make_xparse_arg_spec:nnN
              { #2 } { #3 } \l__tblr_child_arg_spec_tl
            \exp_args:NcV \NewDocumentCommand
```

```
              { __tblr_child_indexer_ #1 } \l__tblr_child_arg_spec_tl { #4 }
            \clist_put_right:Nn \lTblrUsedChildIndexerClist { #1 }
          }
      }
      { \msg_error:nnn { tabularray } { invalid-child-indexer } { #1 } }
  }
\cs_generate_variant:Nn \__tblr_new_child_indexer_aux:nnnn { ennn }

\NewDocumentCommand \NewTblrChildSelector { m O{0} o m }
  {
    \__tblr_new_child_selector_aux:ennn { \tl_trim_spaces:n {#1} } {#2} {#3} {#4}
  }
\cs_set_eq:NN \NewChildSelector \NewTblrChildSelector

\cs_new_protected:Npn \__tblr_new_child_selector_aux:nnnn #1 #2 #3 #4
  {
    \__tblr_if_head_lower:nTF { #1 }
      {
        \clist_if_in:NnTF \lTblrUsedChildSelectorClist { #1 }
          {
            \msg_error:nnn { tabularray } { used-child-selector } { #1 }
            \clist_log:N \lTblrUsedChildSelectorClist
          }
          {
            \__tblr_make_xparse_arg_spec:nnN
              { #2 } { #3 } \l__tblr_child_arg_spec_tl
            \exp_args:NcV \NewDocumentCommand
              { __tblr_child_selector_ #1 } \l__tblr_child_arg_spec_tl { #4 }
            \clist_put_right:Nn \lTblrUsedChildSelectorClist { #1 }
          }
      }
      { \msg_error:nnn { tabularray } { invalid-child-selector } { #1 } }
  }
\cs_generate_variant:Nn \__tblr_new_child_selector_aux:nnnn { ennn }

%% #1: argument number, #2: optional argument default, #3: result tl
\cs_new_protected:Npn \__tblr_make_xparse_arg_spec:nnN #1 #2 #3
  {
    \tl_clear:N #3
    \int_compare:nNnT { #1 } > { 0 }
      {
        \IfValueTF { #2 }
          { \tl_set:Nn #3 { O{#2} } }
          { \tl_set:Nn #3 { m } }
        \tl_put_right:Ne #3 { \prg_replicate:nn { #1 - 1 } { m } }
      }
  }

\int_new:N \lTblrChildTotalInt
\int_new:N \lTblrChildHtotalInt
\int_new:N \lTblrChildVtotalInt
\tl_new:N \lTblrChildIndexTl % may be <i> or {<i>}{<j>}
\clist_new:N \lTblrChildClist

\NewTblrChildIndexer {U}
  {
    \tl_set:Ne \lTblrChildIndexTl
```

```
      { \int_eval:n { \int_max:nn { \lTblrChildTotalInt - 5 } { 1 } } }
  }
\NewTblrChildIndexer {V}
  {
    \tl_set:Ne \lTblrChildIndexTl
      { \int_eval:n { \int_max:nn { \lTblrChildTotalInt - 4 } { 1 } } }
  }
\NewTblrChildIndexer {W}
  {
    \tl_set:Ne \lTblrChildIndexTl
      { \int_eval:n { \int_max:nn { \lTblrChildTotalInt - 3 } { 1 } } }
  }
\NewTblrChildIndexer {X}
  {
    \tl_set:Ne \lTblrChildIndexTl
      { \int_eval:n { \int_max:nn { \lTblrChildTotalInt - 2 } { 1 } } }
  }
\NewTblrChildIndexer {Y}
  {
    \tl_set:Ne \lTblrChildIndexTl
      { \int_eval:n { \int_max:nn { \lTblrChildTotalInt - 1 } { 1 } } }
  }
\NewTblrChildIndexer {Z} [1] [1]
  {
    \tl_set:Ne \lTblrChildIndexTl
      { \int_eval:n { \int_max:nn { \lTblrChildTotalInt + 1 - #1 } { 1 } } }
  }


\NewTblrChildSelector { odd } [1] [1]
  {
    \int_if_odd:nTF {#1}
      {
        \clist_set:Ne \lTblrChildClist
          { {#1} {2} {\int_use:N \lTblrChildTotalInt} }
      }
      {
        \clist_set:Ne \lTblrChildClist
          { {\int_eval:n {#1 + 1}} {2} {\int_use:N \lTblrChildTotalInt} }
      }
  }


\NewTblrChildSelector { even } [1] [2]
  {
    \int_if_even:nTF {#1}
      {
        \clist_set:Ne \lTblrChildClist
          { {#1} {2} {\int_use:N \lTblrChildTotalInt} }
      }
      {
        \clist_set:Ne \lTblrChildClist
          { {\int_eval:n {#1 + 1 }} {2} {\int_use:N \lTblrChildTotalInt} }
      }
  }

%% #1: step; #2: start index; #3: end index
\NewTblrChildSelector { every } [3] [1]
  {
```

```
    \clist_set:Ne \lTblrChildClist
      {
        {
          \int_compare:nNnTF {#2} < {0}
            { \int_eval:n { \lTblrChildTotalInt + 1 #2 } } {#2}
        }
        { #1 }
        {
          \int_compare:nNnTF {#3} < {0}
            { \int_eval:n { \lTblrChildTotalInt + 1 #3 } } {#3}
        }
      }
  }

\clist_new:N \l__tblr_child_whole_clist
\seq_new:N \l__child_spec_seq
\tl_new:N \l__tblr_child_item_head_tl

%% #1, tl of child specifications; #2, total number of children.
%% The result will be put into \l__tblr_child_whole_clist
\cs_new_protected:Npn \__tblr_child_parse:nn #1 #2
  {
    \clist_clear:N \l__tblr_child_whole_clist
    \seq_set_split:Nnn \l__child_spec_seq {,} {#1}
    \seq_map_inline:Nn \l__child_spec_seq
      {
        \clist_clear:N \lTblrChildClist
        \int_set:Nn \lTblrChildTotalInt {#2}
        \tl_set:Ne \l__tblr_child_item_head_tl { \tl_head:n {##1} }
        \tl_if_eq:NnTF \l__tblr_child_item_head_tl {-}
          { \clist_set:Ne \lTblrChildClist { {1} {1} {#2} } }
          {
            \__tblr_tl_if_lower:VTF \l__tblr_child_item_head_tl
              {
                \__tblr_child_run_selector:n {##1}
              }
              {
                \__tblr_tl_if_upper:VTF \l__tblr_child_item_head_tl
                  {
                    \__tblr_child_split_item:n {##1}
                    \__tblr_child_run_indexer_from:
                    \__tblr_child_parse_indexer_to:N \__tblr_child_set_clist:
                  }
                  {
                    \__tblr_tl_if_digit:VTF \l__tblr_child_item_head_tl
                      {
                        \__tblr_child_split_item:n {##1}
                        \__tblr_child_parse_indexer_to:N \__tblr_child_set_clist:
                      }
                      {
                        % error
                      }
                  }
              }
          }
        \clist_put_right:NV \l__tblr_child_whole_clist \lTblrChildClist
      }
    %\clist_log:N \l__tblr_child_whole_clist
```

```
  }
\cs_generate_variant:Nn \__tblr_child_parse:nn { ne }

\cs_new_protected:Npn \__tblr_child_run_selector:n #1
  {
    \tl_rescan:ne { \cctab_select:N \c_code_cctab }
      { \c_backslash_str __tblr_child_selector_ #1 }
    \scan_stop: % for selectors with only optional arguments
  }

\cs_new_protected:Npn \__tblr_child_run_indexer:n #1
  {
    \tl_rescan:ne { \cctab_select:N \c_code_cctab }
      { \c_backslash_str __tblr_child_indexer_ #1 }
    \scan_stop: % for indexers with only optional arguments
  }
\cs_generate_variant:Nn \__tblr_child_run_indexer:n { V }

\seq_new:N \l__tblr_child_item_seq
\tl_new:N \l__tblr_child_from_tl
\tl_new:N \l__tblr_child_to_tl

\cs_new_protected:Npn \__tblr_child_split_item:n #1
  {
    \seq_set_split:Nnn  \l__tblr_child_item_seq {-} {#1}
    \tl_set:Ne \l__tblr_child_from_tl
      { \seq_item:Nn \l__tblr_child_item_seq {1} }
    \tl_set:Ne \l__tblr_child_to_tl
      { \seq_item:Nn \l__tblr_child_item_seq {2} }
  }

\cs_new_protected:Npn \__tblr_child_parse_indexer_to:N #1
  {
    \str_if_empty:NTF \l__tblr_child_to_tl
      { \clist_set:NV \lTblrChildClist \l__tblr_child_from_tl }
      {
        \__tblr_tl_if_digit:eF { \tl_head:N \l__tblr_child_to_tl }
          { \__tblr_child_run_indexer_to: }
        #1
      }
  }

\cs_new_protected:Npn \__tblr_child_run_indexer_from:
  {
    \__tblr_child_run_indexer:V \l__tblr_child_from_tl
    \tl_set_eq:NN \l__tblr_child_from_tl \lTblrChildIndexTl
  }

\cs_new_protected:Npn \__tblr_child_run_indexer_to:
  {
    \__tblr_child_run_indexer:V \l__tblr_child_to_tl
    \tl_set_eq:NN \l__tblr_child_to_tl \lTblrChildIndexTl
  }

\cs_new_protected:Npn \__tblr_child_set_clist:
  {
```

```
    \clist_set:Ne \lTblrChildClist
      {
        { \l__tblr_child_from_tl } { 1 } { \l__tblr_child_to_tl }
      }
  }

\clist_new:N \l__tblr_child_tuple_whole_clist

%% #1, tl of child tuple specifications; #2 and #3, total numbers of children.
%% The result will be put into \l__tblr_child_tuple_whole_clist
\cs_new_protected:Npn \__tblr_child_parse_tuple:nnn #1 #2 #3
  {
    \clist_clear:N \l__tblr_child_tuple_whole_clist
    \seq_set_split:Nnn \l__child_spec_seq {,} {#1}
    \seq_map_inline:Nn \l__child_spec_seq
      {
        \clist_clear:N \lTblrChildClist
        \int_set:Nn \lTblrChildHtotalInt {#2}
        \int_set:Nn \lTblrChildVtotalInt {#3}
        \tl_if_head_is_group:nTF {##1}
          {
            \__tblr_child_split_item:n {##1}
            \__tblr_child_parse_indexer_to:N \__tblr_child_set_clist_tuple:
          }
          {
            \tl_set:Ne \l__tblr_child_item_head_tl { \tl_head:n {##1} }
            \__tblr_tl_if_upper:VTF \l__tblr_child_item_head_tl
              {
                \__tblr_child_split_item:n {##1}
                \__tblr_child_run_indexer_from:
                \__tblr_child_parse_indexer_to:N \__tblr_child_set_clist_tuple:
              }
              {
                \__tblr_tl_if_lower:VTF \l__tblr_child_item_head_tl
                  { \__tblr_child_run_selector:n {##1} }
                  {
                    % error
                  }
              }
          }
        \clist_put_right:NV \l__tblr_child_tuple_whole_clist \lTblrChildClist
      }
    %\clist_log:N \l__tblr_child_tuple_whole_clist
  }
\cs_generate_variant:Nn \__tblr_child_parse_tuple:nnn { nee }

\int_new:N \l__child_diff_h_int
\int_new:N \l__child_diff_v_int
\int_new:N \l__child_sign_h_int
\int_new:N \l__child_sign_v_int
\int_new:N \l__child_step_int

\cs_new_protected:Npn \__tblr_child_set_clist_tuple:
  {
    \int_set:Nn \l__child_diff_h_int
      {
          \exp_after:wN \use_i:nn \l__tblr_child_to_tl
```

```
        - \exp_after:wN \use_i:nn \l__tblr_child_from_tl
      }
    \int_set:Nn \l__child_diff_v_int
      {
          \exp_after:wN \use_ii:nn \l__tblr_child_to_tl
        - \exp_after:wN \use_ii:nn \l__tblr_child_from_tl
      }
    \int_set:Nn \l__child_sign_h_int { \int_sign:n { \l__child_diff_h_int } }
    \int_set:Nn \l__child_sign_v_int { \int_sign:n { \l__child_diff_v_int } }
    \int_set:Nn \l__child_step_int
      {
        \int_min:nn
          { \int_abs:n { \l__child_diff_h_int } }
          { \int_abs:n { \l__child_diff_v_int } }
      }
    \int_step_inline:nnn { 0 } { \l__child_step_int }
      {
        \clist_put_right:Ne \lTblrChildClist
          {
            {
              \int_eval:n
                {
                  \exp_after:wN \use_i:nn \l__tblr_child_from_tl
                  + ##1 * \l__child_sign_h_int
                }
            }
            {
              \int_eval:n
                {
                  \exp_after:wN \use_ii:nn \l__tblr_child_from_tl
                  + ##1 * \l__child_sign_v_int
                }
            }
          }
      }
  }

%% Map over child index lists. We only support two level nesting.

\cs_new_protected:Npn \__tblr_child_map_inline:n #1
  {
    \__tblr_child_map_inline_aux:Nn \__tblr_child_map_aux:n {#1}
  }

\cs_new_protected:Npn \__tblr_child_submap_inline:n #1
  {
    \__tblr_child_map_inline_aux:Nn \__tblr_child_submap_aux:n {#1}
  }

%% #1: function for storing the inline code; #2: the inline code.
\cs_new_protected:Npn \__tblr_child_map_inline_aux:Nn #1 #2
  {
    \cs_set_protected:Npn #1 ##1 { #2 }
    \exp_args:NV \clist_map_inline:nn \l__tblr_child_whole_clist
      {
        \tl_if_head_is_group:nTF { ##1 }
          { \int_step_inline:nnnn ##1 { #1 { ####1 } } } { #1 { ##1 } }
```

```
      }
  }

%% Get the first or last index of a child index list (used by 'endpos' key).

\cs_new_protected:Npn \__tblr_child_get_first:N #1
  {
    \tl_set:Ne #1 { \clist_item:Nn \l__tblr_child_whole_clist { 1 } }
    \exp_args:NV \tl_if_head_is_group:nT #1
      { \tl_set:Ne #1 { \exp_after:wN \use_i:nnn #1 } }
  }

\cs_new_protected:Npn \__tblr_child_get_last:N #1
  {
    \tl_set:Ne #1 { \clist_item:Nn \l__tblr_child_whole_clist { -1 } }
    \exp_args:NV \tl_if_head_is_group:nT #1
      { \exp_after:wN \__tblr_child_get_step_last:nnnN #1 #1 }
  }

%% #1: from; #2: step; #3: to; #4: tl var for storing the result
\cs_new_protected:Npn \__tblr_child_get_step_last:nnnN #1 #2 #3 #4
  {
    \tl_set:Ne #4 { \int_div_truncate:nn { #3 - #1 } { #2 } }
    \int_compare:nNnTF { #4 } > { 0 }
      { \tl_set:Ne #4 { \int_eval:n { #1 + #4 * #2 } } }
      { \tl_set:Nn #4 { #1 } }
  }
```

9.10 New table commands

```
%% We need some commands to modify table/row/column/cell specifications.
%% These commands must be defined with \NewTblrTableCommand command,
%% so that we could extract them, execute them once, then disable them.

\clist_new:N \g__tblr_table_commands_clist

\msg_new:nnn { tabularray } { defined-table-command }
  { Table ~ command ~ #1 already ~ defined! }

\NewDocumentCommand \NewTblrTableCommand { m O{0} o m }
  {
    \clist_if_in:NnTF \g__tblr_table_commands_clist { #1 }
      {
        \msg_error:nnn { tabularray } { defined-table-command } { #1 }
        \clist_log:N \g__tblr_table_commands_clist
      }
      {
        \__tblr_make_xparse_arg_spec:nnN { #2 } { #3 } \l__tblr_a_tl
        \exp_args:NcV \NewDocumentCommand
          { __tblr_table_command_ \cs_to_str:N #1 :w } \l__tblr_a_tl { #4 }
        %% we can not use \cs_if_exist:NTF here (see issue #328)
        \__tblr_cs_if_defined:NTF #1
          {
            \cs_set_eq:cN { __tblr_table_command_ \cs_to_str:N #1 _saved:w } #1
          }
```

```
          {
            \exp_args:NcV \NewDocumentCommand
              { __tblr_table_command_ \cs_to_str:N #1 _saved:w } \l__tblr_a_tl { }
          }
        \tl_new:c { g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl }
        \IfValueTF { #3 }
          {
            \tl_gset:cn { g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl } {-#2}
          }
          {
            \tl_gset:cn { g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl } {#2}
          }
        \clist_gput_right:Nn \g__tblr_table_commands_clist { #1 }
      }
  }
\cs_set_eq:NN \NewTableCommand \NewTblrTableCommand

\cs_new_protected:Npn \__tblr_enable_table_commands:
  {
    \clist_map_inline:Nn \g__tblr_table_commands_clist
      { \cs_set_eq:Nc ##1 { __tblr_table_command_ \cs_to_str:N ##1 :w } }
  }

\cs_new_protected:Npn \__tblr_disable_table_commands:
  {
    \clist_map_inline:Nn \g__tblr_table_commands_clist
      { \cs_set_eq:Nc ##1 { __tblr_table_command_ \cs_to_str:N ##1 _saved:w } }
  }

\cs_new_protected:Npn \__tblr_execute_table_commands:
  {
    \__tblr_prop_map_inline:nn { command }
      {
        \__tblr_set_row_col_from_key_name:w ##1
        ##2
      }
    \LogTblrTracing { cell }
  }

\cs_new_protected:Npn \__tblr_set_row_col_from_key_name:w [#1][#2]
  {
    \int_set:Nn \c@rownum {#1}
    \int_set:Nn \c@colnum {#2}
  }

%% Add \empty as a table command so that users can write \\\empty\hline (see #328)
\NewTblrTableCommand\empty{}

%% Table commands are defined only inside tblr environments,
%% but some packages such as csvsimple need to use them outside tblr environments,
%% therefore we define some of them first here.
\ProvideDocumentCommand \SetHlines  { o m m } {}
\ProvideDocumentCommand \SetHline   { o m m } {}
\ProvideDocumentCommand \SetVlines  { o m m } {}
\ProvideDocumentCommand \SetVline   { o m m } {}
\ProvideDocumentCommand \SetCells   { o m } {}
```

```
\ProvideDocumentCommand \SetCell    { o m } {}
\ProvideDocumentCommand \SetRows    { o m } {}
\ProvideDocumentCommand \SetRow     { o m } {}
\ProvideDocumentCommand \SetColumns { o m } {}
\ProvideDocumentCommand \SetColumn  { o m } {}
```

9.11 Child ids and child classes

```
\NewTblrTableCommand \SetChild [1]
  {
    \__tblr_keys_set:nn { child/index } {#1}
  }


\__tblr_keys_define:nn { child/index }
  {
    id  .code:n = \__tblr_child_add_id:n {#1},
    idh .code:n = \__tblr_child_add_idh:n {#1},
    idv .code:n = \__tblr_child_add_idv:n {#1},
    id* .meta:n = { id = #1, idh = #1, idv = #1 },
    class  .code:n = \__tblr_child_add_class:n {#1},
    classh .code:n = \__tblr_child_add_classh:n {#1},
    classv .code:n = \__tblr_child_add_classv:n {#1},
    class* .meta:n = { class = #1, classh = #1, classv = #1 }
  }


\clist_new:N \l__tblr_child_id_clist
\clist_new:N \l__tblr_child_class_clist


\cs_new_protected:Npn \__tblr_child_add_id:n #1
  {
    \clist_if_in:NnF \l__tblr_child_id_clist {#1}
      {
        \clist_put_right:Nn \l__tblr_child_id_clist {#1}
        \tl_clear_new:c { l__tblr_child_id_#1_tl }
      }
    \tl_set:ce { l__tblr_child_id_#1_tl }
      { { \int_use:N \c@rownum } { \int_use:N \c@colnum } }
  }
\cs_new_protected:Npn \__tblr_child_add_idh:n #1
  {
    \clist_if_in:NnF \l__tblr_child_id_clist {#1h}
      {
        \clist_put_right:Nn \l__tblr_child_id_clist {#1h}
        \tl_clear_new:c { l__tblr_child_id_#1h_tl }
      }
    \tl_set:ce { l__tblr_child_id_#1h_tl } { \int_use:N \c@rownum }
  }
\cs_new_protected:Npn \__tblr_child_add_idv:n #1
  {
    \clist_if_in:NnF \l__tblr_child_id_clist {#1v}
      {
        \clist_put_right:Nn \l__tblr_child_id_clist {#1v}
        \tl_clear_new:c { l__tblr_child_id_#1v_tl }
      }
    \tl_set:ce { l__tblr_child_id_#1v_tl } { \int_use:N \c@colnum }
  }
```

```
\cs_new_protected:Npn \__tblr_child_add_class:n #1
  {
    \clist_if_in:NnTF \l__tblr_child_class_clist {#1}
      {
        \clist_put_right:ce { l__tblr_child_class_#1_clist }
          { { \int_use:N \c@rownum } { \int_use:N \c@colnum } }
      }
      {
        \clist_put_right:Nn \l__tblr_child_class_clist {#1}
        \clist_clear_new:c { l__tblr_child_class_#1_clist }
        \clist_set:ce { l__tblr_child_class_#1_clist }
          { { \int_use:N \c@rownum } { \int_use:N \c@colnum } }
      }
  }
\cs_new_protected:Npn \__tblr_child_add_classh:n #1
  {
    \clist_if_in:NnTF \l__tblr_child_class_clist {#1h}
      {
        \clist_put_right:ce { l__tblr_child_class_#1h_clist }
          { \int_use:N \c@rownum }
      }
      {
        \clist_put_right:Nn \l__tblr_child_class_clist {#1h}
        \clist_clear_new:c { l__tblr_child_class_#1h_clist }
        \clist_set:ce { l__tblr_child_class_#1h_clist }
          { \int_use:N \c@rownum }
      }
  }
\cs_new_protected:Npn \__tblr_child_add_classv:n #1
  {
    \clist_if_in:NnTF \l__tblr_child_class_clist {#1v}
      {
        \clist_put_right:ce { l__tblr_child_class_#1v_clist }
          { \int_use:N \c@colnum }
      }
      {
        \clist_put_right:Nn \l__tblr_child_class_clist {#1v}
        \clist_clear_new:c { l__tblr_child_class_#1v_clist }
        \clist_set:ce { l__tblr_child_class_#1v_clist }
          { \int_use:N \c@colnum }
      }
  }


\cs_new_protected:Npn \__tblr_child_split_table_before:
  {
    \clist_clear:N \l__tblr_child_id_clist
    \clist_clear:N \l__tblr_child_class_clist
  }


\cs_new_protected:Npn \__tblr_child_split_table_after:
  {
    \clist_map_inline:Nn \l__tblr_child_id_clist
      {
        \exp_args:Nne
        \NewTblrChildIndexer {##1}
          {
            \tl_set:Nn \exp_not:N \lTblrChildIndexTl
              { \use:c { l__tblr_child_id_##1_tl } }
```

```
          }
      }
    \clist_map_inline:Nn \l__tblr_child_class_clist
      {
        \exp_args:Nne
        \NewTblrChildSelector {##1}
          {
            \clist_set:Nn \exp_not:N \lTblrChildClist
              { \use:c { l__tblr_child_class_##1_clist } }
          }
      }
  }

\cs_new_protected:Npn \__tblr_child_extract_index_command:N #1
  {
    \tl_if_head_eq_meaning:VNT #1 \SetChild
      {
        \tl_set:Ne #1 { \tl_tail:N #1 }
        \__tblr_keys_set:ne { child/index } { \tl_head:N #1 }
        \tl_set:Ne #1 { \tl_tail:N #1 }
      }
  }

\NewExpandableDocumentCommand \ExpTblrChildId {m}
  {
    \use:c { l__tblr_child_id_#1_tl }
  }

\NewExpandableDocumentCommand \ExpTblrChildClass {m}
  {
    \use:c { l__tblr_child_class_#1_clist }
  }
```

9.12 New content commands

```
%% We need to emulate or fix some commands such as \diagbox in other packages
%% These commands must be defined with \NewTblrContentCommand command
%% We only enable them inside tblr environment to avoid potential conflict

\clist_new:N \g__tblr_content_commands_clist

\msg_new:nnn { tabularray } { defined-content-command }
  { Content ~ command ~ #1 already ~ defined! }

\NewDocumentCommand \NewTblrContentCommand { m O{0} o m }
  {
    \clist_if_in:NnTF \g__tblr_content_commands_clist { #1 }
      {
        \msg_error:nnn { tabularray } { defined-content-command } { #1 }
        \clist_log:N \g__tblr_content_commands_clist
      }
      {
        \__tblr_make_xparse_arg_spec:nnN { #2 } { #3 } \l__tblr_a_tl
        \exp_args:NcV \NewDocumentCommand
          { __tblr_content_command_ \cs_to_str:N #1 :w } \l__tblr_a_tl { #4 }
```

```
        \clist_gput_right:Nn \g__tblr_content_commands_clist { #1 }
      }
  }
\cs_set_eq:NN \NewContentCommand \NewTblrContentCommand

\cs_new_protected:Npn \__tblr_enable_content_commands:
  {
    \clist_map_inline:Nn \g__tblr_content_commands_clist
      { \cs_set_eq:Nc ##1 { __tblr_content_command_ \cs_to_str:N ##1 :w } }
  }
```

9.13 New dash styles

```
%% \NewTblrDashStyle commands

\dim_new:N \lTblrDefaultHruleWidthDim
\dim_new:N \lTblrDefaultVruleWidthDim
\dim_set:Nn \lTblrDefaultHruleWidthDim {0.4pt}
\dim_set:Nn \lTblrDefaultVruleWidthDim {0.4pt}

\prop_new:N \g__tblr_defined_hdash_styles_prop
\prop_new:N \g__tblr_defined_vdash_styles_prop

\prop_gset_from_keyval:Nn \g__tblr_defined_hdash_styles_prop
  { solid = \hrule height \lTblrDefaultHruleWidthDim }
\prop_gset_from_keyval:Nn \g__tblr_defined_vdash_styles_prop
  { solid = \vrule width \lTblrDefaultVruleWidthDim }

\NewDocumentCommand \NewTblrDashStyle { m m }
  {
    \seq_set_split:Nnn \l_tmpa_seq { ~ } {#2}
    \tl_set:Ne \l__tblr_a_tl { \seq_item:Nn \l_tmpa_seq {1} }
    \tl_set:Ne \l__tblr_b_tl { \seq_item:Nn \l_tmpa_seq {2} }
    \tl_set:Ne \l__tblr_c_tl { \seq_item:Nn \l_tmpa_seq {3} }
    \tl_set:Ne \l__tblr_d_tl { \seq_item:Nn \l_tmpa_seq {4} }
    \tl_if_eq:NnT \l__tblr_a_tl { on }
      {
        \tl_if_eq:NnT \l__tblr_c_tl { off }
          {
            \__tblr_dash_style_make_boxes:nee {#1}
              { \dim_eval:n {\l__tblr_b_tl} } { \dim_eval:n {\l__tblr_d_tl} }
          }
      }
  }
\cs_set_eq:NN \NewDashStyle \NewTblrDashStyle

\cs_new_protected:Npn \__tblr_dash_style_make_boxes:nnn #1 #2 #3
  {
    \dim_set:Nn \l_tmpa_dim { #2 + #3 }
    \tl_set:Nn \l__tblr_h_tl { \hbox_to_wd:nn }
    \tl_put_right:Ne \l__tblr_h_tl { { \dim_use:N \l_tmpa_dim } }
    \tl_put_right:Nn \l__tblr_h_tl
      {
        {
          \hss
```

```
          \vbox:n
            { \hbox_to_wd:nn {#2} {} \hrule height \lTblrDefaultHruleWidthDim }
          \hss
        }
      }
    \prop_gput:NnV \g__tblr_defined_hdash_styles_prop {#1} \l__tblr_h_tl
    %\prop_log:N \g__tblr_defined_hdash_styles_prop
    \tl_set:Nn \l__tblr_v_tl { \vbox_to_ht:nn }
    \tl_put_right:Ne \l__tblr_v_tl { { \dim_use:N \l_tmpa_dim } }
    \tl_put_right:Nn \l__tblr_v_tl
      {
        {
          \vss
          \hbox:n
            { \vbox_to_ht:nn {#2} {} \vrule width \lTblrDefaultVruleWidthDim }
          \vss
        }
      }
    \prop_gput:NnV \g__tblr_defined_vdash_styles_prop {#1} \l__tblr_v_tl
    %\prop_log:N \g__tblr_defined_vdash_styles_prop
  }
\cs_generate_variant:Nn \__tblr_dash_style_make_boxes:nnn { nee }

\cs_new_protected:Npn \__tblr_get_hline_dash_style:N #1
  {
    \tl_set:Ne \l_tmpa_tl
      { \prop_item:NV \g__tblr_defined_hdash_styles_prop #1 }
    \tl_if_empty:NF \l_tmpa_tl { \tl_set_eq:NN #1 \l_tmpa_tl }
  }

\cs_new_protected:Npn \__tblr_get_vline_dash_style:N #1
  {
    \tl_set:Ne \l_tmpa_tl
      { \prop_item:NV \g__tblr_defined_vdash_styles_prop #1 }
    \tl_if_empty:NF \l_tmpa_tl { \tl_set_eq:NN #1 \l_tmpa_tl }
  }

\NewTblrDashStyle {dashed} {on ~ 2pt ~ off ~ 2pt}
\NewTblrDashStyle {dotted} {on ~ 0.4pt ~ off ~ 1pt}
```

9.14 Set hlines and vlines

```
\quark_new:N \q__tblr_dash
\quark_new:N \q__tblr_text

%% \SetHlines command for setting every hline in the table
\NewTblrTableCommand \SetHlines [3] [+]
  {
    \tblr_set_every_hline:nnn {#1} {#2} {#3}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_hline:nnn #1 #2 #3
  {
    \group_begin:
```

```
    \int_step_inline:nn { \int_eval:n { \c@rowcount + 1 } }
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_hline:nnn {#1} {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_hline in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_hline_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      {
        \int_compare:nNnTF { \tl_count:n {#1} } = {3}
          { \tblr_set_every_hline:nnn #1 }
          { \tblr_set_every_hline:nnn {1} #1 }
      }
      { \tblr_set_every_hline:nnn {1} {-} {#1} }
  }

%% Add \SetHline, \hline and \cline commands

\tl_new:N \l__tblr_hline_count_tl % the count of all hlines
\tl_new:N \l__tblr_hline_num_tl   % the index of the hline
\tl_new:N \l__tblr_hline_cols_tl  % the columns of the hline
\tl_new:N \l__tblr_hline_dash_tl  % dash style
\tl_new:N \l__tblr_hline_fg_tl    % dash foreground
\tl_new:N \l__tblr_hline_wd_tl    % dash width
\tl_new:N \l__tblr_hline_leftpos_tl  % left position
\tl_new:N \l__tblr_hline_rightpos_tl % right position
\bool_new:N \l__tblr_hline_endpos_bool % whether set positions only for both ends

\NewTblrTableCommand \cline [2] [] { \SetHline [=] {#2} {#1} }

\NewTblrTableCommand \hline [1] [] { \SetHline [+] {-} {#1} }

%% #1: the index of the hline (may be + or =)
%% #2: which columns of the hline, separate by commas
%% #3: key=value pairs
\NewTblrTableCommand \SetHline [3] [+]
  {
    \tblr_set_hline:nnn {#1} {#2} {#3}
  }

%% We need to check "text" key first
%% If it does exist and has empty value, then do nothing
\cs_new_protected:Npn \tblr_set_hline:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_keys_set_groups:nnn { hline/inner } { text } {#3}
    % true if "text=" is set
    \tl_if_eq:NNF \l__tblr_hline_dash_tl \q__tblr_text
      {
        \__tblr_set_hline_num:n {#1}
        \tl_clear:N \l__tblr_hline_dash_tl
```

```
        \__tblr_keys_set:nn { hline/inner } { dash = solid, #3 }
        \__tblr_set_hline_cmd:n {#2}
      }
    \group_end:
  }

\cs_new_protected:Npn \tblr_set_hline:nnnn #1 #2 #3 #4
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_eval:n { \c@rowcount + 1 } }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_hline:nnn {#2} {#3} {#4}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_hline in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_hline_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      {
        \int_compare:nNnTF { \tl_count:n {#2} } = {3}
          { \tblr_set_hline:nnnn #1 #2 }
          { \tblr_set_hline:nnnn #1 {1} #2 }
      }
      { \tblr_set_hline:nnnn #1 {1} {-} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_hline_aux:nn { Vn }

%% #1: the index of hline to set (may be + or =)
\cs_new_protected:Npn \__tblr_set_hline_num:n #1
  {
    \tl_clear:N \l__tblr_hline_num_tl
    \tl_set:Ne \l__tblr_hline_count_tl
      { \__tblr_spec_item:ne { hline } { [\int_use:N \c@rownum] / @hline-count } }
    %% \l__tblr_hline_count_tl may be empty when rowspec has extra |'s
    \int_compare:nNnTF { \l__tblr_hline_count_tl + 0 } = {0}
      {
        \tl_set:Ne \l__tblr_hline_num_tl { 1 }
        \__tblr_spec_gput:nen { hline }
          { [\int_use:N \c@rownum] / @hline-count } { 1 }
      }
      {
        \tl_if_eq:nnTF {#1} {+}
          { \__tblr_set_hline_num_incr: }
          {
            \tl_if_eq:nnTF {#1} {=}
              { \tl_set_eq:NN \l__tblr_hline_num_tl \l__tblr_hline_count_tl }
              {
                \int_compare:nNnTF {#1} > { \l__tblr_hline_count_tl }
                  { \__tblr_set_hline_num_incr: }
                  { \tl_set:Nn \l__tblr_hline_num_tl {#1} }
              }
```

```
          }
      }
  }

\cs_new_protected:Npn \__tblr_set_hline_num_incr:
  {
    \tl_set:Ne \l__tblr_hline_count_tl
      { \int_eval:n { \l__tblr_hline_count_tl + 1 } }
    \__tblr_spec_gput:nee { hline }
      { [\int_use:N \c@rownum] / @hline-count } { \l__tblr_hline_count_tl }
    \tl_set_eq:NN \l__tblr_hline_num_tl \l__tblr_hline_count_tl
  }

\__tblr_keys_define:nn { hline/inner }
  {
    dash .code:n = \tl_set:Nn \l__tblr_hline_dash_tl { \q__tblr_dash #1 },
    text .code:n = \tl_set:Nn \l__tblr_hline_dash_tl { \q__tblr_text #1 },
    text .groups:n = { text },
    wd .code:n = \tl_set:Nn \l__tblr_hline_wd_tl { \dim_eval:n {#1} },
    fg .code:n = \tl_set:Nn \l__tblr_hline_fg_tl {#1},
    leftpos  .code:n = \tl_set:Ne \l__tblr_hline_leftpos_tl {#1},
    rightpos .code:n = \tl_set:Ne \l__tblr_hline_rightpos_tl {#1},
    l        .meta:n = { leftpos = #1 },
    l     .default:n = { -0.8 },
    r        .meta:n = { rightpos = #1 },
    r     .default:n = { -0.8 },
    lr       .meta:n = { leftpos = #1, rightpos = #1 },
    lr    .default:n = { -0.8 },
    endpos .bool_set:N = \l__tblr_hline_endpos_bool,
    unknown .code:n = \__tblr_hline_unknown_key:V \l_keys_key_str,
  }

\cs_new_protected:Npn \__tblr_hline_unknown_key:n #1
  {
    \prop_if_in:NnTF \g__tblr_defined_hdash_styles_prop {#1}
      { \tl_set:Nn \l__tblr_hline_dash_tl { \q__tblr_dash #1 } }
      {
        \__tblr_if_color_value:nTF {#1}
          { \tl_set:Nn \l__tblr_hline_fg_tl {#1} }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \tl_set:Nn \l__tblr_hline_wd_tl { \dim_eval:n {\l__tblr_v_tl} }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_hline_unknown_key:n { V }

\cs_new_protected_nopar:Npn \__tblr_set_hline_cmd:n #1
  {
    \__tblr_child_parse:ne {#1} { \int_use:N \c@colcount }
    \__tblr_child_map_inline:n
      {
        % prevent expansion of vline text (see issue #303)
        \__tblr_set_hline_option:nnn { ##1 } { @dash }
          { \exp_not:V \l__tblr_hline_dash_tl }
        \tl_if_empty:NF \l__tblr_hline_wd_tl
          {
```

```
            \__tblr_set_hline_option:nnn { ##1 } { wd } { \l__tblr_hline_wd_tl }
          }
        \tl_if_empty:NF \l__tblr_hline_fg_tl
          {
            \__tblr_set_hline_option:nnn { ##1 } { fg } { \l__tblr_hline_fg_tl }
          }
      }
    \tl_if_empty:NF \l__tblr_hline_leftpos_tl
      {
        \bool_if:NTF \l__tblr_hline_endpos_bool
          {
            \__tblr_child_get_first:N \l_tmpa_tl
            \__tblr_set_hline_option:nnn
              { \l_tmpa_tl } { leftpos } { \l__tblr_hline_leftpos_tl }
          }
          {
            \__tblr_child_map_inline:n
              {
                \__tblr_set_hline_option:nnn
                  { ##1 } { leftpos } { \l__tblr_hline_leftpos_tl }
              }
          }
      }
    \tl_if_empty:NF \l__tblr_hline_rightpos_tl
      {
        \bool_if:NTF \l__tblr_hline_endpos_bool
          {
            \__tblr_child_get_last:N \l_tmpb_tl
            \__tblr_set_hline_option:nnn
              { \l_tmpb_tl } { rightpos } { \l__tblr_hline_rightpos_tl }
          }
          {
            \__tblr_child_map_inline:n
              {
                \__tblr_set_hline_option:nnn
                  { ##1 } { rightpos } { \l__tblr_hline_rightpos_tl }
              }
          }
      }
  }

%% #1: column; #2: key; #3: value
\cs_new_protected_nopar:Npn \__tblr_set_hline_option:nnn #1 #2 #3
  {
    \__tblr_spec_gput:nee { hline }
      { [\int_use:N \c@rownum][#1](\l__tblr_hline_num_tl) / #2 } { #3 }
  }


\msg_new:nnn { tabularray } { obsolete-firsthline }
  { \firsthline ~ is ~ obsolete; ~ use ~ 'baseline=T' ~ instead. }

\msg_new:nnn { tabularray } { obsolete-lasthline }
  { \lasthline ~ is ~ obsolete; ~ use ~ 'baseline=B' ~ instead. }

\NewTblrTableCommand \firsthline [1] []
  {
    \msg_error:nn { tabularray } { obsolete-firsthline }
```

```
  }

\NewTblrTableCommand \lasthline [1] []
  {
    \msg_error:nn { tabularray } { obsolete-lasthline }
  }

%% \SetVlines command for setting every vline in the table
\NewTblrTableCommand \SetVlines [3] [+]
  {
    \tblr_set_every_vline:nnn {#1} {#2} {#3}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_vline:nnn #1 #2 #3
  {
    \group_begin:
    \int_step_inline:nn { \c@colcount + 1 }
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_vline:nnn {#1} {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_vline in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_vline_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      {
        \int_compare:nNnTF { \tl_count:n {#1} } = {3}
          { \tblr_set_every_vline:nnn #1 }
          { \tblr_set_every_vline:nnn {1} #1 }
      }
      { \tblr_set_every_vline:nnn {1} {-} {#1} }
  }

%% Add \SetVline, \vline and \rline commands

\tl_new:N \l__tblr_vline_count_tl % the count of all vlines
\tl_new:N \l__tblr_vline_num_tl   % the index of the vline
\tl_new:N \l__tblr_vline_rows_tl  % the rows of the vline
\tl_new:N \l__tblr_vline_dash_tl  % dash style
\tl_new:N \l__tblr_vline_fg_tl    % dash foreground
\tl_new:N \l__tblr_vline_wd_tl    % dash width
\tl_new:N \l__tblr_vline_abovepos_tl % above position
\tl_new:N \l__tblr_vline_belowpos_tl % below position

\NewTblrTableCommand \rline [2] [] { \SetVline [=] {#2} {#1} }

\NewTblrTableCommand \vline [1] [] { \SetVline [+] {-} {#1} }

%% #1: the index of the vline (may be + or =)
%% #2: which rows of the vline, separate by commas
```

```
%% #3: key=value pairs
\NewTblrTableCommand \SetVline [3] [+]
  {
    \tblr_set_vline:nnn {#1} {#2} {#3}
  }

%% We need to check "text" key first
%% If it does exist and has empty value, then do nothing
\cs_new_protected:Npn \tblr_set_vline:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_keys_set_groups:nnn { vline/inner } { text } {#3}
    % true if "text=" is set
    \tl_if_eq:NNF \l__tblr_vline_dash_tl \q__tblr_text
      {
        \__tblr_set_vline_num:n {#1}
        \tl_clear:N \l__tblr_vline_dash_tl
        \__tblr_keys_set:nn { vline/inner } { dash = solid, #3 }
        \__tblr_set_vline_cmd:n {#2}
      }
    \group_end:
  }

\cs_new_protected:Npn \tblr_set_vline:nnnn #1 #2 #3 #4
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_eval:n { \c@colcount + 1} }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_vline:nnn {#2} {#3} {#4}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_vline in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_vline_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      {
        \int_compare:nNnTF { \tl_count:n {#2} } = {3}
          { \tblr_set_vline:nnnn #1 #2 }
          { \tblr_set_vline:nnnn #1 {1} #2 }
      }
      { \tblr_set_vline:nnnn #1 {1} {-} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_vline_aux:nn { Vn }

%% #1: the index of vline to set (may be + or =)
\cs_new_protected:Npn \__tblr_set_vline_num:n #1
  {
    \tl_clear:N \l__tblr_vline_num_tl
    \tl_set:Ne \l__tblr_vline_count_tl
      { \__tblr_spec_item:ne { vline } { [\int_use:N \c@colnum] / @vline-count } }
    %% \l__tblr_vline_count_tl may be empty when colspec has extra |'s
```

```
    \int_compare:nNnTF { \l__tblr_vline_count_tl + 0 } = {0}
      {
        \tl_set:Ne \l__tblr_vline_num_tl { 1 }
        \__tblr_spec_gput:nen { vline }
          { [\int_use:N \c@colnum] / @vline-count } { 1 }
      }
      {
        \tl_if_eq:nnTF {#1} {+}
          { \__tblr_set_vline_num_incr: }
          {
            \tl_if_eq:nnTF {#1} {=}
              { \tl_set_eq:NN \l__tblr_vline_num_tl \l__tblr_vline_count_tl }
              {
                \int_compare:nNnTF {#1} > { \l__tblr_vline_count_tl }
                  { \__tblr_set_vline_num_incr: }
                  { \tl_set:Nn \l__tblr_vline_num_tl {#1} }
              }
          }
      }
  }

\cs_new_protected:Npn \__tblr_set_vline_num_incr:
  {
    \tl_set:Ne \l__tblr_vline_count_tl
      { \int_eval:n { \l__tblr_vline_count_tl + 1 } }
    \__tblr_spec_gput:nee { vline }
      { [\int_use:N \c@colnum] / @vline-count } { \l__tblr_vline_count_tl }
    \tl_set_eq:NN \l__tblr_vline_num_tl \l__tblr_vline_count_tl
  }

\__tblr_keys_define:nn { vline/inner }
  {
    dash .code:n = \tl_set:Nn \l__tblr_vline_dash_tl { \q__tblr_dash #1 },
    text .code:n = \tl_set:Nn \l__tblr_vline_dash_tl { \q__tblr_text #1 },
    text .groups:n = { text },
    wd .code:n = \tl_set:Nn \l__tblr_vline_wd_tl { \dim_eval:n {#1} },
    fg .code:n = \tl_set:Nn \l__tblr_vline_fg_tl {#1},
    abovepos .code:n = \tl_set:Ne \l__tblr_vline_abovepos_tl {#1},
    belowpos .code:n = \tl_set:Ne \l__tblr_vline_belowpos_tl {#1},
    unknown .code:n = \__tblr_vline_unknown_key:V \l_keys_key_str,
  }

\cs_new_protected:Npn \__tblr_vline_unknown_key:n #1
  {
    \prop_if_in:NnTF \g__tblr_defined_vdash_styles_prop {#1}
      { \tl_set:Nn \l__tblr_vline_dash_tl { \q__tblr_dash #1 } }
      {
        \__tblr_if_color_value:nTF {#1}
          { \tl_set:Nn \l__tblr_vline_fg_tl {#1} }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \tl_set:Nn \l__tblr_vline_wd_tl { \dim_eval:n {\l__tblr_v_tl} }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_vline_unknown_key:n { V }
```

```
\cs_new_protected_nopar:Npn \__tblr_set_vline_cmd:n #1
  {
    \__tblr_child_parse:ne {#1} { \int_use:N \c@rowcount }
    \__tblr_child_map_inline:n
      {
        \__tblr_spec_gput:nee { vline }
          { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / @dash }
          % prevent expansion of vline text (see issue #303)
          { \exp_not:V \l__tblr_vline_dash_tl }
        \tl_if_empty:NF \l__tblr_vline_wd_tl
          {
            \__tblr_spec_gput:nee { vline }
              { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / wd }
              { \l__tblr_vline_wd_tl }
          }
        \tl_if_empty:NF \l__tblr_vline_fg_tl
          {
            \__tblr_spec_gput:nee { vline }
              { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / fg }
              { \l__tblr_vline_fg_tl }
          }
        \tl_if_empty:NF \l__tblr_vline_abovepos_tl
          {
            \__tblr_spec_gput:nee { vline }
              { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / abovepos }
              { \l__tblr_vline_abovepos_tl }
          }
        \tl_if_empty:NF \l__tblr_vline_belowpos_tl
          {
            \__tblr_spec_gput:nee { vline }
              { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / belowpos }
              { \l__tblr_vline_belowpos_tl }
          }
      }
  }
```

9.15 Set hborders and vborders

```
%% Hborder holds keys not related to a specified hline
\NewTblrTableCommand \hborder [1] { \tblr_set_hborder:n {#1} }

\cs_new_protected:Npn \tblr_set_hborder:n #1
  {
    \__tblr_keys_set:nn { hborder/inner } {#1}
  }

\cs_new_protected:Npn \tblr_set_hborder:nn #1 #2
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_eval:n { \c@rowcount + 1 } }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_hborder:n {#2}
      }
    \group_end:
```

```
  }

%% This function is called when parsing table specifications
%% Note that #1 always includes an outer pair of braces
\cs_new_protected:Npn \__tblr_set_hborder_aux:nn #1 #2
  {
    \tblr_set_hborder:nn #1 {#2}
  }
\cs_generate_variant:Nn \__tblr_set_hborder_aux:nn { Vn }

\__tblr_keys_define:nn { hborder/inner }
  {
    abovespace .code:n = \__tblr_row_gput_above:ne
                         { belowsep } { \dim_eval:n {#1} },
    belowspace .code:n = \__tblr_row_gput:ne { abovesep } { \dim_eval:n {#1} },
    abovespace+ .code:n = \__tblr_row_gadd_dimen_above:ne
                         { belowsep } { \dim_eval:n {#1} },
    belowspace+ .code:n = \__tblr_row_gadd_dimen:ne
                         { abovesep } { \dim_eval:n {#1} },
    pagebreak   .code:n = \__tblr_hborder_gput_pagebreak:n {#1},
    pagebreak   .default:n = yes,
    baseline    .code:n = \__tblr_outer_gput_spec:ne
                         { baseline } { - \int_use:N \c@rownum },
  }

\tl_const:Nn \c__tblr_pagebreak_yes_tl  {  1 }
\tl_const:Nn \c__tblr_pagebreak_auto_tl {  0 }
\tl_const:Nn \c__tblr_pagebreak_no_tl   { -1 }

\cs_new_protected:Npn \__tblr_hborder_gput_pagebreak:n #1
  {
    \tl_if_exist:cT { c__tblr_pagebreak_ #1 _tl }
      {
        \__tblr_spec_gput:nee { hline }
          { [\int_use:N \c@rownum] / @pagebreak }
          { \tl_use:c { c__tblr_pagebreak_ #1 _tl } }
      }
  }

%% Vborder holds keys not related to a specified vline
\NewTblrTableCommand \vborder [1] { \tblr_set_vborder:n {#1} }

\cs_new_protected:Npn \tblr_set_vborder:n #1
  {
    \__tblr_keys_set:nn { vborder/inner } {#1}
  }

\cs_new_protected:Npn \tblr_set_vborder:nn #1 #2
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_eval:n { \c@colcount + 1 } }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_vborder:n {#2}
      }
```

```
    \group_end:
  }

%% This function is called when parsing table specifications
%% Note that #1 always includes an outer pair of braces
\cs_new_protected:Npn \__tblr_set_vborder_aux:nn #1 #2
  {
    \tblr_set_vborder:nn #1 {#2}
  }
\cs_generate_variant:Nn \__tblr_set_vborder_aux:nn { Vn }

\__tblr_keys_define:nn { vborder/inner }
  {
    leftspace .code:n = \__tblr_column_gput_left:ne
                        { rightsep } { \dim_eval:n {#1} },
    rightspace .code:n = \__tblr_column_gput:ne { leftsep } { \dim_eval:n {#1} },
    leftspace+ .code:n = \__tblr_column_gadd_dimen_left:ne
                        { rightsep } { \dim_eval:n {#1} },
    rightspace+ .code:n = \__tblr_column_gadd_dimen:ne
                        { leftsep } { \dim_eval:n {#1} },
  }
```

9.16 Set cells

```
%% \SetCells command for setting every cell in the table
\NewTblrTableCommand \SetCells [2] []
  {
    \tblr_set_every_cell:nn {#1} {#2}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_cell:nn #1 #2
  {
    \group_begin:
    \int_step_inline:nn { \c@rowcount }
      {
        \int_set:Nn \c@rownum {##1}
        \int_step_inline:nn { \c@colcount }
          {
            \int_set:Nn \c@colnum {####1}
            \tblr_set_cell:nn {#1} {#2}
          }
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_cell in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_cell_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \tblr_set_every_cell:nn #1 }
      { \tblr_set_every_cell:nn {} {#1} }
  }
```

```
%% \SetCell command for multirow and/or multicolumn cells

\NewTblrTableCommand \SetCell [2] []
  {
    \tblr_set_cell:nn { #1 } { #2 }
  }

\tl_new:N \l__tblr_row_span_num_tl
\tl_new:N \l__tblr_col_span_num_tl

\cs_new_protected:Npn \tblr_set_cell:nn #1 #2
  {
    \tl_set:Nn \l__tblr_row_span_num_tl { 1 }
    \tl_set:Nn \l__tblr_col_span_num_tl { 1 }
    \__tblr_keys_set:nn { cell/outer } { #1 }
    \__tblr_keys_set:nn { cell/inner } { #2 }
    \__tblr_set_span_spec:VV \l__tblr_row_span_num_tl \l__tblr_col_span_num_tl
  }
\cs_generate_variant:Nn \tblr_set_cell:nn { nV }

\cs_new_protected:Npn \tblr_set_cell:nnnn #1 #2 #3 #4
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_use:N \c@rowcount }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@rownum {##1}
        \__tblr_child_parse:ne {#2} { \int_use:N \c@colcount }
        \__tblr_child_submap_inline:n
          {
            \int_set:Nn \c@colnum {####1}
            \tblr_set_cell:nn {#3} {#4}
          }
      }
    \group_end:
  }

\cs_new_protected:Npn \__tblr_set_cell_tuple:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_child_parse_tuple:nee {#1}
      { \int_use:N \c@rowcount } { \int_use:N \c@colcount }
    \clist_map_inline:Nn \l__tblr_child_tuple_whole_clist
      {
        \int_set:Nn \c@rownum { \use_i:nn ##1 }
        \int_set:Nn \c@colnum { \use_ii:nn ##1 }
        \tblr_set_cell:nn {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_cell in different ways.
%% #1 consists of either two tl items {h_1,h_2,...,h_m}{v_1,v_2,...,v_n},
%% or one tl item {{h_1}{v_1},{h_2}{v_2},...,{h_k}{v_k}}.
%% This function is called when parsing table specifications.
\cs_new_protected:Npn \__tblr_set_cell_aux:nn #1 #2
```

```
  {
    \tl_if_single:nTF {#1}
      {
        \tl_if_head_is_group:nTF {#2}
          { \__tblr_set_cell_tuple:nnn #1 #2 }
          { \__tblr_set_cell_tuple:nnn #1 {} {#2} }
      }
      {
        \tl_if_head_is_group:nTF {#2}
          { \tblr_set_cell:nnnn #1 #2 }
          { \tblr_set_cell:nnnn #1 {} {#2} }
      }
  }
\cs_generate_variant:Nn \__tblr_set_cell_aux:nn { Vn }

\__tblr_keys_define:nn { cell/outer }
  {
    r .tl_set:N = \l__tblr_row_span_num_tl,
    c .tl_set:N = \l__tblr_col_span_num_tl,
  }

\__tblr_keys_define:nn { cell/inner }
  {
    halign  .code:n = \__tblr_cell_gput:nn { halign } {#1},
    valign  .code:n = \__tblr_cell_gput:nn { valign } {#1},
    j       .meta:n = { halign = j },
    l       .meta:n = { halign = l },
    c       .meta:n = { halign = c },
    r       .meta:n = { halign = r },
    t       .meta:n = { valign = t },
    p       .meta:n = { valign = t },
    m       .meta:n = { valign = m },
    b       .meta:n = { valign = b },
    h       .meta:n = { valign = h },
    f       .meta:n = { valign = f },
    wd      .code:n = \__tblr_cell_gput:ne { width } {#1},
    bg      .code:n = \__tblr_cell_gput:ne { background } {#1},
    fg      .code:n = \__tblr_cell_gput:ne { foreground } {#1},
    font    .code:n = \__tblr_cell_gput:nn { font } { #1 \selectfont },
    mode    .code:n = \__tblr_cell_gput:nn { mode } {#1},
    $       .meta:n = { mode = math },
    $$      .meta:n = { mode = dmath },
    cmd     .code:n = \__tblr_cell_gput:nn { cmd } {#1},
    preto   .code:n = \__tblr_cell_preto_text:n {#1},
    appto   .code:n = \__tblr_cell_appto_text:n {#1},
    unknown .code:n = \__tblr_cell_unknown_key:V \l_keys_key_str,
  }

\cs_new_protected:Npn \__tblr_cell_gput:nn #1 #2
  {
    \__tblr_data_gput:neenn { cell }
      { \int_use:N \c@rownum } { \int_use:N \c@colnum } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_cell_gput:nn { ne }

\cs_new_protected:Npn \__tblr_cell_gput:nnnn #1 #2 #3 #4
  {
```

```
    \__tblr_data_gput:nnnnn { cell } {#1} {#2} {#3} {#4}
  }
\cs_generate_variant:Nn \__tblr_cell_gput:nnnn
  { nenn, ennn, eenn, nene, enne, eene }

\tl_new:N \l__tblr_cell_text_tl

\cs_new_protected:Npn \__tblr_cell_preto_text:n #1
  {
    \__tblr_cell_preto_text:een
      { \int_use:N \c@rownum } { \int_use:N \c@colnum } {#1}
  }

\cs_new_protected:Npn \__tblr_cell_preto_text:nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_cell_text_tl { \__tblr_spec_item:nn { text } { [#1][#2] } }
    \tl_put_left:Nn \l__tblr_cell_text_tl {#3}
    \__tblr_spec_gput:nnV { text } { [#1][#2] } \l__tblr_cell_text_tl
  }
\cs_generate_variant:Nn \__tblr_cell_preto_text:nnn { nen, enn, een }

\cs_new_protected:Npn \__tblr_cell_appto_text:n #1
  {
    \__tblr_cell_appto_text:een
      { \int_use:N \c@rownum } { \int_use:N \c@colnum } {#1}
  }

\cs_new_protected:Npn \__tblr_cell_appto_text:nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_cell_text_tl { \__tblr_spec_item:ne { text } { [#1][#2] } }
    \tl_put_right:Nn \l__tblr_cell_text_tl {#3}
    \__tblr_spec_gput:neV { text } { [#1][#2] } \l__tblr_cell_text_tl
  }
\cs_generate_variant:Nn \__tblr_cell_appto_text:nnn { nen, enn, een }

\cs_new_protected:Npn \__tblr_cell_unknown_key:n #1
  {
    \__tblr_if_color_value:nTF {#1}
      {
        \__tblr_data_gput:neene { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { background } {#1}
      }
      {
        \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
        \__tblr_data_gput:neene { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { width }
          { \dim_eval:n { \l__tblr_v_tl } }
      }
  }
\cs_generate_variant:Nn \__tblr_cell_unknown_key:n { V }

%% Whether to allow page breaks in the middle of multirow cells
\bool_new:N \lTblrCellBreakBool

\cs_new_protected:Npn \__tblr_set_span_spec:nn #1 #2
  {
```

```
    \int_compare:nNnT { #1 } > { 1 }
      {
        \__tblr_prop_gput:nnn { inner } { rowspan } { true }
        \__tblr_data_gput:neenn { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { rowspan } {#1}
      }
    \int_compare:nNnT { #2 } > { 1 }
      {
        \__tblr_prop_gput:nnn { inner } { colspan } { true }
        \__tblr_data_gput:neenn { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { colspan } {#2}
      }
    \int_step_variable:nnNn
      { \int_use:N \c@rownum } { \int_eval:n { \c@rownum + #1 - 1 } } \l__tblr_i_tl
      {
        \bool_lazy_and:nnT
          { ! \lTblrCellBreakBool }
          { \int_compare_p:nNn {\l__tblr_i_tl} > {\c@rownum} }
          {
            \__tblr_spec_gput:nen {hline} { [\l__tblr_i_tl] / @pagebreak } {-1}
          }
        \int_step_variable:nnNn
          { \int_use:N \c@colnum } { \int_eval:n { \c@colnum + #2 - 1 } }
          \l__tblr_j_tl
          {
            \bool_lazy_and:nnF
              { \int_compare_p:nNn { \l__tblr_i_tl } = { \c@rownum } }
              { \int_compare_p:nNn { \l__tblr_j_tl } = { \c@colnum } }
              {
                \__tblr_data_gput:neenn { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { omit } {1}
              }
            \int_compare:nNnF { \l__tblr_i_tl } = { \c@rownum }
              {
                \__tblr_spec_gput:nen { hline }
                  { [\l__tblr_i_tl][\l__tblr_j_tl] / omit } {true}
              }
            \int_compare:nNnF { \l__tblr_j_tl } = { \c@colnum }
              {
                \__tblr_spec_gput:nee { vline }
                  { [\l__tblr_i_tl][\l__tblr_j_tl] / omit } {true}
              }
          }
      }
    %% Make continuous borders for multirow cells
    \tl_set:Ne \l__tblr_n_tl
      {
        \int_max:nn
          {
            \__tblr_spec_item:ne { vline } { [\int_use:N \c@colnum] / @vline-count }
          }
          { 1 }
      }
    \int_step_variable:nnNn
      { \c@rownum } { \int_eval:n { \c@rownum + #1 - 2 } } \l__tblr_i_tl
      {
        \__tblr_spec_gput:nee { vline }
          { [\l__tblr_i_tl][\int_use:N \c@colnum](\l__tblr_n_tl) / belowpos } {1}
```

```
        \__tblr_spec_gput:nee { vline }
          { [\l__tblr_i_tl][\int_eval:n {\c@colnum + #2}](1) / belowpos } {1}
      }
  }
\cs_generate_variant:Nn \__tblr_set_span_spec:nn { VV }

%% Obsolete \multicolumn and \multirow commands

\msg_new:nnn { tabularray } { obsolete-multicolumn }
  { \multicolumn ~ is ~ obsolete; ~ use ~ \SetCell ~ instead. }

\msg_new:nnn { tabularray } { obsolete-multirow }
  { \multirow ~ is ~ obsolete; ~ use ~ \SetCell ~ instead. }

\NewTblrTableCommand \multicolumn [2]
  {
    \msg_error:nn { tabularray } { obsolete-multicolumn }
  }

\NewTblrTableCommand \multirow [3] [m]
  {
    \msg_error:nn { tabularray } { obsolete-multirow }
  }
```

9.17 Set columns and rows

```
%% \SetColumns command for setting every column in the table
\NewTblrTableCommand \SetColumns [2] []
  {
    \tblr_set_every_column:nn {#1} {#2}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_column:nn #1 #2
  {
    \group_begin:
    \int_step_inline:nn { \c@colcount }
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_column:nn {#1} {#2}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_column in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_column_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \tblr_set_every_column:nn #1 }
      { \tblr_set_every_column:nn {} {#1} }
  }

%% \SetColumn command for current column or each cells in the column
```

```
\NewTblrTableCommand \SetColumn [2] []
  {
    \tblr_set_column:nn {#1} {#2}
  }

\cs_new_protected:Npn \tblr_set_column:nn #1 #2
  {
    \__tblr_keys_set:nn { column/inner } {#2}
  }

\cs_new_protected:Npn \tblr_set_column:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_use:N \c@colcount }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_column:nn {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_column in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_column_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      { \tblr_set_column:nnn #1 #2 }
      { \tblr_set_column:nnn #1 {} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_column_aux:nn { Vn }

\__tblr_keys_define:nn { column/inner }
  {
    halign     .code:n = \__tblr_column_gput_cell:nn { halign } {#1},
    valign     .code:n = \__tblr_column_gput_cell:nn { valign } {#1},
    j          .meta:n = { halign = j },
    l          .meta:n = { halign = l },
    c          .meta:n = { halign = c },
    r          .meta:n = { halign = r },
    t          .meta:n = { valign = t },
    p          .meta:n = { valign = t },
    m          .meta:n = { valign = m },
    b          .meta:n = { valign = b },
    h          .meta:n = { valign = h },
    f          .meta:n = { valign = f },
    bg         .code:n = \__tblr_column_gput_cell:nn { background } {#1},
    fg         .code:n = \__tblr_column_gput_cell:nn { foreground } {#1},
    font       .code:n = \__tblr_column_gput_cell:nn { font } { #1 \selectfont },
    mode       .code:n = \__tblr_column_gput_cell:nn { mode } {#1},
    $          .meta:n = { mode = math },
    $$         .meta:n = { mode = dmath },
    cmd        .code:n = \__tblr_column_gput_cell:nn { cmd } {#1},
    wd         .code:n = \__tblr_column_gput:ne { width } { \dim_eval:n {#1} },
    co         .code:n = \__tblr_column_gput:ne { coefficient } {#1},
    preto      .code:n = \__tblr_preto_text_for_every_column_cell:n {#1},
```

```
    appto      .code:n = \__tblr_appto_text_for_every_column_cell:n {#1},
    leftsep    .code:n = \__tblr_column_gput:ne { leftsep } { \dim_eval:n {#1} },
    rightsep   .code:n = \__tblr_column_gput:ne { rightsep } { \dim_eval:n {#1} },
    colsep     .meta:n = { leftsep = #1, rightsep = #1},
    leftsep+   .code:n = \__tblr_column_gadd_dimen:ne
                           { leftsep } { \dim_eval:n {#1} },
    rightsep+  .code:n = \__tblr_column_gadd_dimen:ne
                           { rightsep } { \dim_eval:n {#1} },
    colsep+    .meta:n = { leftsep+ = #1, rightsep+ = #1},
    unknown    .code:n = \__tblr_column_unknown_key:V \l_keys_key_str,
  }

%% #1: key; #2: value
\cs_new_protected:Npn \__tblr_column_gput:nn #1 #2
  {
    \__tblr_data_gput:nenn { column } { \int_use:N \c@colnum } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_column_gput:nn { ne }

\cs_new_protected:Npn \__tblr_column_gput_left:nn #1 #2
  {
    \__tblr_data_gput:nenn { column } { \int_eval:n { \c@colnum - 1 } } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_column_gput_left:nn { ne }

\cs_new_protected:Npn \__tblr_column_gadd_dimen:nn #1 #2
  {
    \__tblr_data_gadd_dimen_value:nenn { column }
      { \int_use:N \c@colnum } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_column_gadd_dimen:nn { ne }

\cs_new_protected:Npn \__tblr_column_gadd_dimen_left:nn #1 #2
  {
    \__tblr_data_gadd_dimen_value:nenn { column }
      { \int_eval:n { \c@colnum - 1 } } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_column_gadd_dimen_left:nn { ne }

%% #1: key; #2: value
\cs_new_protected:Npn \__tblr_column_gput_cell:nn #1 #2
  {
    \int_step_inline:nn { \c@rowcount }
      {
        \__tblr_cell_gput:nenn {##1} { \int_use:N \c@colnum } {#1} {#2}
      }
  }
\cs_generate_variant:Nn \__tblr_column_gput_cell:nn { ne }

\cs_new_protected:Npn \__tblr_preto_text_for_every_column_cell:n #1
  {
    \int_step_inline:nn { \c@rowcount }
      {
        \__tblr_cell_preto_text:nen {##1} { \int_use:N \c@colnum } {#1}
      }
  }
```

```
\cs_new_protected:Npn \__tblr_appto_text_for_every_column_cell:n #1
  {
    \int_step_inline:nn { \c@rowcount }
      {
        \__tblr_cell_appto_text:nen {##1} { \int_use:N \c@colnum } {#1}
      }
  }

\cs_new_protected:Npn \__tblr_column_unknown_key:n #1
  {
    \__tblr_if_number_value:nTF {#1}
      { \__tblr_column_gput:ne { coefficient } {#1} }
      {
        \__tblr_if_color_value:nTF {#1}
          { \__tblr_column_gput_cell:nn { background } {#1} }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \__tblr_column_gput:ne { width } { \dim_eval:n { \l__tblr_v_tl } }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_column_unknown_key:n { V }

%% \SetRows command for setting every row in the table
\NewTblrTableCommand \SetRows [2] []
  {
    \tblr_set_every_row:nn {#1} {#2}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_row:nn #1 #2
  {
    \group_begin:
    \int_step_inline:nn { \c@rowcount }
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_row:nn {#1} {#2}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_row in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_row_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \tblr_set_every_row:nn #1 }
      { \tblr_set_every_row:nn {} {#1} }
  }

%% \SetRow command for current row or each cells in the row

\NewTblrTableCommand \SetRow [2] []
  {
    \tblr_set_row:nn {#1} {#2}
  }
```

```
\cs_new_protected:Npn \tblr_set_row:nn #1 #2
  {
    \__tblr_keys_set:nn { row/inner } {#2}
  }

\cs_new_protected:Npn \tblr_set_row:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_child_parse:ne {#1} { \int_use:N \c@rowcount }
    \__tblr_child_map_inline:n
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_row:nn {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_row in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_row_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      { \tblr_set_row:nnn #1 #2 }
      { \tblr_set_row:nnn #1 {} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_row_aux:nn { Vn }

\__tblr_keys_define:nn { row/inner }
  {
    halign     .code:n = \__tblr_row_gput_cell:nn { halign } {#1},
    valign     .code:n = \__tblr_row_gput_cell:nn { valign } {#1},
    j          .meta:n = { halign = j },
    l          .meta:n = { halign = l },
    c          .meta:n = { halign = c },
    r          .meta:n = { halign = r },
    t          .meta:n = { valign = t },
    p          .meta:n = { valign = t },
    m          .meta:n = { valign = m },
    b          .meta:n = { valign = b },
    h          .meta:n = { valign = h },
    f          .meta:n = { valign = f },
    bg         .code:n = \__tblr_row_gput_cell:nn { background } {#1},
    fg         .code:n = \__tblr_row_gput_cell:nn { foreground } {#1},
    font       .code:n = \__tblr_row_gput_cell:nn { font } { #1 \selectfont },
    mode       .code:n = \__tblr_row_gput_cell:nn { mode } {#1},
    $          .meta:n = { mode = math },
    $$         .meta:n = { mode = dmath },
    cmd        .code:n = \__tblr_row_gput_cell:nn { cmd } {#1},
    ht         .code:n = \__tblr_row_gput:ne { height } { \dim_eval:n {#1} },
    co         .code:n = \__tblr_row_gput:ne { coefficient } {#1},
    preto      .code:n = \__tblr_preto_text_for_every_row_cell:n {#1},
    appto      .code:n = \__tblr_appto_text_for_every_row_cell:n {#1},
    abovesep   .code:n = \__tblr_row_gput:ne { abovesep } { \dim_eval:n {#1} },
    belowsep   .code:n = \__tblr_row_gput:ne { belowsep } { \dim_eval:n {#1} },
    rowsep     .meta:n = { abovesep = #1, belowsep = #1},
    abovesep+  .code:n = \__tblr_row_gadd_dimen:ne { abovesep } { \dim_eval:n {#1} },
```

```
    belowsep+ .code:n = \__tblr_row_gadd_dimen:ne { belowsep } { \dim_eval:n {#1} },
    rowsep+   .meta:n = { abovesep+ = #1, belowsep+ = #1},
    baseline  .code:n = \__tblr_outer_gput_spec:ne
                          { baseline } { \int_use:N \c@rownum },
    unknown   .code:n = \__tblr_row_unknown_key:V \l_keys_key_str,
  }

%% #1: key; #2: value
\cs_new_protected:Npn \__tblr_row_gput:nn #1 #2
  {
    \__tblr_data_gput:nenn { row } { \int_use:N \c@rownum } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_row_gput:nn { ne }

\cs_new_protected:Npn \__tblr_row_gput_above:nn #1 #2
  {
    \__tblr_data_gput:nenn { row } { \int_eval:n { \c@rownum - 1 } } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_row_gput_above:nn { ne }

\cs_new_protected:Npn \__tblr_row_gadd_dimen:nn #1 #2
  {
    \__tblr_data_gadd_dimen_value:nenn { row } { \int_use:N \c@rownum } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_row_gadd_dimen:nn { ne }

\cs_new_protected:Npn \__tblr_row_gadd_dimen_above:nn #1 #2
  {
    \__tblr_data_gadd_dimen_value:nenn { row }
      { \int_eval:n { \c@rownum - 1 } } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_row_gadd_dimen_above:nn { ne }

%% #1: key; #2: value
\cs_new_protected:Npn \__tblr_row_gput_cell:nn #1 #2
  {
    \int_step_inline:nn { \c@colcount }
      {
        \__tblr_cell_gput:ennn { \int_use:N \c@rownum } {##1} {#1} {#2}
      }
  }
\cs_generate_variant:Nn \__tblr_row_gput_cell:nn { ne }

\cs_new_protected:Npn \__tblr_preto_text_for_every_row_cell:n #1
  {
    \int_step_inline:nn { \c@colcount }
      {
        \__tblr_cell_preto_text:enn { \int_use:N \c@rownum } {##1} {#1}
      }
  }

\cs_new_protected:Npn \__tblr_appto_text_for_every_row_cell:n #1
  {
    \int_step_inline:nn { \c@colcount }
      {
        \__tblr_cell_appto_text:enn { \int_use:N \c@rownum } {##1} {#1}
```

```
      }
  }

\cs_new_protected:Npn \__tblr_row_unknown_key:n #1
  {
    \__tblr_if_number_value:nTF {#1}
      {
        \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
          { coefficient } {#1}
      }
      {
        \__tblr_if_color_value:nTF {#1}
          { \__tblr_row_gput_cell:nn { background } {#1} }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \__tblr_row_gput:ne { height } { \dim_eval:n { \l__tblr_v_tl } }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_row_unknown_key:n { V }

\NewTblrTableCommand \pagebreak [1] [4]
  {
    \hborder { pagebreak = yes }
  }

\NewTblrTableCommand \nopagebreak [1] [4]
  {
    \hborder { pagebreak = no }
  }
```

9.18 Column types and row types

```
%% Some primitive column/row types

\str_const:Nn \cTblrPrimitiveColrowTypeStr { Q | < > }
\tl_new:N \g__tblr_expanded_colrow_spec_tl

\exp_args:Nc \NewDocumentCommand { tblr_primitive_Column_type_ Q } { O{} }
  {
    \__tblr_keys_set:nn { column/inner } { #1 }
    \int_incr:N \c@colnum
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Column_type_ Q } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { Q[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_Row_type_ Q } { O{} }
  {
    \__tblr_keys_set:nn { row/inner } { #1 }
    \int_incr:N \c@rownum
    \__tblr_execute_colrow_spec_next:N
```

```
  }
\exp_args:Nc \NewDocumentCommand { tblr_Row_type_ Q } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { Q[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_Column_type_ | } { O{} }
  {
    \vline [#1]
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Column_type_ | } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { |[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_Row_type_ | } { O{} }
  {
    \hline [#1]
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Row_type_ | } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { |[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_Column_type_ > } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene
          { column }
          { \int_use:N \c@colnum } { leftsep }
          { \dim_eval:n {#1} }
      }
    \tl_if_blank:nF {#2}
      {
        \__tblr_preto_text_for_every_column_cell:n {#2}
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Column_type_ > } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { >[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_Row_type_ > } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
          { abovesep } { \dim_eval:n { #1 } }
      }
```

```
    \tl_if_blank:nF {#2}
      {
        \__tblr_preto_text_for_every_row_cell:n {#2}
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Row_type_ > } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { >[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }


\exp_args:Nc \NewDocumentCommand { tblr_primitive_Column_type_ < } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene { column }
          { \int_eval:n {\c@colnum - 1} } { rightsep } { \dim_eval:n {#1} }
      }
    \tl_if_blank:nF {#2}
      {
        \group_begin:
        \int_decr:N \c@colnum
        \__tblr_appto_text_for_every_column_cell:n {#2}
        \group_end:
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Column_type_ < } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { <[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }


\exp_args:Nc \NewDocumentCommand { tblr_primitive_Row_type_ < } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene { row } { \int_eval:n {\c@rownum - 1} }
          { belowsep } { \dim_eval:n {#1} }
      }
    \tl_if_blank:nF {#2}
      {
        \group_begin:
        \int_decr:N \c@rownum
        \__tblr_appto_text_for_every_row_cell:n {#2}
        \group_end:
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_Row_type_ < } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { <[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }

%% \NewColumnType/\NewRowType command and predefined column/row types
```

```
\str_new:N \gTblrUsedColumnTypeStr
\str_gset_eq:NN \gTblrUsedColumnTypeStr \cTblrPrimitiveColrowTypeStr

\str_new:N \gTblrUsedRowTypeStr
\str_gset_eq:NN \gTblrUsedRowTypeStr \cTblrPrimitiveColrowTypeStr

\bool_new:N \g__tblr_colrow_spec_expand_stop_bool
\tl_new:N \g__tblr_column_or_row_tl

\msg_new:nnn { tabularray } { used-colrow-type }
  { #1 ~ type ~ name ~ #2 ~ has ~ been ~ used! }

\NewDocumentCommand \NewTblrColumnType { m O{0} o m }
  {
    \tl_gset:Nn \g__tblr_column_or_row_tl { Column }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
  }
\cs_set_eq:NN \NewColumnType \NewTblrColumnType

\NewDocumentCommand \NewTblrRowType { m O{0} o m }
  {
    \tl_gset:Nn \g__tblr_column_or_row_tl { Row }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
  }
\cs_set_eq:NN \NewRowType \NewTblrRowType

\NewDocumentCommand \NewTblrColumnRowType { m O{0} o m }
  {
    \tl_gset:Nn \g__tblr_column_or_row_tl { Column }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
    \tl_gset:Nn \g__tblr_column_or_row_tl { Row }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
  }
\cs_set_eq:NN \NewColumnRowType \NewTblrColumnRowType

\cs_new_protected:Npn \__tblr_new_column_or_row_type:nnnn #1 #2 #3 #4
  {
    \str_if_in:cnTF { gTblrUsed \g__tblr_column_or_row_tl TypeStr } {#1}
      {
        \tl_if_eq:NnTF \g__tblr_column_or_row_tl { Row }
          { \msg_error:nnnn { tabularray } { used-colrow-type } { Row } {#1} }
          { \msg_error:nnnn { tabularray } { used-colrow-type } { Column } {#1} }
        \str_log:c { gTblrUsed \g__tblr_column_or_row_tl TypeStr }
      }
      {
        \__tblr_make_xparse_arg_spec:nnN {#2} {#3} \l__tblr_a_tl
        \exp_args:NcV \NewDocumentCommand
          { tblr_ \g__tblr_column_or_row_tl _type_ #1 } \l__tblr_a_tl
          {
            \bool_gset_false:N \g__tblr_colrow_spec_expand_stop_bool
            \tl_gput_right:Nf \g__tblr_expanded_colrow_spec_tl {#4}
            \__tblr_expand_colrow_spec_next:N
          }
        \str_gput_right:cn
          { gTblrUsed \g__tblr_column_or_row_tl TypeStr } {#1}
      }
```

```
  }

\NewTblrColumnRowType { l } { Q[l] }
\NewTblrColumnRowType { c } { Q[c] }
\NewTblrColumnRowType { r } { Q[r] }
\NewTblrColumnRowType { j } { Q[j] }

\NewTblrColumnType { t } [1] { Q[t,wd=#1] }
\NewTblrColumnType { p } [1] { Q[p,wd=#1] }
\NewTblrColumnType { m } [1] { Q[m,wd=#1] }
\NewTblrColumnType { b } [1] { Q[b,wd=#1] }
\NewTblrColumnType { h } [1] { Q[h,wd=#1] }
\NewTblrColumnType { f } [1] { Q[f,wd=#1] }

\NewTblrRowType { t } [1] { Q[t,ht=#1] }
\NewTblrRowType { p } [1] { Q[p,ht=#1] }
\NewTblrRowType { m } [1] { Q[m,ht=#1] }
\NewTblrRowType { b } [1] { Q[b,ht=#1] }
\NewTblrRowType { h } [1] { Q[h,ht=#1] }
\NewTblrRowType { f } [1] { Q[f,ht=#1] }

\NewTblrColumnRowType { X } [1][] { Q[co=1,#1] }

\NewTblrColumnRowType { ! } [1] { |[text={#1}] }
\NewTblrColumnRowType { @ } [1] { <[Opt]{} |[text={#1}] >[Opt]{} }
\NewTblrColumnRowType { * } [2] { \prg_replicate:nn {#1} {#2} }

\cs_new_protected:Npn \__tblr_parse_colrow_spec:nn #1 #2
  {
    \tl_gset:Nn \g__tblr_column_or_row_tl {#1}
    \tl_gset:Nn \g__tblr_expanded_colrow_spec_tl {#2}
    \__tblr_expand_colrow_spec:N \g__tblr_expanded_colrow_spec_tl
    \__tblr_execute_colrow_spec:N \g__tblr_expanded_colrow_spec_tl
  }

%% Expand defined column/row types

\cs_new_protected:Npn \__tblr_expand_colrow_spec:N #1
  {
    \bool_do_until:Nn \g__tblr_colrow_spec_expand_stop_bool
      {
        \LogTblrTracing { colspec, rowspec }
        \bool_gset_true:N \g__tblr_colrow_spec_expand_stop_bool
        \tl_set_eq:NN \l_tmpa_tl #1
        \tl_gclear:N #1
        \exp_last_unbraced:NV
          \__tblr_expand_colrow_spec_next:N \l_tmpa_tl \scan_stop:
      }
  }

\msg_new:nnn { tabularray } { unexpandable-colrow-type }
  { Unexpandable ~ command ~ #2 inside ~ #1 ~ type! }

\msg_new:nnn { tabularray } { unknown-colrow-type }
  { Unknown ~ #1 ~ type ~ #2! }
```

```
\cs_new_protected:Npn \__tblr_expand_colrow_spec_next:N #1
  {
    \token_if_eq_catcode:NNTF #1 \scan_stop:
      {
        \token_if_eq_meaning:NNF #1 \scan_stop:
          {
            \msg_error:nnVn { tabularray } { unexpandable-colrow-type }
              \g__tblr_column_or_row_tl {#1}
          }
      }
      {
        \str_if_in:cnTF { gTblrUsed \g__tblr_column_or_row_tl TypeStr } {#1}
          {
            %% Note that #1 may be an active character (see issue #58)
            \cs:w tblr_ \g__tblr_column_or_row_tl _type_ \token_to_str:N #1 \cs_end:
          }
          {
            \msg_error:nnVn { tabularray } { unknown-colrow-type }
              \g__tblr_column_or_row_tl {#1}
            \str_log:c { gTblrUsed \g__tblr_column_or_row_tl TypeStr }
          }
      }
  }

%% Execute primitive column/row types

\cs_new_protected:Npn \__tblr_execute_colrow_spec:N #1
  {
    \tl_if_eq:NnTF \g__tblr_column_or_row_tl { Row }
      { \int_set:Nn \c@rownum {1} }
      { \int_set:Nn \c@colnum {1} }
    \exp_last_unbraced:NV \__tblr_execute_colrow_spec_next:N #1 \scan_stop:
  }

\cs_new_protected:Npn \__tblr_execute_colrow_spec_next:N #1
  {
    \token_if_eq_meaning:NNF #1 \scan_stop:
      { \cs:w tblr_primitive_ \g__tblr_column_or_row_tl _type_  #1 \cs_end: }
  }
```

9.19 Set environments and new environments

```
\tl_new:N \l__tblr_initial_tblr_outer_tl
\tl_set:Nn \l__tblr_initial_tblr_outer_tl
  {
    halign = c, baseline = m, headsep = 6pt, footsep = 6pt,
    presep = 1.5\bigskipamount, postsep = 1.5\bigskipamount,
  }

%% #1: env name; #2: specifications
\NewDocumentCommand \SetTblrInner { O{tblr} m }
  {
    \clist_map_inline:nn {#1}
      { \tl_put_right:cn { l__tblr_default_ ##1 _inner_tl } { , #2 } }
    \ignorespaces
  }
```

```
\cs_new_eq:NN \SetTblrDefault \SetTblrInner

%% #1: env name; #2: specifications
\NewDocumentCommand \SetTblrOuter { O{tblr} m }
  {
    \clist_map_inline:nn {#1}
      { \tl_put_right:cn { l__tblr_default_ ##1 _outer_tl } { , #2 } }
    \ignorespaces
  }

%% #1: env name
\NewDocumentCommand \NewTblrEnviron { m }
  {
    \NewDocumentEnvironment {#1} { O{} m +b }
      {
        \__tblr_environ_code:nnnn {#1} {##1} {##2} {##3}
      } { }
    \tl_new:c { l__tblr_default_ #1 _inner_tl }
    \tl_new:c { l__tblr_default_ #1 _outer_tl }
    \tl_set_eq:cN { l__tblr_default_ #1 _outer_tl } \l__tblr_initial_tblr_outer_tl
  }

%% Create tblr and longtblr environments
\NewTblrEnviron { tblr }
\NewTblrEnviron { longtblr }
\SetTblrOuter [ longtblr ] { long }
\NewTblrEnviron { talltblr }
\SetTblrOuter [ talltblr ] { tall }

\tl_new:N \l__tblr_env_name_tl
\bool_new:N \l__tblr_math_mode_bool

%% Main environment code
%% We need to add \group_align_safe_begin: and \group_align_safe_end:
%% to make tabularray correctly nest in align environment (see issue #143)
\cs_new_protected:Npn \__tblr_environ_code:nnnn #1 #2 #3 #4
  {
    \group_align_safe_begin:
    \int_gincr:N \c@tblrcount
    \tl_set:Nn \l__tblr_env_name_tl {#1}
    \mode_if_math:TF
      { \bool_set_true:N \l__tblr_math_mode_bool }
      { \bool_set_false:N \l__tblr_math_mode_bool }
    \__tblr_builder:nnn {#2} {#3} {#4}
    \group_align_safe_end:
  }

\bool_new:N \lTblrMeasuringBool

\tl_new:N \l__tblr_inner_spec_tl
\cs_set_eq:NN \__tblr_hook_parse_inner_spec_before: \prg_do_nothing:

%% Read, split and build the table
\cs_new_protected:Npn \__tblr_builder:nnn #1 #2 #3
  {
    \int_gincr:N \gTblrLevelInt
```

```
    \__tblr_hook_use:n { trial/before }
    \bool_set_true:N \lTblrMeasuringBool
    \__tblr_clear_prop_lists:
    \__tblr_clear_spec_lists:
    \LogTblrTracing { step = init ~ table ~ outer ~ spec}
    \__tblr_init_table_outer_spec:
    \LogTblrTracing { step = parse ~ table ~ options }
    \__tblr_parse_table_option:n {#1}
    \LogTblrTracing { outer }
    \LogTblrTracing { option }
    \__tblr_enable_table_commands:
    \LogTblrTracing { step = split ~ table}
    \__tblr_split_table:n {#3}
    \LogTblrTracing { command }
    \bool_if:NT \g__tblr_use_intarray_bool { \__tblr_init_table_data: }
    \LogTblrTracing { step = init ~ table ~ inner ~ spec}
    \__tblr_init_table_inner_spec:
    \LogTblrTracing { inner }
    \LogTblrTracing { step = parse ~ table ~ inner ~ spec}
    \tl_set:Nn \l__tblr_inner_spec_tl {#2}
    \__tblr_hook_parse_inner_spec_before:
    \exp_args:NV \__tblr_parse_table_spec:n \l__tblr_inner_spec_tl
    \LogTblrTracing { step = execute ~ table ~ commands}
    \__tblr_execute_table_commands:
    \__tblr_disable_table_commands:
    \__tblr_functional_calculation:
    \LogTblrTracing { step = calculate ~ cell ~ and ~ line ~ sizes}
    \__tblr_enable_content_commands:
    \__tblr_calc_cell_and_line_sizes:
    \bool_set_false:N \lTblrMeasuringBool
    \__tblr_hook_use:n { trial/after }
    \LogTblrTracing { step = build ~ the ~ whole ~ table}
    \__tblr_build_whole:
    \int_gdecr:N \gTblrLevelInt
  }
```

9.20 Split table contents

```
\tl_new:N \l__tblr_body_tl
\seq_new:N \l__tblr_lines_seq

%% Split table content to cells and store them
%% #1: table content
\cs_new_protected:Npn \__tblr_split_table:n #1
  {
    \__tblr_child_split_table_before:
    \tl_set:Nn \l__tblr_body_tl {#1}
    \__tblr_modify_table_body:
    \int_zero:N \c@rowcount
    \int_zero:N \c@colcount
    \__tblr_split_table_to_lines:NN \l__tblr_body_tl \l__tblr_lines_seq
    \__tblr_split_lines_to_cells:N \l__tblr_lines_seq
    \__tblr_child_split_table_after:
  }

\tl_new:N \l__tblr_expand_tl
```

```
\cs_set_eq:NN \__tblr_hook_split_before: \prg_do_nothing:

\cs_new_protected:Npn \__tblr_modify_table_body:
  {
    \__tblr_hook_split_before:
    \tl_set:Ne \l__tblr_expand_tl { \__tblr_spec_item:nn { outer } { expand } }
    \tl_map_inline:Nn \l__tblr_expand_tl
      {
        \__tblr_expand_table_body:NN \l__tblr_body_tl ##1
      }
  }

%% Expand every occurrence of the specified macro once
%% #1: tl with table content; #2: macro to be expanded
\cs_new_protected:Npn \__tblr_expand_table_body:NN #1 #2
  {
    \tl_set_eq:NN \l_tmpa_tl #1
    \tl_clear:N #1
    \cs_set_protected:Npn \__tblr_expand_table_body_aux:w ##1 #2
      {
        \tl_put_right:Nn #1 {##1}
        \peek_meaning:NTF \q_stop
          { \use_none:n }
          { \exp_last_unbraced:NV \__tblr_expand_table_body_aux:w #2 }
      }
    \exp_last_unbraced:NV \__tblr_expand_table_body_aux:w \l_tmpa_tl #2 \q_stop
  }

%% Split tl #3 into items separated by tl #2, and store the result in seq #1.
%% Spaces on both side of each item and outer braces around each item are kept.
%% We insert \prg_do_nothing: before each item to avoid losing outermost braces.
\cs_new_protected:Npn \__tblr_seq_set_split:Nnn #1 #2 #3
  {
    \seq_clear:N #1
    \cs_set_protected:Npn \__tblr_seq_set_split_aux:Nw ##1 ##2 #2
      {
        \tl_if_eq:nnF { \prg_do_nothing: \c_novalue_tl } { ##2 }
          {
            \seq_put_right:No ##1 { ##2 }
            \__tblr_seq_set_split_aux:Nw ##1 \prg_do_nothing:
          }
      }
    \__tblr_seq_set_split_aux:Nw #1 \prg_do_nothing: #3 #2 \c_novalue_tl #2
  }
\cs_generate_variant:Nn \__tblr_seq_set_split:Nnn { NnV }

%% Split tl #3 into items separated by tl #2, and store the result in seq #1.
%% Spaces on both side of items and outer braces around items are kept.
%% And we prevent splitting inside the body of an environment.
\cs_new_protected:Npn \__tblr_seq_set_split_keep_braces_envs:Nnn #1 #2 #3
  {
    \__tblr_seq_set_split_keep_envs_aux:NnnN
      #1 { #2 }  { #3 } \__tblr_seq_set_split:Nnn
  }
\cs_generate_variant:Nn \__tblr_seq_set_split_keep_braces_envs:Nnn { NnV }

%% Split tl #3 into items separated by tl #2, and store the result in seq #1.
```

```
%% Spaces on both side of items and outer braces around items are removed.
%% And we prevent splitting inside the body of an environment.
\cs_new_protected:Npn \__tblr_seq_set_split_keep_envs:Nnn #1 #2 #3
  {
    \__tblr_seq_set_split_keep_envs_aux:NnnN
      #1 { #2 }  { #3 } \seq_set_split:Nnn
  }
\cs_generate_variant:Nn \__tblr_seq_set_split_keep_envs:Nnn { NnV }

\seq_new:N \l__tblr_split_raw_seq
\seq_new:N \l__tblr_split_temp_seq
\seq_new:N \l__tblr_split_item_seq
\int_new:N \l__tblr_split_balance_int

\cs_new_protected:Npn \__tblr_seq_set_split_keep_envs_aux:NnnN #1 #2 #3 #4
  {
    #4 \l__tblr_split_raw_seq { #2 }  { #3 }
    \seq_clear:N #1
    \seq_clear:N \l__tblr_split_item_seq
    \seq_map_inline:Nn \l__tblr_split_raw_seq
      {
        \seq_put_right:Nn \l__tblr_split_item_seq { ##1 }
        \seq_set_split:Nnn \l__tblr_split_temp_seq { \begin } { ##1 }
        \int_add:Nn \l__tblr_split_balance_int
          { \seq_count:N \l__tblr_split_temp_seq }
        \seq_set_split:Nnn \l__tblr_split_temp_seq { \end } { ##1 }
        \int_sub:Nn \l__tblr_split_balance_int
          { \seq_count:N \l__tblr_split_temp_seq }
        \int_compare:nNnT { \l__tblr_split_balance_int } = { 0 }
          {
            \seq_put_right:Ne #1 { \seq_use:Nn \l__tblr_split_item_seq { #2 } }
            \seq_clear:N \l__tblr_split_item_seq
          }
      }
  }

%% Split table content to a sequence of lines
%% #1: tl with table contents, #2: resulting sequence of lines
\cs_new_protected:Npn \__tblr_split_table_to_lines:NN #1 #2
  {
    \__tblr_seq_set_split_keep_braces_envs:NnV \l_tmpa_seq { \\ } #1
    \seq_clear:N #2
    \seq_pop_left:NNT \l_tmpa_seq \l__tblr_a_tl
      {
        %% skip detection for "*" and "[" for the 1st row (issue #598)
        \seq_put_right:NV #2 \l__tblr_a_tl
        %% apply normal detection for following rows
        \seq_map_inline:Nn \l_tmpa_seq
          {
            \tl_if_head_eq_meaning:nNTF {##1} *
              {
                \tl_set:Nn \l__tblr_b_tl { \hborder { pagebreak = no } }
                \tl_set:Ne \l__tblr_c_tl { \tl_tail:n {##1} }
                %% Ignore spaces between * and [dimen]
                \tl_trim_spaces:N \l__tblr_c_tl
                \tl_if_head_eq_meaning:VNT \l__tblr_c_tl [
                  {
```

```
                \tl_put_right:Nn \l__tblr_b_tl { \RowBefore@AddBelowSep }
              }
            \tl_put_right:NV \l__tblr_b_tl \l__tblr_c_tl
            \seq_put_right:NV #2 \l__tblr_b_tl
          }
          {
            \tl_if_head_eq_meaning:nNTF { ##1 } [
              { \seq_put_right:Nn #2 { \RowBefore@AddBelowSep ##1 } }
              { \seq_put_right:Nn #2 { ##1 } }
          }
      }
    }
    \int_set:Nn \c@rowcount { \seq_count:N #2 }
  }

%% Treat \\[dimen] command
\NewTblrTableCommand \RowBefore@AddBelowSep [1] []
  {
    \IfValueT { #1 }
      {
        \__tblr_data_gadd_dimen_value:nene { row }
          { \int_eval:n {\c@rownum - 1} } { belowsep } {#1}
      }
  }

%% Split table lines to cells and store them
%% #1: sequence of lines
\cs_new_protected:Npn \__tblr_split_lines_to_cells:N #1
  {
    \seq_map_indexed_function:NN #1 \__tblr_split_one_line:nn
    \LogTblrTracing { text }
  }

%% Split one line into cells and store them
%% #1: row number, #2 the line text
\cs_new_protected:Npn \__tblr_split_one_line:nn #1 #2
  {
    \__tblr_seq_set_split_keep_braces_envs:Nnn \l_tmpa_seq { & } { #2 }
    \int_set:Nn \c@rownum {#1}
    \int_zero:N \c@colnum
    \seq_map_inline:Nn \l_tmpa_seq
      {
        \tl_set:Nn \l_tmpa_tl { ##1 }
        \__tblr_trim_par_space_tokens_left:N \l_tmpa_tl
        \int_incr:N \c@colnum
        \__tblr_extract_table_commands:N \l_tmpa_tl
        \__tblr_trim_par_space_tokens:N \l_tmpa_tl
        \__tblr_spec_gput:neV { text } { [#1][\int_use:N \c@colnum] } \l_tmpa_tl
      }
    %% Decrease row count by 1 if the last row has only one empty cell text
    %% We need to do it here since the > or < column type may add text to cells
    \bool_lazy_all:nTF
      {
        { \int_compare_p:nNn {#1} = {\c@rowcount} }
        { \int_compare_p:nNn {\c@colnum} = {1} }
        { \tl_if_empty_p:N \l_tmpa_tl }
        { \int_compare_p:nNn {\c@rowcount} > {1} } % at least one row (issue #326)
```

```
      }
      { \int_decr:N \c@rowcount }
      {
        \__tblr_prop_gput:nne
          {row} { [#1] / cell-number } { \int_use:N \c@colnum }
        \int_compare:nT { \c@colnum > \c@colcount }
          {
            \int_set_eq:NN \c@colcount \c@colnum
          }
      }
  }

\cs_new_protected:Npn \__tblr_trim_par_space_tokens_left:N #1
  {
    \tl_trim_spaces:N #1
    \__tblr_remove_head_par:N #1
    \tl_trim_spaces:N #1
  }

\tl_new:N \l__tblr_trim_temp_tl

\cs_new_protected:Npn \__tblr_trim_par_space_tokens:N #1
  {
    \tl_trim_spaces:N #1
    \__tblr_remove_head_par:N #1
    \tl_set_eq:NN \l__tblr_trim_temp_tl #1
    \tl_reverse:N \l__tblr_trim_temp_tl
    \__tblr_remove_head_par:N \l__tblr_trim_temp_tl
    \tl_reverse:N \l__tblr_trim_temp_tl
    \tl_set_eq:NN #1 \l__tblr_trim_temp_tl
    \tl_trim_spaces:N #1
  }

\cs_new_protected:Npn \__tblr_remove_head_par:N #1
  {
    \tl_if_head_eq_meaning:VNT #1 \par { \tl_set:Ne #1 { \tl_tail:N #1 } }
  }
```

9.21 Extract table commands from cell text

```
%% Extract table commands defined with \NewTblrTableCommand from cell text

\tl_new:N \l__tblr_saved_table_commands_before_cell_text_tl
\tl_new:N \l__tblr_saved_cell_text_after_table_commands_tl

\cs_new_protected:Npn \__tblr_extract_table_commands:N #1
  {
    % We need to execute \SetChild commands before parsing inner specs,
    % but execute other table commands after parsing inner specs.
    \__tblr_child_extract_index_command:N #1
    \tl_clear:N \l__tblr_saved_table_commands_before_cell_text_tl
    \tl_clear:N \l__tblr_saved_cell_text_after_table_commands_tl
    \exp_last_unbraced:NV \__tblr_extract_table_commands_next: #1 \q_stop
    \tl_if_empty:NF \l__tblr_saved_table_commands_before_cell_text_tl
      {
```

```
        \__tblr_prop_gput:neV { command }
          {[\int_use:N \c@rownum][\int_use:N \c@colnum]}
          \l__tblr_saved_table_commands_before_cell_text_tl
      }
    \tl_set_eq:NN #1 \l__tblr_saved_cell_text_after_table_commands_tl
  }

\cs_new_protected:Npn \__tblr_extract_table_commands_next:
  {
    \peek_after:Nw \__tblr_extract_table_commands_next_peek:
  }

\cs_new_protected:Npn \__tblr_extract_table_commands_next_peek:
  {
    \token_if_group_begin:NTF \l_peek_token
      {
        \__tblr_save_real_cell_text:
      }
      {
        \__tblr_extract_table_commands_next_real:n
      }
  }

%% #1 maybe a single token or multiple tokens from a pair of braces
\cs_new_protected:Npn \__tblr_extract_table_commands_next_real:n #1
  {
    \tl_if_single_token:nTF {#1}
      {
        \clist_if_in:NnTF \g__tblr_table_commands_clist { #1 }
          { \__tblr_extract_one_table_command:N #1 }
          {
            \token_if_eq_meaning:NNF #1 \q_stop
              { \__tblr_save_real_cell_text: #1 }
          }
      }
      { \__tblr_save_real_cell_text: {#1} }
  }

\cs_new_protected:Npn \__tblr_extract_one_table_command:N #1
  {
    \int_set:Nn \l__tblr_a_int
      { \tl_use:c { g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl } }
    \tl_put_right:Nn \l__tblr_saved_table_commands_before_cell_text_tl {#1}
    \int_compare:nNnTF {\l__tblr_a_int} < {0}
      {
        \int_set:Nn \l__tblr_a_int { \int_abs:n {\l__tblr_a_int} - 1 }
        \peek_charcode:NTF [
          { \__tblr_extract_table_command_arg_o:w }
          { \__tblr_extract_table_command_arg_next: }
      }
      { \__tblr_extract_table_command_arg_next: }
  }

\cs_new_protected:Npn \__tblr_extract_table_command_arg_o:w [#1]
  {
    \tl_put_right:Nn \l__tblr_saved_table_commands_before_cell_text_tl { [#1] }
    \__tblr_extract_table_command_arg_next:
```

```
  }

\cs_new_protected:Npn \__tblr_extract_table_command_arg_m:n #1
  {
    \tl_put_right:Nn \l__tblr_saved_table_commands_before_cell_text_tl { {#1} }
    \__tblr_extract_table_command_arg_next:
  }

\cs_new_protected:Npn \__tblr_extract_table_command_arg_next:
  {
    \int_compare:nNnTF {\l__tblr_a_int} > {0}
      {
        \int_decr:N \l__tblr_a_int
        \__tblr_extract_table_command_arg_m:n
      }
      { \__tblr_extract_table_commands_next: }
  }

%% We prepend \prg_do_nothing: to avoid losing outermost braces.
\cs_new_protected:Npn \__tblr_save_real_cell_text:
  {
    \__tblr_save_real_cell_text_aux:w \prg_do_nothing:
  }

\cs_new_protected:Npn \__tblr_save_real_cell_text_aux:w #1 \q_stop
  {
    \tl_set:No \l__tblr_saved_cell_text_after_table_commands_tl {#1}
  }
```

9.22 Initialize table inner specifications

```
\prop_new:N \g__tblr_initial_table_prop
\prop_new:N \g__tblr_initial_rows_prop
\prop_new:N \g__tblr_initial_columns_prop
\prop_new:N \g__tblr_initial_cells_prop
\prop_new:N \g__tblr_initial_hlines_prop
\prop_new:N \g__tblr_initial_vlines_prop

\prop_gset_from_keyval:Nn \g__tblr_initial_table_prop
  {
    stretch = 1,
    rulesep = 2pt,
  }

\prop_gset_from_keyval:Nn \g__tblr_initial_rows_prop
  {
    abovesep = 2pt,
    belowsep = 2pt,
    @row-height = 0pt,
    @row-head = 0pt,
    @row-foot = 0pt,
    @row-upper = 0pt,
    @row-lower = 0pt,
  }
```

```
\prop_gset_from_keyval:Nn \g__tblr_initial_columns_prop
  {
    leftsep = 6pt,
    rightsep = 6pt,
    width = -1pt, % column width unset
    coefficient = 0, % column coefficient unset
    @col-width = 0pt,
  }

\prop_gset_from_keyval:Nn \g__tblr_initial_cells_prop
  {
    halign = j,
    valign = t,
    width = -1pt, % cell width unset
    rowspan = 1,
    colspan = 1,
    omit = 0,
  }

\prop_gset_from_keyval:Nn \g__tblr_initial_hlines_prop
  {
    @hline-count = 0,
  }

\prop_gset_from_keyval:Nn \g__tblr_initial_vlines_prop
  {
    @vline-count = 0,
  }

\tl_new:N \l__tblr_inner_spec_measure_tl
\tl_set:Nn \l__tblr_inner_spec_measure_tl { hbox }

\cs_new_protected:Npn \__tblr_init_table_inner_spec:
  {
    \prop_map_inline:Nn \g__tblr_initial_table_prop
      {
        \__tblr_prop_gput:nen { inner } { ##1 } {##2}
      }
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \prop_map_inline:Nn \g__tblr_initial_rows_prop
          {
            \__tblr_data_gput:nVnn { row } \l__tblr_i_tl {##1} {##2}
          }
        \prop_map_inline:Nn \g__tblr_initial_hlines_prop
          {
            \__tblr_spec_gput:nen { hline } { [\l__tblr_i_tl] / ##1 } {##2}
          }
        \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
          {
            \prop_map_inline:Nn \g__tblr_initial_cells_prop
              {
                \__tblr_data_gput:neeen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } {##1} {##2}
              }
          }
      }
```

```
    \prop_map_inline:Nn \g__tblr_initial_hlines_prop
      {
        \__tblr_spec_gput:nen { hline }
          { [\int_eval:n { \c@rowcount + 1}] / ##1 } {##2}
      }
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \prop_map_inline:Nn \g__tblr_initial_columns_prop
          {
            \__tblr_data_gput:nenn { column } { \l__tblr_j_tl } {##1} {##2}
          }
        \prop_map_inline:Nn \g__tblr_initial_vlines_prop
          {
            \__tblr_spec_gput:nen { vline } { [\l__tblr_j_tl] / ##1 } {##2}
          }
      }
    \prop_map_inline:Nn \g__tblr_initial_vlines_prop
      {
        \__tblr_spec_gput:nen { vline }
          { [\int_eval:n { \c@colcount + 1}] / ##1 } {##2}
      }
    \__tblr_keys_set:nv { table/inner }
      { l__tblr_default_ \l__tblr_env_name_tl _inner_tl }
  }
```

9.23 Parse table inner specifications

```
%% We must define these keys to make \__tblr_keys_if_exist work
\__tblr_keys_define:nn { table/inner }
  {
    column  .code:n = ,
    row     .code:n = ,
    cell    .code:n = ,
    hline   .code:n = ,
    vline   .code:n = ,
    hborder .code:n = ,
    vborder .code:n =
  }

\str_new:N \g__tblr_name_str

\__tblr_keys_define:nn { table/inner }
  {
    name .code:n = \__tblr_keys_gput:nn { name } {#1},
    colspec .code:n = \__tblr_parse_colrow_spec:nn { Column } {#1},
    rowspec .code:n = \__tblr_parse_colrow_spec:nn { Row } {#1},
    width .code:n = \__tblr_keys_gput:ne { width } { \dim_eval:n {#1} },
    hspan .code:n = \__tblr_keys_gput:nn { hspan } {#1},
    vspan .code:n = \__tblr_keys_gput:nn { vspan } {#1},
    stretch .code:n = \__tblr_keys_gput:nn { stretch } {#1},
    columns .code:n = \__tblr_set_every_column_aux:n {#1},
    rows    .code:n = \__tblr_set_every_row_aux:n {#1},
    cells   .code:n = \__tblr_set_every_cell_aux:n {#1},
    hlines  .code:n = \__tblr_set_every_hline_aux:n {#1},
    vlines  .code:n = \__tblr_set_every_vline_aux:n {#1},
    leftsep .code:n = \tblr_set_every_column:nn { } { leftsep = #1 },
```

```
    rightsep .code:n = \tblr_set_every_column:nn { } { rightsep = #1 },
    colsep .meta:n = { leftsep = #1, rightsep = #1 },
    abovesep .code:n = \tblr_set_every_row:nn { } { abovesep = #1 },
    belowsep .code:n = \tblr_set_every_row:nn { } { belowsep = #1 },
    rowsep .meta:n = { abovesep = #1, belowsep = #1 },
    rulesep .code:n = \__tblr_keys_gput:nn { rulesep } {#1},
    rowhead .code:n = \__tblr_keys_gput:nn { rowhead } {#1},
    rowfoot .code:n = \__tblr_keys_gput:nn { rowfoot } {#1},
    delimiter .code:n = \__tblr_set_delimiter:n {#1},
    baseline .code:n = \__tblr_outer_gput_spec:nn { baseline } {#1},
    unknown .code:n = \__tblr_table_special_key:Vn \l_keys_key_str {#1},
  }

\msg_new:nnn { tabularray } { unknown-inner-key }
  { Unknown ~ inner ~ key ~ name ~ '#1'. }

\cs_new_protected:Npn \__tblr_table_special_key:nn #1 #2
  {
    \__tblr_key_split_name_args:n {#1}
    \__tblr_keys_if_exist:nVTF { table/inner } \l__tblr_key_split_name_str
      {
        \use:c { __tblr_set_ \l__tblr_key_split_name_str _aux:Vn }
          \l__tblr_key_split_args_tl {#2}
      }
      {
        \msg_error:nnV { tabularray } { unknown-inner-key }
          \l__tblr_key_split_name_str
      }
  }
\cs_generate_variant:Nn \__tblr_table_special_key:nn { Vn }

%% If the first key name is known, treat #1 as table spec;
%% otherwise, treat #1 as colspec.

\cs_new_protected:Npn \__tblr_parse_table_spec:n #1
  {
    \__tblr_keyval_extract_first_name:n {#1}
    \__tblr_keys_if_exist:nVTF { table/inner } \l__tblr_key_split_name_str
      { \__tblr_keys_set:nn { table/inner } {#1} }
      { \__tblr_parse_colrow_spec:nn { Column } {#1} }
    \str_gset:Ne \g__tblr_name_str { \__tblr_prop_item:ne { inner } { name } }
    \str_if_empty:NT \g__tblr_name_str
      { \str_gset:Ne \g__tblr_name_str { @ \int_use:N \c@tblrcount } }
  }

\cs_new_protected:Npn \__tblr_keys_gput:nn #1 #2
  {
    \__tblr_prop_gput:nnn { inner } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_keys_gput:nn { ne }

\__tblr_keys_define:nn { table/delimiter }
  {
    left .code:n  = \__tblr_keys_gput:nn { delim-left } { \left #1 },
    right .code:n = \__tblr_keys_gput:nn { delim-right } { \right #1 }
  }
```

```
\cs_new_protected:Npn \__tblr_set_delimiter:n #1
  {
    \__tblr_keys_set:nn { table/delimiter } {#1}
  }
```

9.24 Initialize and parse table outer specifications

```
\msg_new:nnn { tabularray } { used-theme-name }
  { theme ~ name ~ #1 ~ has ~ been ~ used! }

%% #1: theme names; #2: template and style commands
\NewDocumentCommand \NewTblrTheme { m +m }
  {
    \tl_if_exist:cTF { g__tblr_theme_ #1 _code_tl }
      { \msg_error:nnn { tabularray } { used-theme-name } { #1 } }
      {
        \tl_set:cn { g__tblr_theme_ #1 _code_tl } {#2}
        \ignorespaces
      }
  }

\cs_new_protected:Npn \__tblr_use_theme:n #1
  {
    \ignorespaces
    \tl_use:c { g__tblr_theme_ #1 _code_tl }
  }

\cs_new_protected:Npn \__tblr_init_table_outer_spec:
  {
    \__tblr_keys_set:nv { table/outer }
      { l__tblr_default_ \l__tblr_env_name_tl _outer_tl }
  }

\cs_new_protected:Npn \__tblr_parse_table_option:n #1
  {
    \__tblr_keys_set:nn { table/outer } {#1}
  }

%% We must define these keys to make \__tblr_keys_if_exist work
\__tblr_keys_define:nn { table/outer }
  {
    note    .code:n = ,
    remark .code:n = ,
    more    .code:n =
  }

\__tblr_keys_define:nn { table/outer }
  {
    long     .code:n = \__tblr_outer_gput_spec:nn { portrait } { long },
    tall     .code:n = \__tblr_outer_gput_spec:nn { portrait } { tall },
    halign   .code:n = \__tblr_outer_gput_spec:nn { halign } {#1},
    baseline .code:n = \__tblr_outer_gput_spec:nn { baseline } {#1},
    l        .meta:n = { halign = l },
    c        .meta:n = { halign = c },
    r        .meta:n = { halign = r },
```

```
    t       .meta:n = { baseline = t },
    T       .meta:n = { baseline = T },
    m       .meta:n = { baseline = m },
    M       .meta:n = { baseline = M },
    b       .meta:n = { baseline = b },
    B       .meta:n = { baseline = B },
    valign  .meta:n = { baseline = #1 }, % obsolete, will be removed some day
    expand  .code:n = \__tblr_outer_gput_spec:nn { expand } {#1},
    expand+ .code:n = \__tblr_outer_gconcat_spec:nn { expand } {#1},
    headsep .code:n = \__tblr_outer_gput_spec:nn { headsep } {#1},
    footsep .code:n = \__tblr_outer_gput_spec:nn { footsep } {#1},
    presep  .code:n = \__tblr_outer_gput_spec:nn { presep }  {#1},
    postsep .code:n = \__tblr_outer_gput_spec:nn { postsep } {#1},
    theme   .code:n = \__tblr_use_theme:n {#1},
    caption .code:n = \__tblr_outer_gput_spec:nn { caption } {#1},
    entry   .code:n = \__tblr_outer_gput_spec:nn { entry } {#1},
    label   .code:n = \__tblr_outer_gput_spec:nn { label } {#1},
    unknown .code:n = \__tblr_table_option_key:Vn \l_keys_key_str {#1},
  }

\cs_new_protected:Npn \__tblr_outer_gput_spec:nn #1 #2
  {
    \__tblr_spec_gput:nen { outer } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_outer_gput_spec:nn { ne }

\cs_new_protected:Npn \__tblr_outer_gconcat_spec:nn #1 #2
  {
    \__tblr_outer_gput_spec:ne
      {#1} { \__tblr_spec_item:nn { outer } { #1 } \exp_not:n { #2 } }
  }

\msg_new:nnn { tabularray } { unknown-outer-key }
  { Unknown ~ outer ~ key ~ name ~ '#1'. }

\cs_new_protected:Npn \__tblr_table_option_key:nn #1 #2
  {
    \__tblr_key_split_name_args:n {#1}
    \__tblr_keys_if_exist:nVTF { table/outer } \l__tblr_key_split_name_str
      {
        % remove a pair of outer braces
        \tl_set:Ne \l__tblr_key_split_args_tl
          { \tl_head:N \l__tblr_key_split_args_tl }
        \use:c { __tblr_outer_gput_ \l__tblr_key_split_name_str :Vn }
          \l__tblr_key_split_args_tl {#2}
      }
      {
        \msg_error:nnV { tabularray } { unknown-outer-key }
          \l__tblr_key_split_name_str
      }
  }
\cs_generate_variant:Nn \__tblr_table_option_key:nn { Vn }

\cs_new_protected:Npn \__tblr_outer_gput_note:nn #1 #2
  {
    \__tblr_prop_gput:nnn { note } {#1} {#2}
  }
```

```
\cs_generate_variant:Nn \__tblr_outer_gput_note:nn { Vn }

\cs_new_protected:Npn \__tblr_outer_gput_remark:nn #1 #2
  {
    \__tblr_prop_gput:nnn { remark } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_outer_gput_remark:nn { Vn }

\cs_new_protected:Npn \__tblr_outer_gput_more:nn #1 #2
  {
    \__tblr_prop_gput:nnn { more } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_outer_gput_more:nn { Vn }
```

9.25 Typeset and calculate sizes

```
%% Calculate the width and height for every cell and border

\cs_new_protected:Npn \__tblr_calc_cell_and_line_sizes:
  {
    \__tblr_prepare_stretch:
    \__tblr_calculate_line_sizes:
    \__tblr_calculate_cell_sizes:
    \LogTblrTracing { cell, row, column, hline, vline }
    \__tblr_compute_extendable_column_width:
    \__tblr_adjust_sizes_for_span_cells:
  }

%% prepare stretch option of the table
\fp_new:N \l__tblr_stretch_fp
\dim_new:N \l__tblr_strut_dp_dim
\dim_new:N \l__tblr_strut_ht_dim
\cs_new_protected:Npn \__tblr_prepare_stretch:
  {
    \fp_set:Nn \l__tblr_stretch_fp
      { \__tblr_prop_item:nn { inner } { stretch } }
    \fp_compare:nNnTF \l__tblr_stretch_fp > \c_zero_fp
      {
        \dim_set:Nn \l__tblr_strut_dp_dim
          { \fp_use:N \l__tblr_stretch_fp \box_dp:N \strutbox }
        \dim_set:Nn \l__tblr_strut_ht_dim
          { \fp_use:N \l__tblr_stretch_fp \box_ht:N \strutbox }
        \cs_set_eq:NN \__tblr_leave_vmode: \mode_leave_vertical:
        \cs_set_eq:NN \__tblr_process_stretch: \__tblr_process_stretch_real:
      }
      {
        \cs_set_eq:NN \__tblr_process_stretch: \prg_do_nothing:
        \fp_compare:nNnTF \l__tblr_stretch_fp < \c_zero_fp
          {
            % for lists (see issue #99)
            \cs_set_eq:NN \__tblr_leave_vmode: \@setminipage
          }
          { \cs_set_eq:NN \__tblr_leave_vmode: \mode_leave_vertical: }
      }
  }
\cs_new_eq:NN \__tblr_leave_vmode: \mode_leave_vertical:
```

```
\cs_new_protected:Npn \__tblr_process_stretch_real:
  {
    \dim_compare:nNnT \l__tblr_strut_dp_dim > { \box_dp:N \l_tmpb_box }
      {
        \box_set_dp:Nn \l_tmpa_box
          {
              \box_dp:N \l_tmpa_box
            - \box_dp:N \l_tmpb_box
            + \l__tblr_strut_dp_dim
          }
        \box_set_dp:Nn \l_tmpb_box { \l__tblr_strut_dp_dim }
      }
    \dim_compare:nNnT \l__tblr_strut_ht_dim > { \box_ht:N \l_tmpa_box }
      {
        \hbox_set:Nn \l_tmpa_box { \box_use:N \l_tmpa_box }
        \hbox_set:Nn \l_tmpb_box { \box_use:N \l_tmpb_box }
        \box_set_ht:Nn \l_tmpb_box
          {
              \box_ht:N \l_tmpb_box
            - \box_ht:N \l_tmpa_box
            + \l__tblr_strut_ht_dim
          }
        \box_set_ht:Nn \l_tmpa_box { \l__tblr_strut_ht_dim }
        %% return vbox for vertical-align: \c__tblr_middle_m_tl
        \vbox_set_top:Nn \l_tmpa_box { \box_use:N \l_tmpa_box }
        \vbox_set:Nn \l_tmpb_box { \box_use:N \l_tmpb_box }
      }
  }
\cs_new_eq:NN \__tblr_process_stretch: \__tblr_process_stretch_real:

%% Calculate the thickness for every hline and vline
\cs_new_protected:Npn \__tblr_calculate_line_sizes:
  {
    %% We need these two counters in executing hline and vline commands
    \int_zero:N \c@rownum
    \int_zero:N \c@colnum
    \int_step_inline:nn { \c@rowcount + 1 }
      {
        \int_incr:N \c@rownum
        \int_zero:N \c@colnum
        \int_step_inline:nn { \c@colcount + 1 }
          {
            \int_incr:N \c@colnum
            \int_compare:nNnT { ##1 } < { \c@rowcount + 1 }
              {
                \__tblr_measure_and_update_vline_size:nn { ##1 } { ####1 }
              }
            \int_compare:nNnT { ####1 } < { \c@colcount + 1 }
              {
                \__tblr_measure_and_update_hline_size:nn { ##1 } { ####1 }
              }
          }
      }
  }

%% Measure and update thickness of the vline
%% #1: row number, #2 column number
\cs_new_protected:Npn \__tblr_measure_and_update_vline_size:nn #1 #2
```

```
  {
    \dim_zero:N \l__tblr_w_dim
    \tl_set:Ne \l__tblr_n_tl
      { \__tblr_spec_item:ne { vline } { [#2] / @vline-count } }
    \int_compare:nNnT { \l__tblr_n_tl } > {0}
      {
        \tl_set:Ne \l__tblr_s_tl
          { \__tblr_prop_item:ne { inner } { rulesep } }
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \vbox_set_to_ht:Nnn \l__tblr_b_box {1pt}
              {
                \__tblr_get_vline_segment_child:nnnnn
                  {#1} {#2} {##1} {1pt} {1pt}
              }
            \tl_set:Ne \l__tblr_w_tl { \dim_eval:n { \box_wd:N \l__tblr_b_box } }
            \__tblr_spec_gput_if_larger:nee { vline }
              { [#2](##1) / @vline-width } { \l__tblr_w_tl }
            \dim_add:Nn \l__tblr_w_dim
              {
                \__tblr_spec_item:nn  { vline } { [#2](##1) / @vline-width }
              }
            \dim_add:Nn \l__tblr_w_dim { \l__tblr_s_tl }
          }
        \dim_add:Nn \l__tblr_w_dim { - \l__tblr_s_tl }
      }
    \__tblr_spec_gput_if_larger:nee { vline }
      { [#2]/ @vline-width } { \dim_use:N \l__tblr_w_dim }
  }

%% Get text of a vline segment
%% #1: row number, #2: column number; #3: index number; #4: height; #5: depth
%% We put all code inside a group to avoid conflicts of local variables
\cs_new_protected:Npn \__tblr_get_vline_segment_child:nnnnn #1 #2 #3 #4 #5
  {
    \group_begin:
    \tl_set:Ne \l__tblr_w_tl
      { \__tblr_spec_item:ne { vline } { [#1][#2](#3) / wd } }
    \tl_if_empty:NF \l__tblr_w_tl
      { \dim_set:Nn \lTblrDefaultVruleWidthDim { \l__tblr_w_tl } }
    \tl_set:Ne \l__tblr_d_tl
      { \__tblr_spec_item:ne { vline } { [#1][#2](#3) / @dash } }
    \tl_set:Ne \l__tblr_b_tl { \tl_tail:N \l__tblr_d_tl }
    \tl_if_head_eq_meaning:VNTF \l__tblr_d_tl \q__tblr_dash
      {
        \__tblr_get_vline_dash_style:N \l__tblr_b_tl
        \xleaders \l__tblr_b_tl \vfil
      }
      {
        %% When using text as vline, we need to omit abovepos and belowpos.
        \unskip
        \hbox_set:Nn \l__tblr_d_box
          {
            \bool_if:NTF \l__tblr_math_mode_bool
              { $ \l__tblr_b_tl $ } { \l__tblr_b_tl }
          }
        \box_set_ht:Nn \l__tblr_d_box {#4}
        \box_set_dp:Nn \l__tblr_d_box {#5}
```

```
        \box_use:N \l__tblr_d_box
        \vss
      }
    \group_end:
  }
\cs_generate_variant:Nn \__tblr_get_vline_segment_child:nnnnn { nnnee }

%% Measure and update thickness of the hline
%% #1: row number, #2 column number
\cs_new_protected:Npn \__tblr_measure_and_update_hline_size:nn #1 #2
  {
    \dim_zero:N \l__tblr_h_dim
    \tl_set:Ne \l__tblr_n_tl
      { \__tblr_spec_item:ne { hline } { [#1] / @hline-count } }
    \int_compare:nNnT { \l__tblr_n_tl } > {0}
      {
        \tl_set:Ne \l__tblr_s_tl
          { \__tblr_prop_item:ne { inner } { rulesep } }
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \hbox_set_to_wd:Nnn \l__tblr_b_box {1pt}
              { \__tblr_get_hline_segment_child:nnn {#1} {#2} {##1} }
            \tl_set:Ne \l__tblr_h_tl
              {
                \dim_eval:n
                  { \box_ht:N \l__tblr_b_box + \box_dp:N \l__tblr_b_box }
              }
            \__tblr_spec_gput_if_larger:nee { hline }
              { [#1](##1) / @hline-height } { \l__tblr_h_tl }
            \dim_add:Nn \l__tblr_h_dim
              {
                \__tblr_spec_item:nn  { hline } { [#1](##1) / @hline-height }
              }
            \dim_add:Nn \l__tblr_h_dim { \l__tblr_s_tl }
          }
        \dim_add:Nn \l__tblr_h_dim { - \l__tblr_s_tl }
      }
    \__tblr_spec_gput_if_larger:nee { hline }
      { [#1] / @hline-height } { \dim_use:N \l__tblr_h_dim }
  }

%% Get text of a hline segment
%% #1: row number, #2: column number; #3: index number
\cs_new_protected:Npn \__tblr_get_hline_segment_child:nnn #1 #2 #3
  {
    \group_begin:
    \tl_set:Ne \l__tblr_w_tl
      { \__tblr_spec_item:ne { hline } { [#1][#2](#3) / wd } }
    \tl_if_empty:NF \l__tblr_w_tl
      { \dim_set:Nn \lTblrDefaultHruleWidthDim { \l__tblr_w_tl } }
    \tl_set:Ne \l__tblr_d_tl
      { \__tblr_spec_item:ne { hline } { [#1][#2](#3) / @dash } }
    \tl_set:Ne \l__tblr_a_tl { \tl_head:N \l__tblr_d_tl }
    \tl_set:Ne \l__tblr_b_tl { \tl_tail:N \l__tblr_d_tl }
    \tl_if_head_eq_meaning:VNTF \l__tblr_d_tl \q__tblr_dash
      {
        \__tblr_get_hline_dash_style:N \l__tblr_b_tl
        \xleaders \l__tblr_b_tl \hfil
```

```
      }
      {
        \bool_if:NTF \l__tblr_math_mode_bool
          { $ \l__tblr_b_tl $ } { \l__tblr_b_tl }
        \hfil
      }
    \group_end:
  }

%% current cell alignments
\tl_new:N \g__tblr_cell_halign_tl
\tl_new:N \g__tblr_cell_valign_tl
\tl_new:N \g__tblr_cell_middle_tl

\tl_const:Nn \c__tblr_valign_h_tl { h }
\tl_const:Nn \c__tblr_valign_m_tl { m }
\tl_const:Nn \c__tblr_valign_f_tl { f }
\tl_const:Nn \c__tblr_valign_t_tl { t }
\tl_const:Nn \c__tblr_valign_b_tl { b }

\tl_const:Nn \c__tblr_middle_t_tl { t }
\tl_const:Nn \c__tblr_middle_m_tl { m }
\tl_const:Nn \c__tblr_middle_b_tl { b }

%% #1: row number; #2: column number
\cs_new_protected:Npn \__tblr_get_cell_alignments:nn #1 #2
  {
    \group_begin:
    \tl_gset:Ne \g__tblr_cell_halign_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { halign } }
    \tl_set:Ne \l__tblr_v_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { valign } }
    \tl_case:NnF \l__tblr_v_tl
      {
        \c__tblr_valign_t_tl
          {
            \tl_gset:Nn \g__tblr_cell_valign_tl {m}
            \tl_gset:Nn \g__tblr_cell_middle_tl {t}
          }
        \c__tblr_valign_m_tl
          {
            \tl_gset:Nn \g__tblr_cell_valign_tl {m}
            \tl_gset:Nn \g__tblr_cell_middle_tl {m}
          }
        \c__tblr_valign_b_tl
          {
            \tl_gset:Nn \g__tblr_cell_valign_tl {m}
            \tl_gset:Nn \g__tblr_cell_middle_tl {b}
          }
      }
      {
        \tl_gset_eq:NN \g__tblr_cell_valign_tl \l__tblr_v_tl
        \tl_gclear:N \g__tblr_cell_middle_tl
      }
    \group_end:
  }
```

```
%% current cell dimensions
\dim_new:N \g__tblr_cell_wd_dim
\dim_new:N \g__tblr_cell_ht_dim
\dim_new:N \g__tblr_cell_head_dim
\dim_new:N \g__tblr_cell_foot_dim

%% Calculate the width and height for every cell
\cs_new_protected:Npn \__tblr_calculate_cell_sizes:
  {
    %% You can use these two counters in cell text
    \int_zero:N \c@rownum
    \int_zero:N \c@colnum
    \__tblr_save_counters:n { table }
    \int_step_inline:nn { \c@rowcount }
      {
        \int_incr:N \c@rownum
        \int_zero:N \c@colnum
        \__tblr_update_rowsep_registers:
        %% Note that \__tblr_data_item always returns nonempty value,
        %% but \__tblr_prop_item may return empty value.
        %% To make \__tblr_map_data_to_prop: work, we need to add + 0pt.
        \tl_set:Ne \l__tblr_h_tl
          {
            \__tblr_data_item:nen { row } { \int_use:N \c@rownum } { height }
            + 0pt
          }
        %% We didn't initialize row heights with -1pt
        \dim_compare:nNnF { \l__tblr_h_tl } = { 0pt }
          {
            \__tblr_data_gput:nenV { row } { \int_use:N \c@rownum }
              { @row-height } \l__tblr_h_tl
          }
        \int_step_inline:nn { \c@colcount }
          {
            \int_incr:N \c@colnum
            \__tblr_update_colsep_registers:
            \__tblr_measure_cell_update_sizes:nnNNNN
              { \int_use:N \c@rownum }
              { \int_use:N \c@colnum }
              \g__tblr_cell_wd_dim
              \g__tblr_cell_ht_dim
              \g__tblr_cell_head_dim
              \g__tblr_cell_foot_dim
          }
      }
    \__tblr_restore_counters:n { table }
    \int_step_inline:nn { \c@colcount }
      {
        \tl_set:Ne \l__tblr_w_tl
          { \__tblr_data_item:nen { column } {##1} { width } }
        \dim_compare:nNnF { \l__tblr_w_tl } < { 0pt }
          {
            \__tblr_data_gput:nenV { column } {##1} { @col-width } \l__tblr_w_tl
          }
      }
  }

\cs_new_protected:Npn \__tblr_update_rowsep_registers:
```

```
  {
    \dim_set:Nn \abovesep
      { \__tblr_data_item:nen { row } { \int_use:N \c@rownum } { abovesep } }
    \dim_set:Nn \belowsep
      { \__tblr_data_item:nen { row } { \int_use:N \c@rownum } { belowsep } }
  }

\cs_new_protected:Npn \__tblr_update_colsep_registers:
  {
    \dim_set:Nn \leftsep
      { \__tblr_data_item:nen { column } { \int_use:N \c@colnum } { leftsep } }
    \dim_set:Nn \rightsep
      { \__tblr_data_item:nen { column } { \int_use:N \c@colnum } { rightsep } }
  }

%% Measure and update natural dimensions of the row/column/cell
%% #1: row number; #2 column number; #3: width dimension;
%% #4: total height dimension; #5: head dimension; #6: foot dimension
\cs_new_protected:Npn \__tblr_measure_cell_update_sizes:nnNNNN #1 #2 #3 #4 #5 #6
  {
    \__tblr_get_cell_alignments:nn {#1} {#2}
    \hbox_set:Nn \l_tmpa_box { \__tblr_get_cell_text:nn {#1} {#2} }
    \__tblr_update_cell_size:nnNNNN {#1} {#2} #3 #4 #5 #6
    \__tblr_update_row_size:nnNNN {#1} {#2} #4 #5 #6
    \__tblr_update_col_size:nN {#2} #3
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_get_cell_text:nn #1 #2
  {
    \int_compare:nNnTF { \__tblr_data_item:neen { cell } {#1} {#2} { omit } } > {0}
      {
        \dim_gzero:N \g__tblr_cell_wd_dim
        \dim_gzero:N \g__tblr_cell_ht_dim
        \dim_gzero:N \g__tblr_cell_head_dim
        \dim_gzero:N \g__tblr_cell_foot_dim
      }
      { \__tblr_get_cell_text_real:nn { #1 } { #2 } }
  }

\tl_new:N \l__tblr_cell_fg_tl
\tl_new:N \l__tblr_cell_cmd_tl
\tl_new:N \l__tblr_cell_mode_tl
\bool_new:N \l__tblr_cell_math_mode_bool
\tl_const:Nn \c__tblr_cell_math_style_tl { \relax }
\tl_const:Nn \c__tblr_cell_imath_style_tl { \textstyle }
\tl_const:Nn \c__tblr_cell_dmath_style_tl { \displaystyle }
\box_new:N \l__tblr_measured_cell_box

%% Get cell text, #1: row number, #2: column number
%% If the width of the cell is not set, split it with \\ and compute the width
%% Therefore we always get a vbox for any cell
\cs_new_protected:Npn \__tblr_get_cell_text_real:nn #1 #2
  {
    \group_begin:
    \tl_set:Ne \l__tblr_c_tl { \__tblr_spec_item:ne { text } {[#1][#2]} }
    %% when the cell text is guarded by a pair of curly braces,
```

```
    %% we unbrace it and ignore cmd option of the cell, see issue #90.
    \bool_lazy_and:nnTF
      { \tl_if_single_p:N \l__tblr_c_tl }
      { \exp_args:NV \tl_if_head_is_group_p:n \l__tblr_c_tl }
      { \exp_last_unbraced:NNV \tl_set:Nn \l__tblr_c_tl \l__tblr_c_tl }
      {
        \tl_set:Ne \l__tblr_cell_cmd_tl
          { \__tblr_data_item:neen { cell } {#1} {#2} { cmd } }
        \tl_if_empty:NF \l__tblr_cell_cmd_tl
          {
            \tl_set:Ne \l__tblr_c_tl
              { \exp_not:V \l__tblr_cell_cmd_tl { \exp_not:V \l__tblr_c_tl } }
          }
      }
    \tl_set:Ne \l__tblr_cell_mode_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { mode } }
    \tl_if_empty:NT \l__tblr_cell_mode_tl
      {
        \bool_if:NTF \l__tblr_math_mode_bool
          { \tl_set:Nn \l__tblr_cell_mode_tl { math } }
          { \tl_set:Nn \l__tblr_cell_mode_tl { text } }
      }
    \tl_if_eq:NnTF \l__tblr_cell_mode_tl { text }
      { \bool_set_false:N \l__tblr_cell_math_mode_bool }
      {
        \bool_set_true:N \l__tblr_cell_math_mode_bool
        \tl_put_left:Nv \l__tblr_c_tl
          { c__tblr_cell_ \l__tblr_cell_mode_tl _style_tl }
        \tl_put_left:Nn \l__tblr_c_tl { $ }
        \tl_put_right:Nn \l__tblr_c_tl { $ }
      }
    \tl_set:Ne \l__tblr_f_tl { \__tblr_data_item:neen { cell } {#1} {#2} { font } }
    \tl_set:Ne \l__tblr_w_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { width } }
    \dim_compare:nNnT { \l__tblr_w_tl } < { 0pt } % cell width unset
      {
        \int_compare:nNnT
          { \__tblr_data_item:neen { cell } {#1} {#2} { colspan } } < {2}
          {
            \tl_set:Ne \l__tblr_w_tl
              { \__tblr_data_item:nen { column } {#2} { width } }
          }
      }
    \box_clear:N \l__tblr_measured_cell_box
    \dim_compare:nNnT { \l__tblr_w_tl } < { 0pt } % column width unset
      {
        %% To keep \lTblrMeasuringBool correct we need measure=vstore from varwidth.
        %% A temporary private hook for testfiles/supporthook/regression-test.cfg.
        \use:c { __tblr_temp_hook_for_measure_vstore_testing: }
        \tl_if_eq:NnTF \l__tblr_inner_spec_measure_tl { vstore }
          {
            \__tblr_build_vcell_with_vstore:
          }
          {
            \__tblr_save_counters:n { cell }
            \bool_if:NTF \l__tblr_cell_math_mode_bool
              {
                %% Note that font = \boldmath will increase cell width (issue #137)
```

```
                \hbox_set:Nn \l_tmpa_box { \l__tblr_f_tl \l__tblr_c_tl }
                \tl_set:Ne \l__tblr_w_tl { \dim_eval:n { \box_wd:N \l_tmpa_box } }
              }
              {
                \__tblr_get_cell_size_with_box:
              }
            \__tblr_restore_counters:n { cell }
          }
      }
    \tl_put_left:NV \l__tblr_c_tl \l__tblr_f_tl
    \tl_set:Ne \l__tblr_cell_fg_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { foreground } }
    \tl_if_empty:NF \l__tblr_cell_fg_tl
      { \exp_args:NV \color \l__tblr_cell_fg_tl }
    \__tblr_get_vcell_and_sizes:NN \l__tblr_c_tl \l__tblr_w_tl
    \group_end:
  }

\box_new:N \g__tblr_last_box
\box_new:N \l__tblr_temp_box

%% Build cell vbox with varwidth and use it later when hook library is loaded
%% We apply \lastbox trick to get the cell vbox with correct width
\cs_new_protected:Npn  \__tblr_build_vcell_with_vstore:
  {
    \vbox_set:Nn \l__tblr_temp_box % we don't use it since its width is too large
      {
        \begin{varwidth}[t]{\paperwidth}
          \TblrParboxRestore
          \cs:w __tblr_halign_command_ \g__tblr_cell_halign_tl : \cs_end:
          \__tblr_leave_vmode:
          \l__tblr_f_tl
          \l__tblr_c_tl
        \end{varwidth}
        \box_gset_to_last:N \g__tblr_last_box
      }
    \box_set_eq:NN \l__tblr_measured_cell_box \g__tblr_last_box
    \tl_set:Ne \l__tblr_w_tl
      { \dim_eval:n { \box_wd:N \l__tblr_measured_cell_box } }
  }

%% Measure cell width only and build it later when hook library is not loaded
\cs_new_protected:Npn \__tblr_get_cell_size_with_box:
  {
    \tl_if_eq:NnTF \l__tblr_inner_spec_measure_tl { vbox }
      { \__tblr_get_cell_size_with_vbox: }
      { \__tblr_get_cell_size_with_hbox: }
  }

%% Varwidth won't work as expected when \color command occurs in it,
%% and we can not fix this problem with \leavevmode command.
%% See https://tex.stackexchange.com/q/460489.
%% But we need to use \color command for fg option,
%% or users may use it in the middle of the cell text,
%% so we have redefine \color command and disable it before measuring cell.

%% \mathcolor uses '\group_insert_after:N \use_none:n' to gobble \reset@color,
```

```
%% so we need to insert a \scan_stop: for it. See issue #603

\NewDocumentCommand \__tblr_fake_color_command:w { o m }
  { \group_insert_after:N \scan_stop: }

%% In order to correctly measure an enumerate environment,
%% we need to enclose varwidth with NoHyper environment (see issue #196).

\cs_new_protected:Npn \__tblr_get_cell_size_with_vbox:
  {
    \hbox_set:Nn \l_tmpa_box
      {
        \cs_set_eq:NN \color \__tblr_fake_color_command:w
        \begin{tblrNoHyper}
        \begin{varwidth}{\paperwidth}
           \l__tblr_f_tl
           \l__tblr_c_tl
        \end{varwidth}
        \end{tblrNoHyper}
      }
    \tl_set:Ne \l__tblr_w_tl { \dim_eval:n { \box_wd:N \l_tmpa_box } }
  }

\cs_new_protected:Npn \__tblr_get_cell_size_with_hbox:
  {
    \tl_set_eq:NN \l_tmpb_tl \l__tblr_c_tl
    \__tblr_seq_set_split_keep_envs:NnV \l_tmpa_seq { \\ } \l_tmpb_tl
    \tl_set:Nn \l__tblr_w_tl { 0pt }
    \seq_map_variable:NNn \l_tmpa_seq \l_tmpa_tl
      {
        \hbox_set:Nn \l_tmpa_box
          {
            \l__tblr_f_tl
            \l_tmpa_tl
          }
        \tl_set:Ne \l__tblr_w_tl
          { \dim_max:nn { \l__tblr_w_tl } { \box_wd:N \l_tmpa_box } }
      }
  }

%% #1: cell text; #2: box width
\cs_new_protected:Npn \__tblr_get_vcell_and_sizes:NN #1 #2
  {
    \group_begin:
    \box_if_empty:NTF \l__tblr_measured_cell_box
      { \vbox_set_top:Nn \l_tmpa_box { \__tblr_make_vcell_text:NN #1 #2 } }
      { \box_set_eq:NN \l_tmpa_box \l__tblr_measured_cell_box }
    \vbox_set:Nn \l_tmpb_box { \vbox_unpack:N \l_tmpa_box }
    \__tblr_process_stretch:
    \dim_gset:Nn \g__tblr_cell_wd_dim { \box_wd:N \l_tmpb_box }
    \dim_gset:Nn \g__tblr_cell_ht_dim
      { \box_ht:N \l_tmpb_box + \box_dp:N \l_tmpb_box }
    \dim_gset:Nn \g__tblr_cell_head_dim { \box_ht:N \l_tmpa_box }
    \dim_gset:Nn \g__tblr_cell_foot_dim { \box_dp:N \l_tmpb_box }
    \tl_case:Nn \g__tblr_cell_valign_tl
      {
        \c__tblr_valign_h_tl
```

```
          { \box_use:N \l_tmpa_box }
        \c__tblr_valign_m_tl
          {
            \tl_case:Nn \g__tblr_cell_middle_tl
              {
                \c__tblr_middle_t_tl
                  { \box_use:N \l_tmpa_box }
                \c__tblr_middle_m_tl
                  {
                    \tl_set:Ne \l__tblr_b_tl
                      {
                        \dim_eval:n
                          {
                            ( \g__tblr_cell_ht_dim - \g__tblr_cell_head_dim
                                                   - \g__tblr_cell_foot_dim ) / 2
                          }
                      }
                    \box_set_ht:Nn \l_tmpb_box
                      { \g__tblr_cell_head_dim + \l__tblr_b_tl }
                    \box_set_dp:Nn \l_tmpb_box
                      { \g__tblr_cell_foot_dim + \l__tblr_b_tl }
                    \box_use:N \l_tmpb_box
                  }
                \c__tblr_middle_b_tl
                  { \box_use:N \l_tmpb_box }
              }
          }
        \c__tblr_valign_f_tl
          { \box_use:N \l_tmpb_box }
      }
    \group_end:
  }

%% #1: cell text; #2: box width
%% All halign commands are defined at the beginning of the file
\cs_new_protected:Npn \__tblr_make_vcell_text:NN #1 #2
  {
    \dim_set:Nn \tex_hsize:D { #2 }
    \TblrParboxRestore
    \cs:w __tblr_halign_command_ \g__tblr_cell_halign_tl : \cs_end:
    \__tblr_leave_vmode:
    #1
  }

%% #1: total height dimension; #2: head dimension; #3: foot dimension;
%% #4: tl for resulting upper size; #5: tl for resulting lower size

\tl_new:N \l__tblr_middle_body_tl

\cs_new_protected:Npn \__tblr_get_middle_cell_upper_lower:NNNNN #1 #2 #3 #4 #5
  {
    \tl_case:Nn \g__tblr_cell_middle_tl
      {
        \c__tblr_middle_t_tl
          {
            \tl_set:Ne #4 { \dim_use:N #2 }
            \tl_set:Ne #5 { \dim_eval:n { #1 - #2 } }
```

```
          }
        \c__tblr_middle_m_tl
          {
            \tl_set:Ne \l__tblr_middle_body_tl { \dim_eval:n { #1 - #2 - #3 } }
            \tl_set:Ne #4 { \dim_eval:n { #2 + \l__tblr_middle_body_tl / 2 } }
            \tl_set:Ne #5 { \dim_eval:n { #3 + \l__tblr_middle_body_tl / 2 } }
          }
        \c__tblr_middle_b_tl
          {
            \tl_set:Ne #4 { \dim_eval:n { #1 - #3 } }
            \tl_set:Ne #5 { \dim_use:N #3 }
          }
      }
  }

%% Update natural dimensions of the cell
%% #1: row number; #2 column number; #3: width dimension;
%% #4: total height dimension; #5: head dimension; #6: foot dimension
\cs_new_protected:Npn \__tblr_update_cell_size:nnNNNN #1 #2 #3 #4 #5 #6
  {
    \group_begin:
    \tl_set:Ne \l__tblr_c_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { colspan } }
    \int_compare:nNnT { \l__tblr_c_tl } > {1}
      {
        \__tblr_data_gput:neene { cell } {#1} {#2} { @cell-width } {\dim_use:N #3}
        \dim_gzero:N #3 % don't affect column width
      }
    \tl_set:Ne \l__tblr_r_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { rowspan } }
    \int_compare:nNnT { \l__tblr_r_tl } > {1}
      {
        \tl_case:Nn \g__tblr_cell_valign_tl
          {
            \c__tblr_valign_h_tl
              {
                \tl_set:Ne \l__tblr_u_tl { \dim_use:N #5 }
                \tl_set:Ne \l__tblr_v_tl { \dim_eval:n { #4 - #5 } }
                %% Update the head size of the first span row here
                \__tblr_data_gput_if_larger:nene
                  { row } {#1} { @row-head } { \dim_use:N #5 }
              }
            \c__tblr_valign_f_tl
              {
                \tl_set:Ne \l__tblr_u_tl { \dim_eval:n { #4 - #6 } }
                \tl_set:Ne \l__tblr_v_tl { \dim_use:N #6 }
                %% Update the foot size of the last span row here
                \__tblr_data_gput_if_larger:nene
                  { row }
                  { \int_eval:n { #1 + \l__tblr_r_tl - 1 } }
                  { @row-foot }
                  { \dim_use:N #6 }
              }
            \c__tblr_valign_m_tl
              {
                \__tblr_get_middle_cell_upper_lower:NNNNN
                  #4 #5 #6 \l__tblr_u_tl \l__tblr_v_tl
              }
```

```
          }
        \__tblr_data_gput:neenV { cell } {#1} {#2} { @cell-height } \l__tblr_u_tl
        \__tblr_data_gput:neenV { cell } {#1} {#2} { @cell-depth } \l__tblr_v_tl
        %% Don't affect row sizes
        \dim_gzero:N #4
        \dim_gzero:N #5
        \dim_gzero:N #6
      }
    \group_end:
  }

%% Update size of the row. #1: row number; #2: column number;
%% #3: total height dimension; #4: head dimension; #5: foot dimension
\cs_new_protected:Npn \__tblr_update_row_size:nnNNN #1 #2 #3 #4 #5
  {
    \group_begin:
    %% Note that \l__tblr_h_tl may be empty
    \tl_set:Ne \l__tblr_h_tl
      { \__tblr_data_item:nen { row } {#1} { @row-height } }
    \tl_if_eq:NNTF \g__tblr_cell_valign_tl \c__tblr_valign_m_tl
      {
        \tl_set:Ne \l__tblr_a_tl
          { \__tblr_data_item:nen { row } {#1} { @row-upper } }
        \tl_set:Ne \l__tblr_b_tl
          { \__tblr_data_item:nen { row } {#1} { @row-lower } }
        \__tblr_get_middle_cell_upper_lower:NNNNN
          #3 #4 #5 \l__tblr_u_tl \l__tblr_v_tl
        \dim_compare:nNnT { \l__tblr_u_tl } > { \l__tblr_a_tl }
          {
            \tl_set_eq:NN \l__tblr_a_tl \l__tblr_u_tl
            \__tblr_data_gput:nenV { row } {#1} { @row-upper } \l__tblr_a_tl
          }
        \dim_compare:nNnT { \l__tblr_v_tl } > { \l__tblr_b_tl }
          {
            \tl_set_eq:NN \l__tblr_b_tl \l__tblr_v_tl
            \__tblr_data_gput:nenV { row } {#1} { @row-lower } \l__tblr_b_tl
          }
        \dim_compare:nNnT
          { \l__tblr_a_tl + \l__tblr_b_tl } > { \l__tblr_h_tl + 0pt }
          {
            \__tblr_data_gput:nene { row } {#1} { @row-height }
              { \dim_eval:n { \l__tblr_a_tl + \l__tblr_b_tl } }
          }
      }
      {
        \tl_set:Ne \l__tblr_e_tl
          { \__tblr_data_item:nen { row } {#1} { @row-head } }
        \tl_set:Ne \l__tblr_f_tl
          { \__tblr_data_item:nen { row } {#1} { @row-foot } }
        \dim_compare:nNnT {#4} > {\l__tblr_e_tl}
          {
            \__tblr_data_gput:nene { row } {#1} { @row-head } { \dim_use:N #4 }
          }
        \dim_compare:nNnT {#5} > {\l__tblr_f_tl}
          {
            \__tblr_data_gput:nene { row } {#1} { @row-foot } { \dim_use:N #5 }
          }
        \tl_set:Ne \l__tblr_x_tl { \dim_max:nn {#4} { \l__tblr_e_tl } }
```

```
        \tl_set:Ne \l__tblr_y_tl { \dim_max:nn {#5} { \l__tblr_f_tl } }
        \dim_compare:nNnT
          { #3 - #4 - #5 } > { \l__tblr_h_tl - \l__tblr_x_tl - \l__tblr_y_tl }
          {
            \__tblr_data_gput:nene { row } {#1} { @row-height }
              {
                \dim_eval:n
                  {
                    \l__tblr_x_tl
                    + \dim_use:N #3 - \dim_use:N #4 - \dim_use:N #5
                    + \l__tblr_y_tl
                  }
              }
          }
      }
    \group_end:
  }

%% Update size of the column. #1: column number; #2: width dimension

\cs_new_protected:Npn \__tblr_update_col_size:nN #1 #2
  {
    \tl_set:Ne \l_tmpb_tl
      { \__tblr_data_item:nen { column } {#1} { @col-width } }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l_tmpb_tl }
      { \dim_compare_p:nNn { \dim_use:N #2 } > { \l_tmpb_tl } }
      {
        \__tblr_data_gput:nene { column } {#1} { @col-width } { \dim_use:N #2 }
      }
  }
```

9.26 Calculate and adjust extendable columns

```
%% Compute column widths when there are some extendable columns

\dim_new:N \l__tblr_column_target_dim
\prop_new:N \l__tblr_column_coefficient_prop
\prop_new:N \l__tblr_column_natural_width_prop
\prop_new:N \l__tblr_column_computed_width_prop

\msg_new:nnn { tabularray } { table-width-too-small }
  { Table ~ width ~ is ~ too ~ small, ~ need ~ #1 ~ more! }

\cs_new_protected:Npn \__tblr_compute_extendable_column_width:
  {
    \__tblr_collect_extendable_column_width:
    \dim_compare:nNnTF { \l__tblr_column_target_dim } < { 0pt }
      {
        \tl_if_empty:eF { \__tblr_prop_item:nn { inner } { width } }
          {
            \msg_warning:nne { tabularray } { table-width-too-small }
              { \dim_abs:n { \l__tblr_column_target_dim } }
          }
      }
```

```
      {
        \prop_if_empty:NF \l__tblr_column_coefficient_prop
          { \__tblr_adjust_extendable_column_width: }
      }
  }

\cs_new_protected:Npn \__tblr_collect_extendable_column_width:
  {
    \tl_set:Ne \l_tmpa_tl { \__tblr_prop_item:nn { inner } { width } }
    \tl_if_empty:NTF \l_tmpa_tl
      { \dim_set_eq:NN \l__tblr_column_target_dim \linewidth }
      { \dim_set:Nn \l__tblr_column_target_dim { \l_tmpa_tl } }
    \prop_clear:N \l__tblr_column_coefficient_prop
    \prop_clear:N \l__tblr_column_natural_width_prop
    \prop_clear:N \l__tblr_column_computed_width_prop
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \tl_set:Ne \l__tblr_a_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { width } }
        \tl_set:Ne \l__tblr_b_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { coefficient } }
        \tl_set:Ne \l__tblr_c_tl
          { \__tblr_data_item:nen  { column } { \l__tblr_j_tl } { @col-width } }
        \dim_compare:nNnTF { \l__tblr_a_tl } < { 0pt } % column width unset
          {
            \dim_compare:nNnTF { \l__tblr_b_tl pt } = { 0pt }
              { \dim_sub:Nn \l__tblr_column_target_dim { \l__tblr_c_tl } }
              {
                \prop_put:Nee \l__tblr_column_coefficient_prop
                  { \l__tblr_j_tl } { \l__tblr_b_tl }
                \prop_put:Nen \l__tblr_column_computed_width_prop
                  { \l__tblr_j_tl } { 0pt }
                \dim_compare:nNnF { \l__tblr_b_tl pt } > { 0pt }
                  {
                    \prop_put:Nee \l__tblr_column_natural_width_prop
                      { \l__tblr_j_tl } { \l__tblr_c_tl }
                  }
              }
          }
          { \dim_sub:Nn \l__tblr_column_target_dim { \l__tblr_a_tl } }
        \tl_set:Ne \l__tblr_a_tl
          { \__tblr_spec_item:ne { vline } { [\l__tblr_j_tl] / @vline-width } }
        \tl_set:Ne \l__tblr_b_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { leftsep } }
        \tl_set:Ne \l__tblr_c_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { rightsep } }
        \dim_set:Nn \l__tblr_column_target_dim
          {
            \l__tblr_column_target_dim
            - \l__tblr_a_tl - \l__tblr_b_tl - \l__tblr_c_tl
          }
      }
    \tl_set:Ne \l__tblr_a_tl
      {
        \__tblr_spec_item:ne { vline }
          { [\int_eval:n {\c@colcount + 1}] / @vline-width }
      }
    \tl_if_empty:NF \l__tblr_a_tl
```

```
      { \dim_sub:Nn \l__tblr_column_target_dim { \l__tblr_a_tl } }
    \LogTblrTracing { target }
  }

%% Users may modify \hfuzz, so we use our hfuzz dim variable (see issue #445)
\dim_new:N \l__tblr_hfuzz_dim
\dim_set:Nn \l__tblr_hfuzz_dim { 0.1pt }

%% If all columns have negative coefficients and small natural widths,
%% \l__tblr_column_coefficient_prop will be empty after one or more rounds.
%% We reset @row-height, etc for \linewidth graphics in X columns (issue #80)
\cs_new_protected:Npn \__tblr_adjust_extendable_column_width:
  {
    \bool_while_do:nn
      { \dim_compare_p:nNn { \l__tblr_column_target_dim } > { \l__tblr_hfuzz_dim } }
      {
        \prop_if_empty:NTF \l__tblr_column_coefficient_prop
          { \__tblr_adjust_extendable_column_width_negative: }
          { \__tblr_adjust_extendable_column_width_once: }
      }
    \prop_map_inline:Nn \l__tblr_column_computed_width_prop
      {
        \__tblr_data_gput:nnne { column } {##1} { width } {##2}
        \__tblr_data_gput:nnnn { column } {##1} { @col-width } { 0pt }
      }
    \int_step_inline:nn { \c@rowcount }
      {
        \__tblr_data_gput:nnnn { row } {##1} { @row-height } { 0pt }
        \__tblr_data_gput:nnnn { row } {##1} { @row-head } { 0pt }
        \__tblr_data_gput:nnnn { row } {##1} { @row-foot } { 0pt }
        \__tblr_data_gput:nnnn { row } {##1} { @row-upper } { 0pt }
        \__tblr_data_gput:nnnn { row } {##1} { @row-lower } { 0pt }
      }
    \__tblr_calculate_cell_sizes:
  }

%% We use dimen register, since the coefficient may be a decimal number
\cs_new_protected:Npn \__tblr_adjust_extendable_column_width_once:
  {
    \dim_zero:N \l_tmpa_dim
    \prop_map_inline:Nn \l__tblr_column_coefficient_prop
      {
        \dim_add:Nn \l_tmpa_dim { \dim_abs:n { ##2 pt } }
      }
    \tl_set:Ne \l__tblr_w_tl
      { \dim_ratio:nn { \l__tblr_column_target_dim } { \l_tmpa_dim } }
    \dim_zero:N \l__tblr_column_target_dim
    \prop_map_inline:Nn \l__tblr_column_coefficient_prop
      {
        \tl_set:Ne \l__tblr_a_tl
          { \dim_eval:n { \dim_abs:n { ##2 pt } * \l__tblr_w_tl } }
        \dim_compare:nNnTF { ##2 pt } > { 0pt }
          {
            \__tblr_add_dimen_value:Nnn
              \l__tblr_column_computed_width_prop { ##1 } { \l__tblr_a_tl }
          }
          {
```

```
            \tl_set:Ne \l__tblr_b_tl
              { \prop_item:Nn \l__tblr_column_natural_width_prop { ##1 } }
            \tl_set:Ne \l__tblr_c_tl
              { \prop_item:Nn \l__tblr_column_computed_width_prop { ##1 } }
            \dim_compare:nNnTF { \l__tblr_a_tl + \l__tblr_c_tl } > { \l__tblr_b_tl }
              {
                \prop_put:Nne \l__tblr_column_computed_width_prop
                  { ##1 } { \l__tblr_b_tl }
                \dim_add:Nn \l__tblr_column_target_dim
                  { \l__tblr_a_tl + \l__tblr_c_tl - \l__tblr_b_tl }
                \prop_remove:Nn \l__tblr_column_coefficient_prop { ##1 }
              }
              {
                \__tblr_add_dimen_value:Nnn
                  \l__tblr_column_computed_width_prop { ##1 } { \l__tblr_a_tl }
              }
          }
      }
    \LogTblrTracing { target }
  }

\cs_new_protected:Npn \__tblr_adjust_extendable_column_width_negative:
  {
    \dim_zero:N \l_tmpa_dim
    \prop_map_inline:Nn \l__tblr_column_natural_width_prop
      { \dim_add:Nn \l_tmpa_dim { ##2 } }
    \tl_set:Ne \l_tmpa_tl
      { \dim_ratio:nn { \l__tblr_column_target_dim } { \l_tmpa_dim } }
    \dim_zero:N \l__tblr_column_target_dim
    \prop_map_inline:Nn \l__tblr_column_natural_width_prop
      {
        \tl_set:Ne \l_tmpb_tl { \dim_eval:n { ##2 * \l_tmpa_tl } }
        \__tblr_add_dimen_value:Nnn
          \l__tblr_column_computed_width_prop { ##1 } { \l_tmpb_tl }
      }
    \LogTblrTracing { target }
  }
```

9.27 Calculate and adjust multispan cells

```
%% Compute and adjust widths when there are some span cells.
%% By default, we will compute column widths from span widths;
%% but if we set table option "hspan = minimal",
%% we will compute span widths from column widths.

\cs_new_protected:Npn \__tblr_adjust_sizes_for_span_cells:
  {
    \__tblr_prop_if_in:nnT { inner } { colspan }
      {
        \__tblr_collect_column_widths_skips:
        \str_if_eq:enTF
          { \__tblr_prop_item:ne { inner } { hspan } } { minimal }
          {
            \__tblr_set_span_widths_from_column_widths:
          }
          {
```

```
            \__tblr_collect_span_widths:
            \__tblr_set_column_widths_from_span_widths:
          }
        \LogTblrTracing { column }
        \__tblr_calculate_cell_sizes:
      }
    \__tblr_prop_if_in:nnT { inner } { rowspan }
      {
        \__tblr_collect_row_heights_skips:
        \__tblr_collect_span_heights:
        \__tblr_set_row_heights_from_span_heights:
        \LogTblrTracing { row }
      }
  }

\prop_new:N \l__tblr_col_item_skip_size_prop
\prop_new:N \l__tblr_col_span_size_prop
\prop_new:N \l__tblr_row_item_skip_size_prop
\prop_new:N \l__tblr_row_span_size_prop

\cs_new_protected:Npn \__tblr_collect_column_widths_skips:
  {
    \prop_clear:N \l__tblr_col_item_skip_size_prop
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \int_compare:nNnTF { \l__tblr_j_tl } > { 1 }
          {
            \prop_put:Nee \l__tblr_col_item_skip_size_prop { skip[\l__tblr_j_tl] }
              {
                \dim_eval:n
                  {
                    \__tblr_data_item:nen { column }
                      { \int_eval:n { \l__tblr_j_tl - 1 } } { rightsep }
                    +
                    \__tblr_spec_item:ne { vline }
                      { [\l__tblr_j_tl] / @vline-width }
                    +
                    \__tblr_data_item:nen { column } { \l__tblr_j_tl } { leftsep }
                  }
              }
          }
          {
            \prop_put:Nen \l__tblr_col_item_skip_size_prop { skip[\l__tblr_j_tl] }
              { 0pt }
          }
        \prop_put:Nee \l__tblr_col_item_skip_size_prop { item[\l__tblr_j_tl] }
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { @col-width } }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      { \prop_log:N \l__tblr_col_item_skip_size_prop }
  }

\cs_new_protected:Npn \__tblr_collect_row_heights_skips:
  {
    \prop_clear:N \l__tblr_row_item_skip_size_prop
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
```

```
        \int_compare:nNnTF { \l__tblr_i_tl } > { 1 }
          {
            \prop_put:Nee \l__tblr_row_item_skip_size_prop { skip[\l__tblr_i_tl] }
              {
                \dim_eval:n
                  {
                    \__tblr_data_item:nen { row }
                      { \int_eval:n {\l__tblr_i_tl - 1} } { belowsep }
                    +
                    \__tblr_spec_item:ne { hline }
                      { [\l__tblr_i_tl] / @hline-height }
                    +
                    \__tblr_data_item:nen { row } { \l__tblr_i_tl } { abovesep }
                  }
              }
          }
          {
            \prop_put:Nen \l__tblr_row_item_skip_size_prop { skip[\l__tblr_i_tl] }
              { 0pt }
          }
        \__tblr_collect_one_row_height:NN \l__tblr_i_tl \l__tblr_h_tl
        \prop_put:Nee \l__tblr_row_item_skip_size_prop
          { item[\l__tblr_i_tl] } { \l__tblr_h_tl }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      { \prop_log:N \l__tblr_row_item_skip_size_prop }
  }

%% #1: row number; #2: tl with result
\cs_new_protected:Npn \__tblr_collect_one_row_height:NN #1 #2
  {
    \tl_set:Ne #2 { \__tblr_data_item:nen { row } {#1} { @row-height } }
  }

\cs_new_protected:Npn \__tblr_collect_span_widths:
  {
    \prop_clear:N \l__tblr_col_span_size_prop
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
          {
            \tl_set:Ne \l__tblr_a_tl
              {
                \__tblr_data_item:neen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { colspan }
              }
            \int_compare:nNnT { \l__tblr_a_tl } > {1}
              {
                \__tblr_put_if_larger:Nee \l__tblr_col_span_size_prop
                  {
                    ( \l__tblr_j_tl -
                      \int_eval:n {\l__tblr_j_tl + \l__tblr_a_tl - 1} )
                  }
                  {
                    \__tblr_data_item:neen { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-width }
                  }
              }
```

```
          }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      { \prop_log:N \l__tblr_col_span_size_prop }
  }

\prop_new:N \l__tblr_row_span_to_row_prop

\cs_new_protected:Npn \__tblr_collect_span_heights:
  {
    \prop_clear:N \l__tblr_row_span_to_row_prop
    \prop_clear:N \l__tblr_row_span_size_prop
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
          {
            \tl_set:Ne \l__tblr_a_tl
              {
                \__tblr_data_item:neen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { rowspan }
              }
            \int_compare:nNnT { \l__tblr_a_tl } > {1}
              {
                \tl_set:Ne \l__tblr_v_tl
                  {
                    \__tblr_data_item:neen { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { valign }
                  }
                \tl_if_eq:NnT \l__tblr_v_tl { h }
                  {
                    \tl_set:Ne \l__tblr_h_tl
                      {
                        \__tblr_data_item:nen { row }
                          { \l__tblr_i_tl } { @row-head }
                      }
                    \__tblr_data_gput:neenV { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-height }
                      \l__tblr_h_tl
                  }
                \tl_if_eq:NnT \l__tblr_v_tl { f }
                  {
                    \tl_set:Ne \l__tblr_d_tl
                      {
                        \__tblr_data_item:nen
                          { row }
                          { \int_eval:n { \l__tblr_i_tl + \l__tblr_a_tl - 1 } }
                          { @row-foot }
                      }
                    \__tblr_data_gput:neenV { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-depth }
                      \l__tblr_d_tl
                  }
                \__tblr_put_if_larger:Nee \l__tblr_row_span_size_prop
                  {
                    ( \l__tblr_i_tl -
                      \int_eval:n {\l__tblr_i_tl + \l__tblr_a_tl - 1} )
                  }
                  {
```

```
                    \dim_eval:n
                      {
                        \__tblr_data_item:neen { cell }
                          { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-height }
                        +
                        \__tblr_data_item:neen { cell }
                          { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-depth }
                      }
                  }
                \prop_put:Nee \l__tblr_row_span_to_row_prop
                  { [\l__tblr_i_tl][\l__tblr_j_tl] }
                  { \int_eval:n {\l__tblr_i_tl + \l__tblr_a_tl - 1} }
              }
          }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      {
        \prop_log:N \l__tblr_row_span_to_row_prop
        \prop_log:N \l__tblr_row_span_size_prop
      }
  }

%% Compute and set column widths from span widths
\cs_new_protected:Npn \__tblr_set_column_widths_from_span_widths:
  {
    \str_if_eq:enTF
      { \__tblr_prop_item:ne { inner } { hspan } }
      { even }
      {
        \__tblr_distribute_span_sizes_even:eNN
          { \int_use:N \c@colcount }
          \l__tblr_col_item_skip_size_prop
          \l__tblr_col_span_size_prop
      }
      {
        \__tblr_distribute_span_sizes_default:eNN
          { \int_use:N \c@colcount }
          \l__tblr_col_item_skip_size_prop
          \l__tblr_col_span_size_prop
      }
    \__tblr_set_all_column_widths:
  }

%% Compute and set row heights from span heights
\cs_new_protected:Npn \__tblr_set_row_heights_from_span_heights:
  {
    \str_if_eq:enTF
      { \__tblr_prop_item:ne { inner } { vspan } }
      { even }
      {
        \__tblr_distribute_span_sizes_even:nNN
          { \int_use:N \c@rowcount }
          \l__tblr_row_item_skip_size_prop
          \l__tblr_row_span_size_prop
      }
      {
        \__tblr_distribute_span_sizes_default:eNN
          { \int_use:N \c@rowcount }
```

```
          \l__tblr_row_item_skip_size_prop
          \l__tblr_row_span_size_prop
      }
    \__tblr_set_all_row_heights:
  }

%% See page 245 in Chapter 22 of TeXbook
%% #1: total number of items
%% #2: prop list with item sizes and skip sizes; #3: prop list with span sizes
\cs_new_protected:Npn \__tblr_distribute_span_sizes_default:nNN #1 #2 #3
  {
    \int_step_variable:nNn { #1 } \l__tblr_j_tl
      {
        \dim_set:Nn \l__tblr_w_dim
          {
            \prop_item:Ne #2 { item[\l__tblr_j_tl] }
          }
        \int_step_variable:nNn { \l__tblr_j_tl - 1 } \l__tblr_i_tl
          {
            \tl_set:Ne \l__tblr_a_tl
              { \prop_item:Ne #3 { (\l__tblr_i_tl-\l__tblr_j_tl) } }
            \tl_if_empty:NF \l__tblr_a_tl
              {
                \int_step_variable:nnNn
                  { \l__tblr_i_tl } { \l__tblr_j_tl - 1 } \l__tblr_k_tl
                  {
                    \__tblr_do_if_tracing:nn { cellspan }
                      {
                        \tl_log:e
                          { \l__tblr_j_tl : \l__tblr_i_tl -> \l__tblr_k_tl }
                      }
                    \tl_set:Ne \l_tmpa_tl
                      {
                        \prop_item:Ne #2 { itemskip[\l__tblr_k_tl] }
                      }
                    \tl_set:Ne \l__tblr_a_tl
                      { \dim_eval:n { \l__tblr_a_tl - \l_tmpa_tl } }
                  }
                \dim_compare:nNnT { \l__tblr_a_tl } > { \l__tblr_w_dim }
                  {
                    \dim_set:Nn \l__tblr_w_dim { \l__tblr_a_tl }
                  }
              }
          }
        \prop_put:Nee #2
          { item[\l__tblr_j_tl] } { \dim_use:N \l__tblr_w_dim }
        \int_compare:nNnT { \l__tblr_j_tl } < { #1 }
          {
            \tl_set:Ne \l_tmpb_tl
              {
                \prop_item:Ne #2
                  { skip[\int_eval:n { \l__tblr_j_tl + 1} ] }
              }
            \dim_add:Nn \l__tblr_w_dim { \l_tmpb_tl }
            \prop_put:Nee #2
              { itemskip[\l__tblr_j_tl] } { \dim_use:N \l__tblr_w_dim }
          }
      }
```

```
    \__tblr_do_if_tracing:nn { cellspan } { \prop_log:N #2 }
  }
\cs_generate_variant:Nn \__tblr_distribute_span_sizes_default:nNN { e }

%% #1: total number of items
%% #2: prop list with item sizes and skip sizes; #3: prop list with span sizes
\cs_new_protected:Npn \__tblr_distribute_span_sizes_even:nNN #1 #2 #3
  {
    \prop_clear:N \l_tmpa_prop
    \prop_map_inline:Nn #3
      {
        \__tblr_get_span_from_to:w ##1
        \dim_set:Nn \l_tmpa_dim {##2}
        \dim_sub:Nn \l_tmpa_dim { \prop_item:Ne #2 { item[\l__tblr_a_tl] } }
        \int_step_inline:nnn { \l__tblr_a_tl + 1 } { \l__tblr_b_tl }
          {
            \dim_sub:Nn \l_tmpa_dim
              {
                \prop_item:Ne #2 { skip[####1] } + \prop_item:Nn #2 { item[####1] }
              }
          }
        \__tblr_do_if_tracing:nn { cellspan }
          {
            \tl_log:e { \l__tblr_a_tl -> \l__tblr_b_tl : ~ \dim_use:N \l_tmpa_dim }
          }
        \dim_compare:nNnT {\l_tmpa_dim} > {0pt}
          {
            \tl_set:Ne \l_tmpa_tl
              { \dim_eval:n { \l_tmpa_dim / (\l__tblr_b_tl - \l__tblr_a_tl + 1) } }
            \int_step_inline:nnn { \l__tblr_a_tl } { \l__tblr_b_tl }
              {
                \__tblr_put_if_larger:NnV \l_tmpa_prop {####1} \l_tmpa_tl
              }
          }
      }
    \__tblr_do_if_tracing:nn { cellspan } { \prop_log:N \l_tmpa_prop }
    \prop_map_inline:Nn \l_tmpa_prop
      {
        \__tblr_add_dimen_value:Nnn #2 {item[##1]} {##2}
      }
    \__tblr_do_if_tracing:nn { cellspan } { \prop_log:N #2 }
  }
\cs_generate_variant:Nn \__tblr_distribute_span_sizes_even:nNN { e }

\cs_new_protected:Npn \__tblr_get_span_from_to:w (#1-#2)
  {
    \tl_set:Nn \l__tblr_a_tl {#1}
    \tl_set:Nn \l__tblr_b_tl {#2}
  }

\cs_new_protected:Npn \__tblr_set_all_column_widths:
  {
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \__tblr_data_gput:nene { column }
          { \l__tblr_j_tl } { width }
          { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[\l__tblr_j_tl] } }
```

```
      }
  }

\cs_new_protected:Npn \__tblr_set_all_row_heights:
  {
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \tl_set:Ne \l__tblr_h_tl
          {
            \__tblr_data_item:nen { row } { \l__tblr_i_tl } { @row-head }
          }
        \tl_set:Ne \l__tblr_d_tl
          {
            \__tblr_data_item:nen { row } { \l__tblr_i_tl } { @row-foot }
          }
        \tl_set:Ne \l__tblr_a_tl
          {
            \prop_item:Ne \l__tblr_row_item_skip_size_prop { item[\l__tblr_i_tl] }
          }
        \__tblr_collect_one_row_height:NN \l__tblr_i_tl \l__tblr_t_tl
        \__tblr_data_gput:nene { row }
          { \l__tblr_i_tl } { @row-height } { \l__tblr_a_tl }
      }
  }

%% Compute and set span widths from column widths
\cs_new_protected:Npn \__tblr_set_span_widths_from_column_widths:
  {
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
          {
            \tl_set:Ne \l__tblr_a_tl
              {
                \__tblr_data_item:neen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { colspan }
              }
            \int_compare:nNnT { \l__tblr_a_tl } > {1}
              {
                \__tblr_calc_span_widths:eeN
                  { \l__tblr_j_tl }
                  { \int_eval:n { \l__tblr_j_tl + \l__tblr_a_tl - 1 } }
                  \l__tblr_w_dim
                \__tblr_data_gput:neene { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { width }
                  { \dim_use:N \l__tblr_w_dim }
              }
          }
      }
  }

%% Cell is spanned from col #1 to col #2, #3 is the return dim
\cs_new_protected:Npn \__tblr_calc_span_widths:nnN #1 #2 #3
  {
    \dim_set:Nn #3 { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[#1] } }
    \int_step_inline:nnn { #1 + 1 } { #2 }
      {
```

```
        \tl_set:Ne \l_tmpa_tl
          { \prop_item:Ne \l__tblr_col_item_skip_size_prop { skip[##1] } }
        \tl_set:Ne \l_tmpb_tl
          { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[##1] } }
        \dim_add:Nn #3 { \dim_eval:n { \l_tmpa_tl + \l_tmpb_tl } }
      }
  }
\cs_generate_variant:Nn \__tblr_calc_span_widths:nnnN { ee }
```

9.28 Header and footer styles

```
\prop_new:N \l__tblr_element_styles_prop

\cs_new_protected:Npn \__tblr_style_put:nn #1 #2
  {
    \prop_put:Nnn \l__tblr_element_styles_prop {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_style_put:nn { nV, ne, en, eV }

\cs_new:Npn \__tblr_style_item:n #1
  {
    \prop_item:Nn \l__tblr_element_styles_prop {#1}
  }

\cs_new_protected:Npn \__tblr_style_log:
  {
    \prop_log:N \l__tblr_element_styles_prop
  }

\tl_new:N \l__tblr_element_name_tl
\tl_new:N \l__tblr_element_styles_tl

%% #1: list of element names; #2: element styles
\NewDocumentCommand \SetTblrStyle { m +m }
  {
    \tl_set:Nn \l__tblr_element_styles_tl {#2}
    \__tblr_keys_set:nn { template/element } {#1}
    \ignorespaces
  }

\__tblr_keys_define:nn { template/element }
  {
    head    .meta:n = { firsthead, middlehead, lasthead },
    foot    .meta:n = { firstfoot, middlefoot, lastfoot },
    unknown .code:n = \__tblr_set_element_styles:V \l_keys_key_str,
  }

\cs_new_protected:Npn \__tblr_set_element_styles:n #1
  {
    \tl_set:Nn \l__tblr_element_name_tl {#1}
    \__tblr_keys_set:nV { template/style } \l__tblr_element_styles_tl
  }
\cs_generate_variant:Nn \__tblr_set_element_styles:n { V }

\__tblr_keys_define:nn { template/style }
```

```
  {
    halign   .code:n = \__tblr_element_gput_style:nn { halign } {#1},
    l        .meta:n = { halign = l },
    c        .meta:n = { halign = c },
    r        .meta:n = { halign = r },
    j        .meta:n = { halign = j },
    fg       .code:n = \__tblr_element_gput_style:nn { fg } {#1},
    font     .code:n = \__tblr_element_gput_style:nn { font } {#1},
    hang     .code:n = \__tblr_element_gput_style:nn { hang } {#1},
    indent   .code:n = \__tblr_element_gput_style:nn { indent } {#1},
    unknown .code:n = \__tblr_element_unknown_key:Vn \l_keys_key_str {#1},
  }

\cs_new_protected:Npn \__tblr_element_gput_style:nn #1 #2
  {
    \__tblr_style_put:en { \l__tblr_element_name_tl / #1 } {#2}
  }

\cs_new_protected:Npn \__tblr_element_unknown_key:nn #1 #2
  {
    \__tblr_if_color_value:nTF {#1}
      { \__tblr_style_put:en { \l__tblr_element_name_tl / fg } {#1} }
      {
        %% unknown key name has been changed to string in \l_keys_key_str
        \tl_set_rescan:Nnn \l__tblr_f_tl {} {#1}
        \tl_if_head_eq_catcode:VNTF \l__tblr_f_tl \scan_stop:
          {
            \__tblr_style_put:eV { \l__tblr_element_name_tl / font } \l__tblr_f_tl
          }
          {
            \__tblr_style_put:en { \l__tblr_element_name_tl / #1 } {#2}
          }
      }
  }
\cs_generate_variant:Nn \__tblr_element_unknown_key:nn { Vn }
```

9.29 Helper functions for templates

```
\tl_new:N \l__tblr_template_name_tl
\tl_new:N \l__tblr_template_code_tl

\__tblr_keys_define:nn { template/def }
  {
    unknown .code:n = \__tblr_def_template:V \l_keys_key_str,
  }

%% #1: head/foot element; #2: template name; #3: template code
%% If the template name = default, we enable the template at once
%% Otherwise, we may enable the template by using \SetTblrTemplate command
\NewDocumentCommand \DeclareTblrTemplate { m m +m }
  {
    \tl_set:Nn \l__tblr_template_name_tl {#2}
    \tl_set:Nn \l__tblr_template_code_tl {#3}
    \__tblr_keys_set:nn { template/def } {#1}
    \ignorespaces
  }
```

```
\cs_new_eq:NN \DefTblrTemplate \DeclareTblrTemplate

\cs_new_protected:Npn \__tblr_def_template:n #1
  {
    \tl_if_exist:cF { l__tblr_template_ #1 _ \l__tblr_template_name_tl _tl }
      { \tl_new:c { l__tblr_template_ #1 _ \l__tblr_template_name_tl _tl } }
    \tl_set_eq:cN { l__tblr_template_ #1 _ \l__tblr_template_name_tl _tl }
      \l__tblr_template_code_tl
  }
\cs_generate_variant:Nn \__tblr_def_template:n { V }

\__tblr_keys_define:nn { template/set }
  {
    unknown .code:n = \__tblr_set_template:V \l_keys_key_str,
  }

%% #1: head/foot element; #2: template name
\NewDocumentCommand \SetTblrTemplate { m m }
  {
    \tl_set:Nn \l__tblr_template_name_tl {#2}
    \__tblr_keys_set:nn { template/set } {#1}
    \ignorespaces
  }

\msg_new:nnn { tabularray } { template-undefined }
  {
    Undefined ~ template ~ "#2" ~ for ~ element ~ "#1".
  }
\cs_new_protected:Npn \__tblr_set_template:n #1
  {
    \tl_if_exist:cTF { l__tblr_template_ #1 _ \l__tblr_template_name_tl _tl }
      {
        \tl_if_exist:cF { l__tblr_template_ #1 _default_tl }
          { \tl_new:c { l__tblr_template_ #1 _default_tl } }
        \tl_set_eq:cc { l__tblr_template_ #1 _default_tl }
          { l__tblr_template_ #1 _ \l__tblr_template_name_tl _tl }
      }
      {
        \msg_error:nnee { tabularray } { template-undefined }
          { #1 } { \l__tblr_template_name_tl }
      }
  }
\cs_generate_variant:Nn \__tblr_set_template:n { V }

\NewExpandableDocumentCommand \GetTblrStyle { m m }
  {
    \__tblr_style_item:n { #1 / #2 }
  }

\NewDocumentCommand \UseTblrFont { m }
  {
    \GetTblrStyle {#1} { font } \selectfont
  }

\tl_new:N \l__tblr_use_color_tl
```

```
\NewDocumentCommand \UseTblrColor { m }
  {
    \tl_set:Ne \l__tblr_use_color_tl { \GetTblrStyle {#1} { fg } }
    \tl_if_empty:NF \l__tblr_use_color_tl { \color { \l__tblr_use_color_tl } }
  }

%% All halign commands are defined at the beginning of the file
\NewDocumentCommand \UseTblrAlign { m }
  {
    \use:c { __tblr_halign_command_ \GetTblrStyle {#1} { halign } : }
  }

\tl_new:N \l__tblr_use_hang_tl

\NewDocumentCommand \UseTblrHang { m }
  {
    \tl_set:Ne \l__tblr_use_hang_tl { \GetTblrStyle {#1} { hang } }
    \tl_if_empty:NF \l__tblr_use_hang_tl
      {
        \tl_put_left:Nn \l__tblr_use_hang_tl
          { \hangafter = 1 \relax \hangindent = }
        \tl_put_right:Nn \l__tblr_use_hang_tl { \relax }
        \exp_args:NV \everypar \l__tblr_use_hang_tl
      }
  }

\tl_new:N \l__tblr_use_indent_tl

\NewDocumentCommand \UseTblrIndent { m }
  {
    \tl_set:Ne \l__tblr_use_indent_tl { \GetTblrStyle {#1} { indent } }
    \tl_if_empty:NF \l__tblr_use_indent_tl
      { \exp_args:NNV \setlength \parindent \l__tblr_use_indent_tl }
  }

\AtBeginDocument
  {
    \@ifpackageloaded{xcolor}{}{\RenewDocumentCommand \UseTblrColor {m} {}}
  }

%% #1: head/foot element; #2: template name
\NewExpandableDocumentCommand \ExpTblrTemplate { m m }
  {
    \tl_use:c { l__tblr_template_ #1 _ #2 _tl }
  }

%% #1: head/foot element; #2: template name
\NewDocumentCommand \UseTblrTemplate { m m }
  {
    \group_begin:
    \UseTblrFont {#1}
    \UseTblrColor {#1}
    \tl_use:c { l__tblr_template_ #1 _ #2 _tl }
    \group_end:
  }
```

```
\NewDocumentCommand \MapTblrNotes { +m }
  {
    \__tblr_prop_map_inline:nn { note }
      {
        \tl_set_rescan:Nnn \InsertTblrNoteTag {} {##1}
        \tl_set:Nn \InsertTblrNoteText {##2}
        #1
      }
  }

\NewDocumentCommand \MapTblrRemarks { +m }
  {
    \__tblr_prop_map_inline:nn { remark }
      {
        \tl_set_rescan:Nnn \InsertTblrRemarkTag {} {##1}
        \tl_set:Nn \InsertTblrRemarkText {##2}
        #1
      }
  }

\NewExpandableDocumentCommand \InsertTblrText { m }
  {
    \__tblr_spec_item:nn { outer } {#1}
  }

\NewExpandableDocumentCommand \InsertTblrMore { m }
  {
    \__tblr_prop_item:nn { more } {#1}
  }
```

9.30 Table continuation templates

```
\tl_if_exist:NF \tblrcontfootname
  {
    \tl_new:N \tblrcontfootname
    \tl_set:Nn \tblrcontfootname { Continued ~ on ~ next ~ page }
  }

\tl_if_exist:NF \tblrcontheadname
  {
    \tl_new:N \tblrcontheadname
    \tl_set:Nn \tblrcontheadname { ( Continued ) }
  }

\DeclareTblrTemplate { contfoot-text } { normal } { \tblrcontfootname }
\SetTblrTemplate { contfoot-text } { normal }

\DeclareTblrTemplate { contfoot } { empty } { }
\DeclareTblrTemplate { contfoot } { plain }
  {
    \noindent
    \raggedleft
    \UseTblrTemplate { contfoot-text } { default }
    \par
  }
```

```
\DeclareTblrTemplate { contfoot } { normal }
  {
    %% need to set parindent after alignment
    \raggedleft
    \UseTblrAlign { contfoot }
    \UseTblrIndent { contfoot }
    \UseTblrHang { contfoot }
    \leavevmode
    \UseTblrTemplate { contfoot-text } { default }
    \par
  }
\SetTblrTemplate { contfoot } { normal }

\DeclareTblrTemplate { conthead-pre } { empty } { }
\DeclareTblrTemplate { conthead-pre } { normal } { \space }
\SetTblrTemplate { conthead-pre } { normal }

\DeclareTblrTemplate { conthead-text } { normal } { \tblrcontheadname }
\SetTblrTemplate { conthead-text } { normal }

\DeclareTblrTemplate { conthead } { empty } { }
\DeclareTblrTemplate { conthead } { plain }
  {
    \noindent
    \raggedright
    \UseTblrTemplate { conthead-text } { default }
    \par
  }
\DeclareTblrTemplate { conthead } { normal }
  {
    %% need to set parindent after alignment
    \raggedright
    \UseTblrAlign { conthead }
    \UseTblrIndent { conthead }
    \UseTblrHang { conthead }
    \leavevmode
    \UseTblrTemplate { conthead-text } { default }
    \par
  }
\SetTblrTemplate { conthead } { normal }
```

9.31 Table caption templates

```
\tl_new:N \l__tblr_caption_short_tl

\DeclareTblrTemplate { caption-lot } { empty } { }
\DeclareTblrTemplate { caption-lot } { normal }
  {
    \tl_if_empty:NTF \lTblrEntryTl
      { \tl_set_eq:NN \l__tblr_caption_short_tl \lTblrCaptionTl }
      { \tl_set_eq:NN \l__tblr_caption_short_tl \lTblrEntryTl }
    \addcontentsline { lot } { table }
      { \protect\numberline { \thetable } { \l__tblr_caption_short_tl } }
  }
\SetTblrTemplate { caption-lot } { normal }
```

```
%% We need to use \hspace and \enskip, but not ~ or \space,
%% since we want a correct hangindent caption paragraph.

\DeclareTblrTemplate { caption-tag } { empty } { }
\DeclareTblrTemplate { caption-tag } { normal } { \tablename\hspace{0.25em}\thetable }
\SetTblrTemplate { caption-tag } { normal }

\DeclareTblrTemplate { caption-sep } { empty } { }
\DeclareTblrTemplate { caption-sep } { normal } { : \enskip }
\SetTblrTemplate { caption-sep } { normal }

\DeclareTblrTemplate { caption-text } { empty } { }
\DeclareTblrTemplate { caption-text } { normal } { \InsertTblrText { caption } }
\SetTblrTemplate { caption-text } { normal }

\box_new:N \l__tblr_caption_box
\box_new:N \l__tblr_caption_left_box

\DeclareTblrTemplate { caption } { empty } { }
\DeclareTblrTemplate { caption } { plain }
  {
    \hbox_set:Nn \l__tblr_caption_box
      {
        \UseTblrTemplate { caption-tag } { default }
        \UseTblrTemplate { caption-sep } { default }
        \UseTblrTemplate { caption-text } { default }
      }
    \dim_compare:nNnTF { \box_wd:N \l__tblr_caption_box } > { \hsize }
      {
        \noindent
        \hbox_unpack:N \l__tblr_caption_box
        \par
      }
      {
        \centering
        \makebox [\hsize] [c] { \box_use:N \l__tblr_caption_box }
        \par
      }
  }
\DeclareTblrTemplate { caption } { normal }
  {
    \hbox_set:Nn \l__tblr_caption_box
      {
        \UseTblrTemplate { caption-tag } { default }
        \UseTblrTemplate { caption-sep } { default }
        \UseTblrTemplate { caption-text } { default }
      }
    \dim_compare:nNnTF { \box_wd:N \l__tblr_caption_box } > { \hsize }
      {
        \UseTblrAlign { caption }
        \UseTblrIndent { caption }
        \hbox_set:Nn \l__tblr_caption_left_box
          {
            \UseTblrTemplate { caption-tag } { default }
            \UseTblrTemplate { caption-sep } { default }
          }
        \hangindent = \box_wd:N \l__tblr_caption_left_box
```

```
        \hangafter = 1
        \UseTblrHang { caption }
        \leavevmode
        \hbox_unpack:N \l__tblr_caption_box
        \par
      }
      {
        \centering
        \makebox [\hsize] [c] { \box_use:N \l__tblr_caption_box }
        \par
      }
  }
\DeclareTblrTemplate { caption } { simple }
  {
    \UseTblrAlign { caption }
    \UseTblrIndent { caption }
    \UseTblrHang { caption }
    \leavevmode
    \UseTblrTemplate { caption-tag } { default }
    \UseTblrTemplate { caption-sep } { default }
    \UseTblrTemplate { caption-text } { default }
    \par
  }
\SetTblrTemplate { caption } { normal }

\DeclareTblrTemplate { capcont } { empty } { }
\DeclareTblrTemplate { capcont } { plain }
  {
    \hbox_set:Nn \l__tblr_caption_box
      {
        \UseTblrTemplate { caption-tag } { default }
        \UseTblrTemplate { caption-sep } { default }
        \UseTblrTemplate { caption-text } { default }
        \UseTblrTemplate { conthead-pre } { default }
        \UseTblrTemplate { conthead-text } { default }
      }
    \dim_compare:nNnTF { \box_wd:N \l__tblr_caption_box } > { \hsize }
      {
        \noindent
        \hbox_unpack:N \l__tblr_caption_box
        \par
      }
      {
        \centering
        \makebox [\hsize] [c] { \box_use:N \l__tblr_caption_box }
        \par
      }
  }
\DeclareTblrTemplate { capcont } { normal }
  {
    \hbox_set:Nn \l__tblr_caption_box
      {
        \UseTblrTemplate { caption-tag } { default }
        \UseTblrTemplate { caption-sep } { default }
        \UseTblrTemplate { caption-text } { default }
        \UseTblrTemplate { conthead-pre } { default }
        \UseTblrTemplate { conthead-text } { default }
      }
```

```
    \dim_compare:nNnTF { \box_wd:N \l__tblr_caption_box } > { \hsize }
      {
        \UseTblrAlign { capcont }
        \UseTblrIndent { capcont }
        \hbox_set:Nn \l__tblr_caption_left_box
          {
            \UseTblrTemplate { caption-tag } { default }
            \UseTblrTemplate { caption-sep } { default }
          }
        \hangindent = \box_wd:N \l__tblr_caption_left_box
        \hangafter = 1
        \UseTblrHang { capcont }
        \leavevmode
        \hbox_unpack:N \l__tblr_caption_box
        \par
      }
      {
        \centering
        \makebox [\hsize] [c] { \box_use:N \l__tblr_caption_box }
        \par
      }
  }
\DeclareTblrTemplate { capcont } { simple }
  {
    \UseTblrAlign { caption }
    \UseTblrIndent { caption }
    \UseTblrHang { caption }
    \leavevmode
    \UseTblrTemplate { caption-tag } { default }
    \UseTblrTemplate { caption-sep } { default }
    \UseTblrTemplate { caption-text } { default }
    \UseTblrTemplate { conthead-pre } { default }
    \UseTblrTemplate { conthead-text } { default }
    \par
  }
\SetTblrTemplate { capcont} { normal }
```

9.32 Table notes templates

```
%% By default the targets generated by \hypertarget are too low
%% Therefore we need to use \Hy@raisedlink command to fix this problem
%% See https://tex.stackexchange.com/questions/17057
%% We also use \use:c in case the private command \Hy@raisedlink is removed
\cs_new_protected:Npn \__tblr_hyper_target:n #1
  {
    \cs_if_exist:NT \hypertarget
      {
        \use:c { Hy@raisedlink }
          {
            \hypertarget
              { tblr / \int_use:N \c@tblrcount / \tl_to_str:n {#1} }
              { }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_hyper_target:n { V }
```

```
\cs_new_protected:Npn \__tblr_hyper_link:nn #1 #2
  {
    \cs_if_exist:NTF \hyperlink
      {
        \hyperlink
          { tblr / \int_use:N \c@tblrcount / \tl_to_str:n {#1} }
          { #2 }
      }
      { #2 }
  }

\DeclareTblrTemplate { note-border } { empty }
  {
    \hypersetup { pdfborder = { 0 ~ 0 ~ 0 } }
  }
\DeclareTblrTemplate { note-border } { normal }
  {
    \hypersetup { pdfborder = { 0 ~ 0 ~ 1 } }
  }
\SetTblrTemplate { note-border } { empty }

\cs_set_eq:NN \TblrOverlap \rlap

\NewDocumentCommand \TblrNote { m }
  {
    \cs_if_exist:NT \hypersetup { \ExpTblrTemplate { note-border }{ default } }
    \TblrOverlap
      {
        \__tblr_hyper_link:nn {#1}
          { \textsuperscript { \sffamily \UseTblrFont { note-tag } #1 } }
      }
  }

\DeclareTblrTemplate { note-tag } { empty } { }
\DeclareTblrTemplate { note-tag } { normal }
  {
    \textsuperscript { \sffamily \UseTblrFont { note-tag } \InsertTblrNoteTag }
  }
\SetTblrTemplate { note-tag } { normal }

\DeclareTblrTemplate { note-target } { normal }
  {
    \__tblr_hyper_target:V \InsertTblrNoteTag
  }
\SetTblrTemplate { note-target } { normal }

\DeclareTblrTemplate { note-sep } { empty } { }
\DeclareTblrTemplate { note-sep } { normal } { \space }
\SetTblrTemplate { note-sep } { normal }

\DeclareTblrTemplate { note-text } { empty } { }
\DeclareTblrTemplate { note-text } { normal } { \InsertTblrNoteText }
\SetTblrTemplate { note-text } { normal }

\DeclareTblrTemplate { note } { empty } { }
\DeclareTblrTemplate { note } { plain }
```

```
  {
    \MapTblrNotes
      {
        \noindent
        \UseTblrTemplate { note-tag } { default }
        \UseTblrTemplate { note-target } { default }
        \UseTblrTemplate { note-sep } { default }
        \UseTblrTemplate { note-text } { default }
        \par
      }
  }
\DeclareTblrTemplate { note } { normal }
  {
    \UseTblrAlign { note }
    \UseTblrIndent { note }
    \MapTblrNotes
      {
        \hangindent = 0.7em
        \hangafter = 1
        \UseTblrHang { note }
        \leavevmode
        \hbox_to_wd:nn { \the\hangindent }
          {
            \UseTblrTemplate { note-tag } { default }
            \UseTblrTemplate { note-target } { default }
            \hfil
          }
        \UseTblrTemplate { note-text } { default }
        \par
      }
  }
\DeclareTblrTemplate { note } { inline }
  {
    \UseTblrAlign { note }
    \UseTblrIndent { note }
    \UseTblrHang { note }
    \leavevmode
    \MapTblrNotes
      {
        \UseTblrTemplate { note-tag } { default }
        \UseTblrTemplate { note-target } { default }
        \UseTblrTemplate { note-sep } { default }
        \UseTblrTemplate { note-text } { default }
        \quad
      }
    \par
  }
\SetTblrTemplate { note } { normal }
```

9.33 Table remarks templates

```
\DeclareTblrTemplate { remark-tag } { empty } { }
\DeclareTblrTemplate { remark-tag } { normal }
  {
    \itshape \UseTblrFont { remark-tag } \InsertTblrRemarkTag
  }
\SetTblrTemplate { remark-tag } { normal }
```

```
\DeclareTblrTemplate { remark-sep } { empty } { }
\DeclareTblrTemplate { remark-sep } { normal } { : \space }
\SetTblrTemplate { remark-sep } { normal }

\DeclareTblrTemplate { remark-text } { empty } { }
\DeclareTblrTemplate { remark-text } { normal } { \InsertTblrRemarkText }
\SetTblrTemplate { remark-text } { normal }

\DeclareTblrTemplate { remark } { empty } { }
\DeclareTblrTemplate { remark } { plain }
  {
    \MapTblrRemarks
      {
        \noindent
        \UseTblrTemplate { remark-tag } { default }
        \UseTblrTemplate { remark-sep } { default }
        \UseTblrTemplate { remark-text } { default }
        \par
      }
  }
\DeclareTblrTemplate { remark } { normal }
  {
    \UseTblrAlign { remark }
    \UseTblrIndent { remark }
    \MapTblrRemarks
      {
        \hangindent = 0.7em
        \hangafter = 1
        \UseTblrHang { remark }
        \leavevmode
        \UseTblrTemplate { remark-tag } { default }
        \UseTblrTemplate { remark-sep } { default }
        \UseTblrTemplate { remark-text } { default }
        \par
      }
  }
\DeclareTblrTemplate { remark } { inline }
  {
    \UseTblrAlign { remark }
    \UseTblrIndent { remark }
    \UseTblrHang { remark }
    \leavevmode
    \MapTblrRemarks
      {
        \UseTblrTemplate { remark-tag } { default }
        \UseTblrTemplate { remark-sep } { default }
        \UseTblrTemplate { remark-text } { default }
        \quad
      }
    \par
  }
\SetTblrTemplate { remark } { normal }
```

9.34 Header and footer templates

```
\tl_new:N \g__tblr_template_firsthead_default_tl
\tl_new:N \g__tblr_template_middlehead_default_tl
\tl_new:N \g__tblr_template_lasthead_default_tl
\tl_new:N \g__tblr_template_firstfoot_default_tl
\tl_new:N \g__tblr_template_middlefoot_default_tl
\tl_new:N \g__tblr_template_lastfoot_default_tl

\__tblr_keys_define:nn { template/def }
  {
    head .meta:n = { firsthead, middlehead, lasthead },
    foot .meta:n = { firstfoot, middlefoot, lastfoot },
  }

\__tblr_keys_define:nn { template/set }
  {
    head .meta:n = { firsthead, middlehead, lasthead },
    foot .meta:n = { firstfoot, middlefoot, lastfoot },
  }

\DeclareTblrTemplate { head } { empty } { }
\DeclareTblrTemplate { foot } { empty } { }

\DeclareTblrTemplate { firsthead } { normal }
  {
    \UseTblrTemplate { caption } { default }
  }

\DeclareTblrTemplate { middlehead, lasthead } { normal }
  {
    \UseTblrTemplate { capcont } { default }
  }

\DeclareTblrTemplate { firstfoot, middlefoot } { normal }
  {
    \UseTblrTemplate { contfoot } { default }
  }

\DeclareTblrTemplate { lastfoot } { normal }
  {
    \UseTblrTemplate { note } { default }
    \UseTblrTemplate { remark } { default }
  }

\SetTblrTemplate { head } { normal }
\SetTblrTemplate { foot } { normal }
```

9.35 Build the whole table

```
\cs_new:Npn \__tblr_box_height:N #1
  {
    \dim_eval:n { \box_ht:N #1 + \box_dp:N #1 }
  }
```

```
\cs_new_protected:Npn \__tblr_build_head_foot:
  {
    \__tblr_build_row_head_foot:
    \__tblr_build_table_head_foot:
  }

\int_new:N \lTblrRowHeadInt
\int_new:N \lTblrRowFootInt
\box_new:N \l__tblr_row_head_box
\box_new:N \l__tblr_row_foot_box
\dim_new:N \l__tblr_row_head_foot_dim

\cs_new_protected:Npn \__tblr_build_row_head_foot:
  {
    %% \lTblrRowHeadInt could not be empty, so we append '+ 0'.
    \int_set:Nn \lTblrRowHeadInt
      { \__tblr_prop_item:ne { inner } { rowhead } + 0 }
    \int_compare:nNnTF { \lTblrRowHeadInt } > { 0 }
      {
        \__tblr_build_one_table:nnNN {1} { \lTblrRowHeadInt }
          \c_true_bool \c_true_bool
      }
      { \__tblr_build_one_hline:n {1} }
    \box_set_eq:NN \l__tblr_row_head_box \l__tblr_table_box
    %% \lTblrRowFootInt could not be empty, so we append '+ 0'.
    \int_set:Nn \lTblrRowFootInt
      { \__tblr_prop_item:ne { inner } { rowfoot } + 0 }
    \int_compare:nNnTF { \lTblrRowFootInt } > { 0 }
      {
        \__tblr_build_one_table:nnNN
          { \c@rowcount - \lTblrRowFootInt + 1 } { \c@rowcount }
          \c_true_bool \c_true_bool
      }
      { \__tblr_build_one_hline:n { \int_eval:n { \c@rowcount + 1 } } }
    \box_set_eq:NN \l__tblr_row_foot_box \l__tblr_table_box
    \dim_set:Nn \l__tblr_row_head_foot_dim
      {
        \__tblr_box_height:N \l__tblr_row_head_box
          + \__tblr_box_height:N \l__tblr_row_foot_box
      }
  }

\dim_new:N \lTblrTableWidthDim
\cs_set_eq:NN \tablewidth \lTblrTableWidthDim

\cs_new_protected:Npn \__tblr_get_table_width:
  {
    \dim_zero:N \lTblrTableWidthDim
    \int_step_inline:nn { \c@colcount }
      {
        \dim_add:Nn \lTblrTableWidthDim
          {
            \__tblr_spec_item:nn { vline } { [##1] / @vline-width }
            +
            \__tblr_data_item:nnn { column } {##1} { leftsep }
            +
            \__tblr_data_item:nnn { column } {##1} { @col-width }
```

```
            +
            \__tblr_data_item:nnn { column } {##1} { rightsep }
          }
      }
    \dim_add:Nn \lTblrTableWidthDim
      {
        \__tblr_spec_item:ne { vline }
          { [\int_eval:n { \c@colcount + 1 }] / @vline-width }
      }
  }

\box_new:N \l__tblr_table_firsthead_box
\box_new:N \l__tblr_table_middlehead_box
\box_new:N \l__tblr_table_lasthead_box
\box_new:N \l__tblr_table_firstfoot_box
\box_new:N \l__tblr_table_middlefoot_box
\box_new:N \l__tblr_table_lastfoot_box

\cs_new_protected:Npn \__tblr_build_table_head_foot:
  {
    \__tblr_get_table_width:
    % make each of \lTblrCaptionTl, \lTblrEntryTl, \lTblrLabelTl and the
    % three corresponding booleans available in all head-foot templates
    \__tblr_set_table_label_entry:
    \__tblr_build_table_head_aux:Nn \l__tblr_table_firsthead_box
      {
        \__tblr_build_table_label_entry:
        \UseTblrTemplate { firsthead } { default }
      }
    \__tblr_build_table_head_aux:Nn \l__tblr_table_middlehead_box
      {
        \UseTblrTemplate { middlehead } { default }
      }
    \__tblr_build_table_head_aux:Nn \l__tblr_table_lasthead_box
      {
        \UseTblrTemplate { lasthead } { default }
      }
    \__tblr_build_table_foot_aux:Nn \l__tblr_table_firstfoot_box
      {
        \UseTblrTemplate { firstfoot } { default }
      }
    \__tblr_build_table_foot_aux:Nn \l__tblr_table_middlefoot_box
      {
        \UseTblrTemplate { middlefoot } { default }
      }
    \__tblr_build_table_foot_aux:Nn \l__tblr_table_lastfoot_box
      {
        \UseTblrTemplate { lastfoot } { default }
      }
  }

\bool_new:N \l__tblr_table_no_title_bool
\bool_new:N \l__tblr_table_no_entry_bool
\bool_new:N \l__tblr_table_no_label_bool
\tl_const:Nn \cTblrNoneTl { none }

\cs_new_protected:Npn \__tblr_set_table_label_entry:
```

```
  {
    \tl_set:Ne \lTblrCaptionTl { \InsertTblrText { caption } }
    \tl_set:Ne \lTblrEntryTl { \InsertTblrText { entry } }
    \tl_set:Ne \lTblrLabelTl { \InsertTblrText { label } }
    \bool_set:Nn \l__tblr_table_no_title_bool
      { \tl_if_empty_p:N \lTblrCaptionTl }
    \bool_set:Nn \l__tblr_table_no_entry_bool
      { \tl_if_eq_p:NN \lTblrEntryTl \cTblrNoneTl }
    \bool_set:Nn \l__tblr_table_no_label_bool
      { \tl_if_eq_p:NN \lTblrLabelTl \cTblrNoneTl }
    \bool_if:NT \l__tblr_table_no_title_bool
      {
        \SetTblrTemplate { conthead-pre } { empty }
      }
    \bool_if:NT \l__tblr_table_no_label_bool
      {
        \SetTblrTemplate { caption-tag }{ empty }
        \SetTblrTemplate { caption-sep }{ empty }
      }
  }

\cs_new_protected:Npn \__tblr_build_tall_table_head_foot:
  {
    \__tblr_get_table_width:
    \__tblr_set_table_label_entry:
    \__tblr_build_table_head_aux:Nn \l__tblr_table_firsthead_box
      {
        \__tblr_build_table_label_entry:
        \UseTblrTemplate { firsthead } { default }
      }
    \__tblr_build_table_foot_aux:Nn
      \l__tblr_table_lastfoot_box  { \UseTblrTemplate { lastfoot } { default } }
  }

\tl_new:N \lTblrCaptionTl
\tl_new:N \lTblrEntryTl
\tl_new:N \lTblrLabelTl
\clist_new:N \lTblrRefMoreClist

\cs_new_protected:Npn \__tblr_build_table_label_entry:
  {
    \bool_if:NF \l__tblr_table_no_label_bool
      {
        \refstepcounter { table }
        \tl_if_empty:NF \lTblrLabelTl
          {
            \clist_map_inline:Nn \lTblrRefMoreClist
              { \ExpTblrTemplate { caption-ref } { ##1 } }
            \exp_args:NV \label \lTblrLabelTl
          }
      }
    %% We put caption-lot code at last, so that a user can modify \lTblrEntryTl
    %% in a caption-label template. For example, a user may want to use
    %% short caption in nameref, but at the same time not to add LoT entry.
    \bool_if:NF \l__tblr_table_no_entry_bool
      { \UseTblrTemplate { caption-lot } { default } }
  }
```

```
\cs_new_protected:Npn \__tblr_build_table_head_aux:Nn #1 #2
  {
    \vbox_set:Nn #1
      {
        \hsize = \lTblrTableWidthDim
        \TblrParboxRestore % it will set \linewidth = \hsize
        \vbox_set:Nn \l_tmpa_box {#2}
        \box_use:N \l_tmpa_box
        \dim_compare:nNnT
          { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box } > { 0pt }
          { \skip_vertical:n { \__tblr_spec_item:nn { outer } { headsep } } }
      }
  }

\cs_new_protected:Npn \__tblr_build_table_foot_aux:Nn #1 #2
  {
    \vbox_set:Nn #1
      {
        \hsize = \lTblrTableWidthDim
        \TblrParboxRestore % it will set \linewidth = \hsize
        \vbox_set:Nn \l_tmpb_box {#2}
        \dim_compare:nNnT
          { \box_ht:N \l_tmpb_box + \box_dp:N \l_tmpb_box } > { 0pt }
          { \skip_vertical:n { \__tblr_spec_item:nn { outer } { footsep } } }
        \box_use:N \l_tmpb_box
      }
  }

\tl_new:N \lTblrPortraitTypeTl

\cs_new_protected:Npn \__tblr_build_whole:
  {
    \__tblr_hook_use:n { table/before }
    \tl_set:Ne \lTblrPortraitTypeTl
      { \__tblr_spec_item:nn { outer } { portrait } }
    \tl_if_eq:NnTF \lTblrPortraitTypeTl { long }
      {
        \__tblr_build_long_table:e { \__tblr_spec_item:nn { outer } { halign } }
      }
      {
        \tl_if_eq:NnTF \lTblrPortraitTypeTl { tall }
          {
            \__tblr_build_tall_table:e
              { \__tblr_spec_item:nn { outer } { baseline } }
          }
          {
            \tl_set:Nn \lTblrPortraitTypeTl { short }
            \__tblr_build_short_table:e
              { \__tblr_spec_item:nn { outer } { baseline } }
          }
      }
    \__tblr_hook_use:n { table/after }
  }

\dim_new:N \l__tblr_remain_height_dim
\dim_new:N \l__tblr_next_rows_dim
\dim_new:N \l__tblr_page_overfill_dim
```

```
\int_new:N \l__tblr_long_from_int
\int_new:N \l__tblr_long_to_int
\int_new:N \l__tblr_curr_i_int
\int_new:N \l__tblr_prev_i_int
\int_new:N \lTblrTablePageInt
\bool_new:N \l__tblr_page_break_curr_bool
\bool_new:N \l__tblr_page_break_prev_bool

%% #1: table alignment
%% For long table, we need to leave hmode first to get correct \pagetotal
%% Also remove topskip and presep if we are at the beginning of the page
\cs_new_protected:Npn \__tblr_build_long_table:n #1
  {
    \LogTblrTracing { page }
    \par
    \skip_zero:N \parskip % see issue #203
    \LogTblrTracing { page }
    \dim_compare:nNnTF { \pagegoal } = { \maxdimen }
      { \hbox{}\kern-\topskip\nobreak }
      { \skip_vertical:n { \__tblr_spec_item:nn { outer } { presep } } }
    \LogTblrTracing { page }
    \nointerlineskip
    \mode_leave_vertical: % enter horizontal mode to update \pagetotal
    \LogTblrTracing { page }
    \hrule height ~ 0pt
    \nobreak % prevent page break after \hrule (see issue #42)
    \LogTblrTracing { page }
    \int_set:Nn \lTblrTablePageInt {1}
    \__tblr_build_head_foot:
    \dim_set:Nn \l__tblr_remain_height_dim
      { \pagegoal - \pagetotal - \l__tblr_row_head_foot_dim }
    \int_set:Nn \l__tblr_long_from_int { \lTblrRowHeadInt + 1 }
    \int_set:Nn \l__tblr_long_to_int { \c@rowcount - \lTblrRowFootInt }
    \int_set:Nn \l__tblr_curr_i_int { \l__tblr_long_from_int - 1 }
    \int_do_while:nNnn { \l__tblr_curr_i_int } < { \l__tblr_long_to_int }
      {
        \int_set_eq:NN \l__tblr_prev_i_int \l__tblr_curr_i_int
        \__tblr_get_next_table_rows:NNNN
          \l__tblr_long_to_int \l__tblr_curr_i_int
          \l__tblr_next_rows_dim \l__tblr_page_break_curr_bool
        \__tblr_check_table_page_break:NNN
          \l__tblr_remain_height_dim
          \l__tblr_next_rows_dim
          \l__tblr_page_break_prev_bool
        \__tblr_do_if_tracing:nn { page } { \int_log:N \l__tblr_curr_i_int }
        \bool_if:NTF \l__tblr_page_break_prev_bool
          {
            \int_compare:nNnTF
              { \l__tblr_long_from_int } > { \l__tblr_prev_i_int }
              {
                % See issue #42: if longtblr starts at the bottom of a page,
                % \pagetotal maybe exceed \pagegoal after adding presep,
                % or after adding rowhead or rowfoot of the table.
                % In these cases, we will not typeset table in this page,
                % but rather force a page break.
                \dim_set:Nn \l__tblr_page_overfill_dim
                  {
                    % Issue #361 and #536: overfill the page (including shrink).
```

```
                \pagegoal - \pagetotal + \pageshrink + 1sp
              }
            \skip_vertical:n { \l__tblr_page_overfill_dim }
            \tex_penalty:D 9999
            \skip_vertical:n { -\l__tblr_page_overfill_dim }
          }
          {
            \__tblr_build_page_table:nee {#1}
              { \int_use:N \l__tblr_long_from_int }
              { \int_use:N \l__tblr_prev_i_int }
            \int_incr:N \lTblrTablePageInt
            \int_set:Nn \l__tblr_long_from_int { \l__tblr_prev_i_int + 1 }
            \TblrNewPage
          }
        \hbox{}\kern-\topskip\nobreak
        \noindent
        \LogTblrTracing { page }
        \dim_set:Nn \l__tblr_remain_height_dim
          {
            \pagegoal - \pagetotal
            - \l__tblr_row_head_foot_dim - \l__tblr_next_rows_dim
          }
      }
      {
        \bool_if:NTF \l__tblr_page_break_curr_bool
          {
            \__tblr_build_page_table:nee {#1}
              { \int_use:N \l__tblr_long_from_int }
              { \int_use:N \l__tblr_curr_i_int }
            \int_incr:N \lTblrTablePageInt
            \TblrNewPage
            \hbox{}\kern-\topskip\nobreak
            \noindent
            \LogTblrTracing { page }
            \dim_set:Nn \l__tblr_remain_height_dim
              { \pagegoal - \pagetotal - \l__tblr_row_head_foot_dim }
            \int_set:Nn \l__tblr_long_from_int { \l__tblr_curr_i_int + 1 }
          }
          {
            \dim_add:Nn \l__tblr_remain_height_dim { -\l__tblr_next_rows_dim }
          }
      }
  }
\int_compare:nNnTF { \lTblrTablePageInt } = {1}
  {
    \box_set_eq:NN \l__tblr_table_head_box \l__tblr_table_firsthead_box
    \box_set_eq:NN \l__tblr_table_foot_box \l__tblr_table_lastfoot_box
  }
  {
    \box_set_eq:NN \l__tblr_table_head_box \l__tblr_table_lasthead_box
    \box_set_eq:NN \l__tblr_table_foot_box \l__tblr_table_lastfoot_box
  }
\__tblr_build_page_table:nee {#1}
  { \int_use:N \l__tblr_long_from_int }
  { \int_use:N \l__tblr_long_to_int }
\skip_vertical:n { \__tblr_spec_item:nn { outer } { postsep } }
% In the past we used "\hrule height ~ 0pt" to get strict postsep,
% but the postsep was not discarded when page breaks, see issue #39.
```

```
    % Therefore we use \nointerlineskip here.
    \nointerlineskip
  }
\cs_generate_variant:Nn \__tblr_build_long_table:n { e }

%% #1: int with index of the last row; #2: int with index of current row;
%% #3: row dimension; #4: break page or not.
\cs_new_protected:Npn \__tblr_get_next_table_rows:NNNN #1 #2 #3 #4
  {
    \bool_set_true:N \l_tmpa_bool
    \dim_zero:N #3
    \bool_set_false:N #4
    \bool_while_do:Nn \l_tmpa_bool
      {
        \int_incr:N #2
        \dim_add:Nn #3
          {
            \__tblr_data_item:nen { row } { \int_use:N #2 } { abovesep }
            +
            \__tblr_data_item:nen { row } { \int_use:N #2 } { @row-height }
            +
            \__tblr_data_item:nen { row } { \int_use:N #2 } { belowsep }
            +
            \__tblr_spec_item:ne { hline }
              { [ \int_eval:n { #2 + 1 } ] / @hline-height }
          }
        \int_compare:nNnTF {#2} < {#1}
          {
            \tl_set:Ne \l__tblr_b_tl
              {
                \__tblr_spec_item:ne { hline }
                  { [ \int_eval:n { #2 + 1 } ] / @pagebreak }
              }
            % Note that \l__tblr_b_tl may be empty
            \int_compare:nNnTF { \l__tblr_b_tl + 0 } < { 0 }
              { \bool_set_true:N \l_tmpa_bool }
              {
                \bool_set_false:N \l_tmpa_bool
                \int_compare:nNnT { \l__tblr_b_tl + 0 } > { 0 }
                  { \bool_set_true:N #4 }
              }
          }
          { \bool_set_false:N \l_tmpa_bool }
      }
  }

\box_new:N \l__tblr_table_head_box
\box_new:N \l__tblr_table_foot_box
\dim_new:N \l__tblr_table_head_foot_dim
\dim_new:N \l__tblr_table_head_body_foot_dim

%% #1: remain dimension; #2: row dimension; #3: break page or not
\cs_new_protected:Npn \__tblr_check_table_page_break:NNN #1 #2 #3
  {
    \int_compare:nNnTF { \lTblrTablePageInt } = {1}
      {
        \dim_set:Nn \l__tblr_table_head_body_foot_dim
```

```
          {
            \__tblr_box_height:N \l__tblr_table_firsthead_box
              + #2 + \__tblr_box_height:N \l__tblr_table_firstfoot_box
          }
        \box_set_eq:NN \l__tblr_table_head_box \l__tblr_table_firsthead_box
        \dim_compare:nNnTF
          { \l__tblr_table_head_body_foot_dim } > {#1}
          {
            \bool_set_true:N #3
            \box_set_eq:NN \l__tblr_table_foot_box \l__tblr_table_firstfoot_box
          }
          { \bool_set_false:N #3 }
      }
      {
        \dim_set:Nn \l__tblr_table_head_body_foot_dim
          {
            \__tblr_box_height:N \l__tblr_table_middlehead_box
              + #2 + \__tblr_box_height:N \l__tblr_table_middlefoot_box
          }
        \box_set_eq:NN \l__tblr_table_head_box \l__tblr_table_middlehead_box
        \dim_compare:nNnTF
          { \l__tblr_table_head_body_foot_dim } > {#1}
          {
            \bool_set_true:N #3
            \box_set_eq:NN \l__tblr_table_foot_box \l__tblr_table_middlefoot_box
          }
          { \bool_set_false:N #3 }
      }
  }

\box_new:N \l__tblr_table_box
\int_new:N \lTblrRowFirstInt
\int_new:N \lTblrRowLastInt

%% #1: table alignment; #2: row from; #3: row to
\cs_new_protected:Npn \__tblr_build_page_table:nnn #1 #2 #3
  {
    \int_set:Nn \lTblrRowFirstInt {#2}
    \int_set:Nn \lTblrRowLastInt {#3}
    \__tblr_build_one_table:nnNN {#2} {#3} \c_false_bool \c_false_bool
    \vbox_set:Nn \l__tblr_table_box
      {
        \box_use:N \l__tblr_table_head_box
        \__tblr_cover_two_vboxes:NN \l__tblr_row_head_box \l__tblr_table_box
        \box_use:N \l__tblr_row_foot_box
        \hrule height ~ 0pt
        \box_use:N \l__tblr_table_foot_box
      }
    \__tblr_hook_use:n { private/output/before }
    \__tblr_halign_whole:Nn \l__tblr_table_box {#1}
    \__tblr_hook_use:n { private/output/after }
  }
\cs_generate_variant:Nn \__tblr_build_page_table:nnn { nee }

\dim_new:N \l__tblr_cover_box_a_dim
\dim_new:N \l__tblr_cover_box_b_dim
```

```
%% To solve the problem of missing hlines of long tables in some PDF readers,
%% We need to draw body rows before head rows (see issue #88).
\cs_new_protected:Npn \__tblr_cover_two_vboxes:NN #1 #2
  {
    \dim_set:Nn \l__tblr_cover_box_a_dim { \box_ht:N #1 + \box_dp:N #1 }
    \dim_set:Nn \l__tblr_cover_box_b_dim { \box_ht:N #2 + \box_dp:N #2 }
    \skip_vertical:N \l__tblr_cover_box_a_dim
    \hrule height ~ 0pt
    \box_use:N #2
    \skip_vertical:n { - \l__tblr_cover_box_a_dim - \l__tblr_cover_box_b_dim }
    \hrule height ~ 0pt
    \box_use:N #1
    \skip_vertical:N \l__tblr_cover_box_b_dim
    \hrule height ~ 0pt
  }

\cs_new_protected:Npn \__tblr_halign_whole:Nn #1 #2
  {
    \noindent
    \hbox_to_wd:nn { \linewidth }
      {
        \tl_if_eq:nnF {#2} {l} { \hfil }
        \box_use:N #1
        \tl_if_eq:nnF {#2} {r} { \hfil }
      }
  }

%% #1: table alignment
%% For tall table, we need to leave vmode first.
%% Since there may be \centering in table environment,
%% We use \raggedright to reset alignment for table head/foot.
\cs_new_protected:Npn \__tblr_build_tall_table:n #1
  {
    \mode_leave_vertical:
    \__tblr_build_tall_table_head_foot:
    \__tblr_build_one_table:nnNN {1} {\c@rowcount} \c_true_bool \c_true_bool
    \vbox_set:Nn \l__tblr_table_box
      {
        \box_use:N \l__tblr_table_firsthead_box
        \hrule height ~ 0pt
        \box_use:N \l__tblr_table_box
        \hrule height ~ 0pt
        \box_use:N \l__tblr_table_lastfoot_box
      }
    \__tblr_hook_use:n { private/output/before }
    \__tblr_valign_whole:Nn \l__tblr_table_box {#1}
    \__tblr_hook_use:n { private/output/after }
  }
\cs_generate_variant:Nn \__tblr_build_tall_table:n { e }

%% #1: table alignment
%% For short table, we need to leave vmode first
\cs_new_protected:Npn \__tblr_build_short_table:n #1
  {
    \mode_leave_vertical:
    \__tblr_build_one_table:nnNN {1} {\c@rowcount} \c_true_bool \c_true_bool
    \__tblr_hook_use:n { private/output/before }
```

```
    \__tblr_valign_whole:Nn \l__tblr_table_box {#1}
    \__tblr_hook_use:n { private/output/after }
  }
\cs_generate_variant:Nn \__tblr_build_short_table:n { e }

\box_new:N \l__tblr_table_hlines_box
\box_new:N \l__tblr_hline_box
\box_new:N \l__tblr_row_box

%% #1: row from; #2: row to
%% #3: whether build first hline or not; #4: whether build last hline or not
%% To fix disappeared hlines with colorful tables in Adobe Reader (see #76),
%% we collect all hlines and draw them at the end of the table.
\cs_new_protected:Npn \__tblr_build_one_table:nnNN #1 #2 #3 #4
  {
    \box_clear:N \l__tblr_table_hlines_box
    \tblr_vbox_set:Nn \l__tblr_table_box
      {
        \int_step_variable:nnNn {#1} {#2} \l__tblr_i_tl
          {
            \bool_lazy_or:nnT
              { \int_compare_p:nNn { \l__tblr_i_tl } > {#1} }
              { \bool_if_p:N #3 }
              { \__tblr_put_one_hline:n { \__tblr_build_hline:V \l__tblr_i_tl } }
            \tblr_hrule_ht:n { 0pt } % remove lineskip between hlines and rows
            \__tblr_put_one_row:n { \__tblr_build_row:N \l__tblr_i_tl }
            \tblr_hrule_ht:n { 0pt }
          }
        \bool_if:NT #4
          {
            \__tblr_put_one_hline:n
              { \__tblr_build_hline:n { \int_eval:n {#2 + 1} } }
          }
        \skip_vertical:n
          {
            - \box_ht:N \l__tblr_table_hlines_box
            - \box_dp:N \l__tblr_table_hlines_box
          }
        \tblr_box_use:N \l__tblr_table_hlines_box
      }
  }

\cs_new_protected:Npn \__tblr_put_one_hline:n #1
  {
    \hbox_set:Nn \l__tblr_hline_box {#1}
    \skip_vertical:n { \box_ht:N \l__tblr_hline_box + \box_dp:N \l__tblr_hline_box }
    \vbox_set:Nn \l__tblr_table_hlines_box
      {
        \vbox_unpack:N \l__tblr_table_hlines_box
        \box_use:N \l__tblr_hline_box
      }
  }

\cs_new_protected:Npn \__tblr_put_one_row:n #1
  {
    \hbox_set:Nn \l__tblr_row_box {#1}
    \vbox_set:Nn \l__tblr_table_hlines_box
```

```
      {
        \vbox_unpack:N \l__tblr_table_hlines_box
        \skip_vertical:n
          { \box_ht:N \l__tblr_row_box + \box_dp:N \l__tblr_row_box }
      }
    \box_use:N \l__tblr_row_box
  }

%% #1: hline number
\cs_new_protected:Npn \__tblr_build_one_hline:n #1
  {
    \vbox_set:Nn \l__tblr_table_box { \hbox:n { \__tblr_build_hline:n { #1 } } }
  }

\tl_new:N \l__tblr_vbox_align_tl
\tl_const:Nn \c__tblr_vbox_t_tl {t}
\tl_const:Nn \c__tblr_vbox_T_tl {T}
\tl_const:Nn \c__tblr_vbox_m_tl {m}
\tl_const:Nn \c__tblr_vbox_M_tl {M}
\tl_const:Nn \c__tblr_vbox_c_tl {c}
\tl_const:Nn \c__tblr_vbox_b_tl {b}
\tl_const:Nn \c__tblr_vbox_B_tl {B}

\tl_new:N \l__tblr_delim_left_tl
\tl_new:N \l__tblr_delim_right_tl

\cs_new_protected:Npn \__tblr_valign_whole:Nn #1 #2
  {
    \group_begin:
    \tl_set:Ne \l__tblr_delim_left_tl
      { \__tblr_prop_item:nn { inner } { delim-left } }
    \tl_set:Ne \l__tblr_delim_right_tl
      { \__tblr_prop_item:nn { inner } { delim-right } }
    \tl_set:Nn \l__tblr_vbox_align_tl {#2}
    \dim_set:Nn \l__tblr_t_dim { \box_ht:N #1 + \box_dp:N #1 }
    \tl_case:NnF \l__tblr_vbox_align_tl
      {
        \c__tblr_vbox_m_tl
          { \__tblr_valign_whole_middle:N #1 }
        \c__tblr_vbox_c_tl
          { \__tblr_valign_whole_middle:N #1 }
        \c__tblr_vbox_M_tl
          { \__tblr_valign_whole_middle_row_or_border:N #1 }
        \c__tblr_vbox_t_tl
          { \__tblr_valign_whole_top:N #1 }
        \c__tblr_vbox_T_tl
          {
            \tl_set:Nn \l__tblr_vbox_align_tl {1}
            \__tblr_valign_whole_at_row_from_above:N #1
          }
        \c__tblr_vbox_b_tl
          { \__tblr_valign_whole_bottom:N #1 }
        \c__tblr_vbox_B_tl
          {
            \tl_set:Ne \l__tblr_vbox_align_tl { \int_use:N \c@rowcount }
            \__tblr_valign_whole_at_row_from_below:N #1
          }
```

```
      }
      {
        \__tblr_if_positive_value:VTF \l__tblr_vbox_align_tl
          { \__tblr_valign_whole_at_row:N #1 }
          {
            \__tblr_if_negative_value:VTF \l__tblr_vbox_align_tl
              { \__tblr_valign_whole_at_border:N #1 }
              { \__tblr_valign_whole_middle:N #1 }
          }
      }
    %% we have done the job when valign is m or c
    \box_if_empty:NF #1 { \__tblr_add_delimiters_to_box:N #1 }
    \group_end:
  }

%% We use the idea of delarray package to shift table box
%% when there are delimiters around the table
\cs_new_protected:Npn \__tblr_add_delimiters_to_box:N #1
  {
    \tl_if_empty:NTF \l__tblr_delim_left_tl
      { \box_use_drop:N #1 }
      {
        \box_move_down:nn
          {
            ( \box_dp:N #1 - \box_ht:N #1 ) / 2
            + \tex_fontdimen:D 22 \tex_textfont:D 2
          }
          { \__tblr_get_vcenter_box:N #1 }
      }
  }

\cs_new_protected:Npn \__tblr_get_vcenter_box:N #1
  {
    \hbox:n
      {
        $ \m@th \l__tblr_delim_left_tl
        \tex_vcenter:D { \vbox_unpack_drop:N #1 }
        \l__tblr_delim_right_tl $
      }
  }

\cs_new_protected:Npn \__tblr_valign_whole_middle:N #1
  {
    \__tblr_get_vcenter_box:N #1
  }

\cs_new_protected:Npn \__tblr_valign_whole_top:N #1
  {
    \dim_set:Nn \l__tblr_h_dim { \__tblr_valign_get_hline_total:n {1} }
    \dim_compare:nNnT \l__tblr_h_dim = { 0pt }
      { \dim_add:Nn \l__tblr_h_dim { \__tblr_valign_get_row_height:n {1} } }
    \box_set_ht:Nn #1 { \l__tblr_h_dim }
    \box_set_dp:Nn #1 { \l__tblr_t_dim - \l__tblr_h_dim }
  }

\cs_new_protected:Npn \__tblr_valign_whole_bottom:N #1
  {
```

```
    \dim_set:Nn \l__tblr_d_dim
      { \__tblr_valign_get_hline_total:n { \int_eval:n { \c@rowcount + 1 } } }
    \dim_compare:nNnTF \l__tblr_d_dim = { 0pt }
      {
        \dim_set:Nn \l__tblr_d_dim
          { \__tblr_valign_get_row_depth:n { \int_use:N \c@rowcount } }
      }
      { \dim_zero:N \l__tblr_d_dim }
    \box_set_ht:Nn #1 { \l__tblr_t_dim - \l__tblr_d_dim }
    \box_set_dp:Nn #1 { \l__tblr_d_dim }
  }

\cs_new_protected:Npn \__tblr_valign_whole_middle_row_or_border:N #1
  {
    \int_if_odd:nTF { \c@rowcount }
      {
        \tl_set:Ne \l__tblr_vbox_align_tl { \int_eval:n { (\c@rowcount + 1) / 2 } }
        \__tblr_valign_whole_at_row_from_above:N #1
      }
      {
        \tl_set:Ne \l__tblr_vbox_align_tl { \int_eval:n { \c@rowcount / 2 + 1 } }
        \__tblr_valign_whole_at_border_from_above:N #1
      }
  }

\cs_new_protected:Npn \__tblr_valign_whole_at_row:N #1
  {
    \int_compare:nNnTF { 2 * \l__tblr_vbox_align_tl } > { \c@rowcount }
      { \__tblr_valign_whole_at_row_from_below:N #1 }
      { \__tblr_valign_whole_at_row_from_above:N #1 }
  }

\cs_new_protected:Npn \__tblr_valign_whole_at_row_from_above:N #1
  {
    \dim_set:Nn \l__tblr_h_dim
      { \__tblr_valign_get_hline_total:n { \l__tblr_vbox_align_tl } }
    \dim_add:Nn \l__tblr_h_dim
      { \__tblr_valign_get_row_height:n { \l__tblr_vbox_align_tl } }
    \int_step_inline:nn { \l__tblr_vbox_align_tl - 1 }
      {
        \dim_add:Nn \l__tblr_h_dim { \__tblr_valign_get_hline_total:n {##1} }
        \dim_add:Nn \l__tblr_h_dim { \__tblr_valign_get_row_total:n {##1} }
      }
    \box_set_ht:Nn #1 { \l__tblr_h_dim }
    \box_set_dp:Nn #1 { \l__tblr_t_dim - \l__tblr_h_dim }
  }

\cs_new_protected:Npn \__tblr_valign_whole_at_row_from_below:N #1
  {
    \dim_set:Nn \l__tblr_d_dim
      { \__tblr_valign_get_hline_total:n { \int_eval:n {\c@rowcount + 1} } }
    \dim_add:Nn \l__tblr_d_dim
      { \__tblr_valign_get_row_depth:n { \l__tblr_vbox_align_tl } }
    \int_step_inline:nnn { \l__tblr_vbox_align_tl + 1 } { \c@rowcount }
      {
        \dim_add:Nn \l__tblr_d_dim { \__tblr_valign_get_hline_total:n {##1} }
        \dim_add:Nn \l__tblr_d_dim { \__tblr_valign_get_row_total:n {##1} }
```

```
      }
    \box_set_dp:Nn #1 { \l__tblr_d_dim }
    \box_set_ht:Nn #1 { \l__tblr_t_dim - \l__tblr_d_dim }
  }

\cs_new_protected:Npn \__tblr_valign_whole_at_border:N #1
  {
    \tl_set:Ne \l__tblr_vbox_align_tl { \int_eval:n { - \l__tblr_vbox_align_tl } }
    \int_compare:nNnTF { 2 * \l__tblr_vbox_align_tl - 2 } > { \c@rowcount }
      { \__tblr_valign_whole_at_border_from_below:N #1 }
      { \__tblr_valign_whole_at_border_from_above:N #1 }
  }

\cs_new_protected:Npn \__tblr_valign_whole_at_border_from_above:N #1
  {
    \dim_set:Nn \l__tblr_h_dim
      { \__tblr_valign_get_hline_total:n { \l__tblr_vbox_align_tl } }
    \int_step_inline:nn { \l__tblr_vbox_align_tl - 1 }
      {
        \dim_add:Nn \l__tblr_h_dim { \__tblr_valign_get_hline_total:n {##1} }
        \dim_add:Nn \l__tblr_h_dim { \__tblr_valign_get_row_total:n {##1} }
      }
    \box_set_ht:Nn #1 { \l__tblr_h_dim }
    \box_set_dp:Nn #1 { \l__tblr_t_dim - \l__tblr_h_dim }
  }

\cs_new_protected:Npn \__tblr_valign_whole_at_border_from_below:N #1
  {
    \dim_zero:N \l__tblr_d_dim
    \int_step_inline:nnn { \l__tblr_vbox_align_tl } { \c@rowcount }
      {
        \dim_add:Nn \l__tblr_d_dim { \__tblr_valign_get_row_total:n {##1} }
        \dim_add:Nn \l__tblr_d_dim
          { \__tblr_valign_get_hline_total:n { \int_eval:n { ##1 + 1 } } }
      }
    \box_set_dp:Nn #1 { \l__tblr_d_dim }
    \box_set_ht:Nn #1 { \l__tblr_t_dim - \l__tblr_d_dim }
  }

\cs_new_nopar:Npn \__tblr_valign_get_hline_total:n #1
  {
    \__tblr_spec_item:ne { hline } { [#1] / @hline-height }
  }

\cs_new_nopar:Npn \__tblr_valign_get_row_total:n #1
  {
    \__tblr_data_item:nen { row } {#1} { abovesep }
    +
    \__tblr_data_item:nen { row } {#1} { @row-height }
    +
    \__tblr_data_item:nen { row } {#1} { belowsep }
  }

\cs_new_nopar:Npn \__tblr_valign_get_row_height:n #1
  {
    \__tblr_data_item:nen { row } {#1} { abovesep }
```

```
    +
    ( \__tblr_data_item:nen { row } {#1} { @row-height }
      +
      \__tblr_data_item:nen { row } {#1} { @row-upper }
      -
      \__tblr_data_item:nen { row } {#1} { @row-lower }
    ) / 2
  }

\cs_new_nopar:Npn \__tblr_valign_get_row_depth:n #1
  {
    ( \__tblr_data_item:nen { row } {#1} { @row-height }
      -
      \__tblr_data_item:nen { row } {#1} { @row-upper }
      +
      \__tblr_data_item:nen { row } {#1} { @row-lower }
    ) / 2
    +
    \__tblr_data_item:nen { row } {#1} { belowsep }
  }
```

9.36 Build table components

```
\dim_new:N \l__tblr_col_o_wd_dim
\dim_new:N \l__tblr_col_b_wd_dim

%% Build hline. #1: row number
\cs_new_protected:Npn \__tblr_build_hline:n #1
  {
    \int_step_inline:nn { \c@colcount }
      { \__tblr_build_hline_segment:nn { #1 } { ##1 } }
  }
\cs_generate_variant:Nn \__tblr_build_hline:n { x, V }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_hline_segment:nn #1 #2
  {
    \tl_set:Ne \l__tblr_n_tl
      { \__tblr_spec_item:ne { hline } { [#1] / @hline-count } }
    \tl_set:Ne \l__tblr_o_tl
      { \__tblr_spec_item:ne { hline } { [#1][#2] / omit } }
    \__tblr_get_col_outer_width_border_width:nNN {#2}
      \l__tblr_col_o_wd_dim \l__tblr_col_b_wd_dim
    \tl_if_empty:NTF \l__tblr_o_tl
      {
        \int_compare:nNnT { \l__tblr_n_tl } > {0}
          { \__tblr_build_hline_segment_real:nn {#1} {#2} }
      }
      { \__tblr_build_hline_segment_omit:nn {#1} {#2} }
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_hline_segment_omit:nn #1 #2
  {
    \skip_horizontal:n { \l__tblr_col_o_wd_dim - \l__tblr_col_b_wd_dim }
  }
```

```
\tl_new:N \lTblrDefaultHruleColorTl

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_hline_segment_real:nn #1 #2
  {
    \tl_set:Ne \l__tblr_s_tl
      { \__tblr_prop_item:ne { inner } { rulesep } }
    \vbox_set:Nn \l__tblr_c_box
      {
        %% add an empty hbox to support vbox width
        \tex_hbox:D to \l__tblr_col_o_wd_dim {}
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \tl_set:Ne \l__tblr_h_tl
              { \__tblr_spec_item:ne { hline } { [#1](##1) / @hline-height } }
            \hrule height ~ 0pt % remove lineskip
            \hbox_set_to_wd:Nnn \l__tblr_b_box { \l__tblr_col_o_wd_dim }
              {
                \__tblr_get_hline_left_right_skips:nnn {#1} {#2} {##1}
                \skip_horizontal:N \l__tblr_hline_leftskip_dim
                \tl_set:Ne \l__tblr_f_tl
                  { \__tblr_spec_item:ne { hline } { [#1][#2](##1) / fg } }
                \tl_if_empty:NTF \l__tblr_f_tl
                  {
                    \tl_if_empty:NF \lTblrDefaultHruleColorTl
                      { \color { \lTblrDefaultHruleColorTl } }
                  }
                  { \color { \l__tblr_f_tl } }
                \__tblr_get_hline_segment_child:nnn {#1} {#2} {##1}
                \skip_horizontal:N \l__tblr_hline_rightskip_dim
              }
            \box_set_ht:Nn \l__tblr_b_box { \l__tblr_h_tl }
            \box_set_dp:Nn \l__tblr_b_box { 0pt }
            \box_use:N \l__tblr_b_box
            \skip_vertical:n { \l__tblr_s_tl }
          }
        \skip_vertical:n { - \l__tblr_s_tl }
      }
    \box_use:N \l__tblr_c_box
    \skip_horizontal:n { - \l__tblr_col_b_wd_dim }
  }

%% Read from table specifications and calculate the widths of row and border
%% column outer width = content width + colsep width + border width
%% #1: the column number, #2: outer width, #3: border width
\cs_new_protected:Npn \__tblr_get_col_outer_width_border_width:nNN #1 #2 #3
  {
    \dim_set:Nn #3
      { \__tblr_spec_item:ne { vline } { [\int_eval:n {#1 + 1}] / @vline-width } }
    \dim_set:Nn #2
      {
        \__tblr_spec_item:ne { vline } { [#1] / @vline-width }
        +
        \__tblr_data_item:nen { column } {#1} { leftsep }
        +
        \__tblr_data_item:nen { column } {#1} { @col-width }
        +
        \__tblr_data_item:nen { column } {#1} { rightsep }
```

```
        +
        #3
      }
  }

\dim_new:N \l__tblr_hline_leftskip_dim
\dim_new:N \l__tblr_hline_rightskip_dim

%% Calculate left and right skips from leftpos and rightpos specifications
%% #1: row number; #2: column number; #3: hline index;
\cs_new_protected:Npn \__tblr_get_hline_left_right_skips:nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_hline_leftpos_tl
      { \__tblr_spec_item:ne { hline } { [#1][#2](#3) / leftpos } }
    \tl_if_empty:NT \l__tblr_hline_leftpos_tl
      { \tl_set:Nn \l__tblr_hline_leftpos_tl {1} } % default position
    \tl_set:Ne \l__tblr_hline_rightpos_tl
      { \__tblr_spec_item:ne { hline } { [#1][#2](#3) / rightpos } }
    \tl_if_empty:NT \l__tblr_hline_rightpos_tl
      { \tl_set:Nn \l__tblr_hline_rightpos_tl {1} } % default position
    \fp_compare:nNnT { \l__tblr_hline_leftpos_tl } < {1}
      {
        \dim_set:Nn \l_tmpa_dim
          { \__tblr_spec_item:ne { vline } { [#2] / @vline-width } }
        \dim_set:Nn \l_tmpb_dim
          { \__tblr_data_item:nen { column } {#2} { leftsep } }
        \fp_compare:nNnTF { \l__tblr_hline_leftpos_tl } < {0}
          {
            \dim_set:Nn \l__tblr_hline_leftskip_dim
              { \l_tmpa_dim - \l__tblr_hline_leftpos_tl \l_tmpb_dim }
          }
          {
            \dim_set:Nn \l__tblr_hline_leftskip_dim
              { \l_tmpa_dim - \l__tblr_hline_leftpos_tl \l_tmpa_dim }
          }
      }
    \fp_compare:nNnT { \l__tblr_hline_rightpos_tl } < {1}
      {
        \dim_set:Nn \l_tmpa_dim
          {
            \__tblr_spec_item:ne { vline }
              { [\int_eval:n { #2 + 1 }] / @vline-width }
          }
        \dim_set:Nn \l_tmpb_dim
          { \__tblr_data_item:nen { column } {#2} { rightsep } }
        \fp_compare:nNnTF { \l__tblr_hline_rightpos_tl } < {0}
          {
            \dim_set:Nn \l__tblr_hline_rightskip_dim
              { \l_tmpa_dim - \l__tblr_hline_rightpos_tl \l_tmpb_dim }
          }
          {
            \dim_set:Nn \l__tblr_hline_rightskip_dim
              { \l_tmpa_dim - \l__tblr_hline_rightpos_tl \l_tmpa_dim }
          }
      }
  }
```

```
\dim_new:N \l__tblr_row_ht_dim
\dim_new:N \l__tblr_row_dp_dim
\dim_new:N \l__tblr_row_abovesep_dim
\dim_new:N \l__tblr_row_belowsep_dim
\box_new:N \l__tblr_row_vlines_box
\box_new:N \l__tblr_vline_box
\box_new:N \l__tblr_cell_box

%% Build current row, #1: row number
%% To fix disappeared vlines with colorful tables in Adobe Reader (see #76),
%% we collect all vlines and draw them at the end of the row.
\cs_new_protected:Npn \__tblr_build_row:N #1
  {
    \int_set:Nn \c@rownum {#1}
    \__tblr_update_rowsep_registers:
    \__tblr_get_row_inner_height_depth:VNNNN #1
      \l__tblr_row_ht_dim \l__tblr_row_dp_dim
      \l__tblr_row_abovesep_dim \l__tblr_row_belowsep_dim
    \__tblr_hook_use:n { row/before }
    \tblr_vrule_wd_ht_dp:nnn {0pt} {\l__tblr_row_ht_dim} {\l__tblr_row_dp_dim}
    \hbox_set:Nn \l__tblr_row_vlines_box
      {
        \tblr_vrule_wd_ht_dp:nnn {0pt} {\l__tblr_row_ht_dim} {\l__tblr_row_dp_dim}
      }
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \__tblr_put_one_vline:n
          { \__tblr_build_vline_segment:nn {#1} { \l__tblr_j_tl } }
        \__tblr_put_one_cell:n { \__tblr_build_cell:NN #1 \l__tblr_j_tl }
      }
    \__tblr_put_one_vline:n
      { \__tblr_build_vline_segment:nn {#1} { \int_eval:n {\c@colcount + 1} } }
    \skip_horizontal:n { - \box_wd:N \l__tblr_row_vlines_box }
    \box_use:N \l__tblr_row_vlines_box
    \__tblr_hook_use:n { row/after }
  }

%% Read from table specifications and calculate inner height/depth of the row
%% inner height = abovesep + above vspace + row upper
%% inner depth = row lower + below vspace + belowsep
%% #1: the row number; #2: resulting inner height; #3: resulting inner depth;
%% #4: restulting abovesep; #5: restulting belowsep.

\dim_new:N \l__tblr_row_upper_dim
\dim_new:N \l__tblr_row_lower_dim
\dim_new:N \l__tblr_row_vpace_dim

\cs_new_protected:Npn \__tblr_get_row_inner_height_depth:nNNNN #1 #2 #3 #4 #5
  {
    \dim_set:Nn #4
      { \__tblr_data_item:nen { row } {#1} { abovesep } }
    \dim_set:Nn #5
      { \__tblr_data_item:nen { row } {#1} { belowsep } }
    \dim_set:Nn \l__tblr_row_upper_dim
      {  \__tblr_data_item:nen { row } {#1} { @row-upper } }
    \dim_set:Nn \l__tblr_row_lower_dim
      {  \__tblr_data_item:nen { row } {#1} { @row-lower } }
```

```
    \dim_set:Nn \l__tblr_row_vpace_dim
      {
        ( \__tblr_data_item:nen { row } {#1} { @row-height }
          - \l__tblr_row_upper_dim - \l__tblr_row_lower_dim ) / 2
      }
    \dim_set:Nn #2 { #4 + \l__tblr_row_vpace_dim + \l__tblr_row_upper_dim }
    \dim_set:Nn #3 { \l__tblr_row_lower_dim + \l__tblr_row_vpace_dim + #5 }
  }
\cs_generate_variant:Nn \__tblr_get_row_inner_height_depth:nNNNN { V }

\cs_new_protected:Npn \__tblr_put_one_vline:n #1
  {
    \hbox_set:Nn \l__tblr_vline_box {#1}
    \skip_horizontal:n { \box_wd:N \l__tblr_vline_box }
    \hbox_set:Nn \l__tblr_row_vlines_box
      {
        \hbox_unpack:N \l__tblr_row_vlines_box
        \box_use:N \l__tblr_vline_box
      }
  }

\cs_new_protected:Npn \__tblr_put_one_cell:n #1
  {
    \hbox_set:Nn \l__tblr_cell_box {#1}
    \hbox_set:Nn \l__tblr_row_vlines_box
      {
        \hbox_unpack:N \l__tblr_row_vlines_box
        \skip_horizontal:n { \box_wd:N \l__tblr_cell_box }
      }
    \box_use:N \l__tblr_cell_box
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_vline_segment:nn #1 #2
  {
    \tl_set:Ne \l__tblr_n_tl
      { \__tblr_spec_item:ne { vline } { [#2] / @vline-count } }
    \tl_set:Ne \l__tblr_o_tl
      { \__tblr_spec_item:ne { vline } { [#1][#2] / omit } }
    \tl_if_empty:NTF \l__tblr_o_tl
      {
        \int_compare:nNnT { \l__tblr_n_tl } > {0}
          { \__tblr_build_vline_segment_real:nn {#1} {#2} }
      }
      { \__tblr_build_vline_segment_omit:nn {#1} {#2} }
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_vline_segment_omit:nn #1 #2
  {
    \tl_set:Ne \l__tblr_w_tl
      { \__tblr_spec_item:ne { vline } { [#2] / @vline-width } }
    \skip_horizontal:N \l__tblr_w_tl
  }

\tl_new:N \lTblrDefaultVruleColorTl
```

```
%% #1: row number, #2: column number
%% We make every vline segment intersect with first hline below
%% to remove gaps in vlines around multirow cells
\cs_new_protected:Npn \__tblr_build_vline_segment_real:nn #1 #2
  {
    \tl_set:Ne \l__tblr_s_tl
      { \__tblr_prop_item:ne { inner } { rulesep } }
    \hbox_set:Nn \l__tblr_a_box
      {
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \tl_set:Ne \l__tblr_w_tl
              { \__tblr_spec_item:ne { vline } { [#2](##1) / @vline-width } }
            \vbox_set_to_ht:Nnn \l__tblr_b_box
              { \dim_eval:n { \l__tblr_row_ht_dim + \l__tblr_row_dp_dim } }
              {
                \tl_set:Ne \l__tblr_f_tl
                  { \__tblr_spec_item:ne { vline } { [#1][#2](##1) / fg } }
                \tl_if_empty:NTF \l__tblr_f_tl
                  {
                    \tl_if_empty:NF \lTblrDefaultVruleColorTl
                      { \color { \lTblrDefaultVruleColorTl } }
                  }
                  { \color { \l__tblr_f_tl } }
                \__tblr_get_vline_above_below_skips:nnn {#1} {#2} {##1}
                \skip_vertical:N \l__tblr_vline_aboveskip_dim
                \__tblr_get_vline_segment_child:nnnee {#1} {#2} {##1}
                  { \dim_eval:n { \l__tblr_row_ht_dim } }
                  { \dim_eval:n { \l__tblr_row_dp_dim } }
                \skip_vertical:N \l__tblr_vline_belowskip_dim
              }
            \box_set_wd:Nn \l__tblr_b_box { \l__tblr_w_tl }
            \box_use:N \l__tblr_b_box
            \skip_horizontal:n { \l__tblr_s_tl }
          }
        \skip_horizontal:n { - \l__tblr_s_tl }
      }
    \vbox_set:Nn \l__tblr_c_box { \box_use:N \l__tblr_a_box }
    \box_set_ht:Nn \l__tblr_c_box { \dim_use:N \l__tblr_row_ht_dim }
    \box_set_dp:Nn \l__tblr_c_box { \dim_use:N \l__tblr_row_dp_dim }
    \box_use:N \l__tblr_c_box
  }

\dim_new:N \l__tblr_vline_aboveskip_dim
\dim_new:N \l__tblr_vline_belowskip_dim

%% Calculate above and below skips from abovepos and belowpos specifications
%% #1: row number; #2: column number; #3: vline index;
\cs_new_protected:Npn \__tblr_get_vline_above_below_skips:nnn #1 #2 #3
  {
    \tl_set:Ne \l__tblr_vline_abovepos_tl
      { \__tblr_spec_item:ne { vline } { [#1][#2](#3) / abovepos } }
    \tl_if_empty:NT \l__tblr_vline_abovepos_tl
      {
        \tl_set:Nn \l__tblr_vline_abovepos_tl {0} % default position
      }
    \fp_compare:nNnF { \l__tblr_vline_abovepos_tl } = {0}
      {
```

```
        \dim_set:Nn \l_tmpa_dim
          { \__tblr_spec_item:ne { hline } { [#1] / @hline-height } }
        \fp_compare:nNnTF { \l__tblr_vline_abovepos_tl } < {0}
          {
            \dim_set:Nn \l__tblr_vline_aboveskip_dim
              { - \l__tblr_vline_abovepos_tl \l__tblr_row_abovesep_dim }
          }
          {
            \dim_set:Nn \l__tblr_vline_aboveskip_dim
              { - \l__tblr_vline_abovepos_tl \l_tmpa_dim }
          }
      }
    %% To join two vline segment above and below a cline,
    %% we choose to extend every vline downwards a little (#55, #272).
    \tl_set:Ne \l__tblr_vline_belowpos_tl
      { \__tblr_spec_item:ne { vline } { [#1][#2](#3) / belowpos } }
    \tl_if_empty:NTF \l__tblr_vline_belowpos_tl
      {
        \dim_set:Nn \l__tblr_vline_belowskip_dim
          {
            - \__tblr_spec_item:ne { hline }
                { [\int_eval:n { #1 + 1 }](1) / @hline-height }
            + 0pt
          }
      }
      {
        \dim_set:Nn \l_tmpa_dim
          {
            \__tblr_spec_item:ne { hline }
              { [\int_eval:n { #1 + 1 }] / @hline-height }
          }
        \fp_compare:nNnTF { \l__tblr_vline_belowpos_tl } < {0}
          {
            \dim_set:Nn \l__tblr_vline_belowskip_dim
              { - \l__tblr_vline_belowpos_tl \l__tblr_row_belowsep_dim }
          }
          {
            \dim_set:Nn \l__tblr_vline_belowskip_dim
              { - \l__tblr_vline_belowpos_tl \l_tmpa_dim }
          }
      }
  }

%% These public variables are updated by default before building a cell
\int_new:N \lTblrCellRowSpanInt
\int_new:N \lTblrCellColSpanInt
\tl_new:N \lTblrCellBackgroundTl
\bool_new:N \lTblrCellOmittedBool

\dim_new:N \l__tblr_cell_wd_dim
\dim_new:N \l__tblr_cell_ht_dim

\cs_new_protected:Npn \__tblr_build_cell:NN #1 #2
  {
    \int_set:Nn \c@colnum {#2}
    \__tblr_update_colsep_registers:
    \group_begin:
```

```
    \tl_set:Ne \l__tblr_w_tl
      { \__tblr_data_item:nen { column } {#2} { @col-width } }
    \tl_set:Ne \l__tblr_h_tl
      { \__tblr_data_item:nen { row } {#1} { @row-height } }
    \tl_set:Ne \l__tblr_x_tl
      { \__tblr_data_item:nen { column } {#2} { leftsep} }
    \tl_set:Ne \l__tblr_y_tl
      { \__tblr_data_item:nen { column } {#2} { rightsep } }
    \int_set:Nn \lTblrCellColSpanInt
      { \__tblr_data_item:neen { cell } {#1} {#2} { colspan } }
    \int_compare:nNnTF { \lTblrCellColSpanInt } < {2}
      { \dim_set:Nn \l__tblr_cell_wd_dim { \l__tblr_w_tl } }
      {
        \__tblr_get_span_horizontal_sizes:NNNNN #1 #2
          \l__tblr_o_dim \l__tblr_cell_wd_dim \l__tblr_q_dim
      }
    \int_set:Nn \lTblrCellRowSpanInt
      { \__tblr_data_item:neen { cell } {#1} {#2} { rowspan } }
    \int_compare:nNnTF { \lTblrCellRowSpanInt } < {2}
      { \dim_set:Nn \l__tblr_cell_ht_dim { \l__tblr_h_tl } }
      {
        \__tblr_get_span_vertical_sizes:NNNNN #1 #2
          \l__tblr_r_dim \l__tblr_cell_ht_dim \l__tblr_t_dim
      }
    \__tblr_get_cell_alignments:nn {#1} {#2}
    \__tblr_build_cell_background:NN #1 #2
    \__tblr_build_cell_content:NN #1 #2
    \group_end:
  }

%% These public variables are updated by html library before building a cell
\tl_new:N \lTblrCellAboveBorderStyleTl
\dim_new:N \lTblrCellAboveBorderWidthDim
\tl_new:N \lTblrCellAboveBorderColorTl
\tl_new:N \lTblrCellBelowBorderStyleTl
\dim_new:N \lTblrCellBelowBorderWidthDim
\tl_new:N \lTblrCellBelowBorderColorTl
\tl_new:N \lTblrCellLeftBorderStyleTl
\dim_new:N \lTblrCellLeftBorderWidthDim
\tl_new:N \lTblrCellLeftBorderColorTl
\tl_new:N \lTblrCellRightBorderStyleTl
\dim_new:N \lTblrCellRightBorderWidthDim
\tl_new:N \lTblrCellRightBorderColorTl

%% #1: row number in tl; #2: column number in tl
%% This function is called only when html library is loaded.
%% The properties can be used by tagpdf, tex4ht and lwarp packages
\cs_new_protected:Npn \__tblr_expose_cell_properties:NN #1 #2
  {
    \__tblr_expose_cell_border:NNnn #1 #2 { hline } { Above }
    \tl_set:Ne \l_tmpa_tl { \int_eval:n { #1 + \lTblrCellRowSpanInt } }
    \__tblr_expose_cell_border:NNnn \l_tmpa_tl #2 { hline } { Below }
    \__tblr_expose_cell_border:NNnn #1 #2 { vline } { Left }
    \tl_set:Ne \l_tmpb_tl { \int_eval:n { #2 + \lTblrCellColSpanInt } }
    \__tblr_expose_cell_border:NNnn #1 \l_tmpb_tl { vline } { Right }
  }
```

```
\tl_new:N \l__tblr_dash_value_tl
\tl_new:N \l__tblr_width_value_tl
\tl_new:N \l__tblr_color_value_tl

%% #1: row number in tl; #2: column number in tl;
%% #3: hline or vline; #4: position of border (Above/Below/Left/Right).
\cs_new_protected:Npn \__tblr_expose_cell_border:NNnn #1 #2 #3 #4
  {
    %% get border style
    \tl_set:Ne \l__tblr_dash_value_tl %% may be empty
      { \__tblr_spec_item:ne { #3 } { [#1][#2](1) / @dash } }
    \tl_if_head_eq_meaning:VNTF \l__tblr_dash_value_tl \q__tblr_dash
      {
        \tl_set:ce { lTblrCell #4 BorderStyleTl }
          { \tl_tail:N \l__tblr_dash_value_tl }
        %% get border width
        \tl_set:Ne \l__tblr_width_value_tl
          { \__tblr_spec_item:ne { #3 } { [#1][#2](1) / wd } }
        \tl_if_empty:NTF \l__tblr_width_value_tl
          { \dim_set:cn { lTblrCell #4 BorderWidthDim } { 0.4pt } }
          {
            \dim_set:cn { lTblrCell #4 BorderWidthDim }
              { \l__tblr_width_value_tl }
          }
        %% get border color
        \tl_set:ce { lTblrCell #4 BorderColorTl }
          { \__tblr_spec_item:ne { #3 } { [#1][#2](1) / fg } }
      }
      {
        \tl_clear:c { lTblrCell #4 BorderStyleTl }
        \dim_set:cn { lTblrCell #4 BorderWidthDim } { 0pt }
        \tl_clear:c { lTblrCell #4 BorderColorTl }
      }
  }

\cs_new_protected:Npn \__tblr_build_cell_content:NN #1 #2
  {
    \bool_if:NT \l__tblr_html_variables_bool
      { \__tblr_expose_cell_properties:NN #1 #2 }
    \__tblr_hook_use:n { cell/before }
    \hbox_set_to_wd:Nnn \l__tblr_a_box { \l__tblr_cell_wd_dim }
      {
        \tl_if_eq:NnTF \g__tblr_cell_halign_tl {j}
          % cell width may be less than column width for j cells
          { \__tblr_get_cell_text:nn {#1} {#2} \hfil }
          {
            \tl_if_eq:NnF \g__tblr_cell_halign_tl {l} { \hfil }
            \__tblr_get_cell_text:nn {#1} {#2}
            \tl_if_eq:NnF \g__tblr_cell_halign_tl {r} { \hfil }
          }
      }
    \vbox_set_to_ht:Nnn \l__tblr_b_box { \l__tblr_cell_ht_dim }
      {
        \tl_case:Nn \g__tblr_cell_valign_tl
          {
            \c__tblr_valign_m_tl
              {
                \vfil
```

```
                \int_compare:nNnT { \lTblrCellRowSpanInt } < {2}
                  {
                    \box_set_ht:Nn \l__tblr_a_box
                      { \__tblr_data_item:nen { row } {#1} { @row-upper } }
                    \box_set_dp:Nn \l__tblr_a_box
                      { \__tblr_data_item:nen { row } {#1} { @row-lower } }
                  }
                \box_use:N \l__tblr_a_box
                \vfil
              }
            \c__tblr_valign_h_tl
              {
                \box_set_ht:Nn \l__tblr_a_box
                  { \__tblr_data_item:nen { row } {#1} { @row-head } }
                \box_use:N \l__tblr_a_box
                \vfil
              }
            \c__tblr_valign_f_tl
              {
                \vfil
                \int_compare:nNnTF { \lTblrCellRowSpanInt } < {2}
                  {
                    \box_set_dp:Nn \l__tblr_a_box
                      { \__tblr_data_item:nen { row } {#1} { @row-foot } }
                  }
                  {
                    \box_set_dp:Nn \l__tblr_a_box
                      {
                        \__tblr_data_item:nen
                          { row }
                          { \int_eval:n { #1 + \lTblrCellRowSpanInt - 1 } }
                          { @row-foot }
                      }
                  }
                \box_use:N \l__tblr_a_box
              }
          }
        \hrule height ~ 0pt %% zero depth
      }
    \vbox_set_to_ht:Nnn \l__tblr_c_box
      { \l__tblr_row_ht_dim - \l__tblr_row_abovesep_dim }
      {
        \box_use:N \l__tblr_b_box
        \vss
      }
    \skip_horizontal:n { \l__tblr_x_tl }
    \box_use:N \l__tblr_c_box
    \skip_horizontal:n { \l__tblr_y_tl - \l__tblr_cell_wd_dim + \l__tblr_w_tl }
    \__tblr_hook_use:n { cell/after }
  }

\cs_new_protected:Npn \__tblr_build_cell_background:NN #1 #2
  {
    \bool_set:Nn \lTblrCellOmittedBool
      {
        \int_compare_p:nNn
          { \__tblr_data_item:neen { cell } {#1} {#2} { omit } } = {1}
      }
```

```
    \bool_if:NF \lTblrCellOmittedBool
      {
        \tl_set:Ne \lTblrCellBackgroundTl
          { \__tblr_data_item:neen { cell } {#1} {#2} { background } }
        \group_begin:
        \tl_if_empty:NF \lTblrCellBackgroundTl
          {
            \__tblr_get_cell_background_width:NNN #1 #2 \l_tmpa_dim
            \__tblr_get_cell_background_depth:NNN #1 #2 \l_tmpb_dim
            \__tblr_build_cell_background:nnnn
              { \dim_use:N \l_tmpa_dim }
              { \l__tblr_row_ht_dim }
              { \dim_use:N \l_tmpb_dim }
              { \lTblrCellBackgroundTl }
          }
        \group_end:
      }
  }

%% #1: row number; #2: column number; #3 resulting dimension
\cs_new_protected:Npn \__tblr_get_cell_background_width:NNN #1 #2 #3
  {
    \int_compare:nNnTF { \lTblrCellColSpanInt } < {2}
      { \dim_set:Nn #3 { \l__tblr_x_tl + \l__tblr_w_tl + \l__tblr_y_tl } }
      {
        \dim_set:Nn #3 { \l__tblr_o_dim + \l__tblr_cell_wd_dim + \l__tblr_q_dim }
      }
  }

%% #1: row number; #2: column number; #3 resulting dimension
\cs_new_protected:Npn \__tblr_get_cell_background_depth:NNN #1 #2 #3
  {
    \int_compare:nNnTF { \lTblrCellRowSpanInt } < {2}
      { \dim_set_eq:NN #3 \l__tblr_row_dp_dim }
      {
        \dim_set:Nn #3
          {
            \l__tblr_r_dim + \l__tblr_cell_ht_dim
                           + \l__tblr_t_dim - \l__tblr_row_ht_dim
          }
      }
  }

%% #1: width, #2: height, #3: depth, #4: color
\cs_new_protected:Npn \__tblr_build_cell_background:nnnn #1 #2 #3 #4
  {
    \hbox_set:Nn \l__tblr_a_box
      {
        \color {#4}
        \vrule width ~ #1 ~ height ~ #2 ~ depth ~ #3
      }
    \box_set_dp:Nn \l__tblr_a_box { 0pt }
    \box_use:N \l__tblr_a_box
    \skip_horizontal:n { - #1 }
  }

%% #1: row number; #2: column number; #3: dimen register for rowsep above.
```

```
%% #4: dimen register for total height; #5: dimen register for rowsep below.
%% We can use \l__tblr_row_item_skip_size_prop which was made before
%% But when vspan=even, there are no itemskip in the prop list.
%% Therefore we need to calculate them from the sizes of items and skips
\cs_new_protected:Npn \__tblr_get_span_vertical_sizes:NNNNN #1 #2 #3 #4 #5
  {
    \dim_set:Nn #3
      { \__tblr_data_item:nen { row } {#1} { abovesep } }
    \dim_zero:N #4
    \dim_add:Nn #4
      { \prop_item:Ne \l__tblr_row_item_skip_size_prop { item[#1] } }
    \int_step_inline:nnn { #1 + 1 } { #1 + \lTblrCellRowSpanInt - 1 }
      {
        \dim_add:Nn #4
          {
            \prop_item:Ne \l__tblr_row_item_skip_size_prop { skip[##1] }
            +
            \prop_item:Ne \l__tblr_row_item_skip_size_prop { item[##1] }
          }
      }
    \dim_set:Nn #5
      {
        \__tblr_data_item:nen { row }
          { \int_eval:n { #1 + \lTblrCellRowSpanInt - 1 } } { belowsep }
      }
    %\tl_log:e { cell[#1][#2] ~:~ \dim_use:N #3, \dim_use:N #4, \dim_use:N #5 }
  }


%% #1: row number; #2: column number; #3: dimen register for colsep left.
%% #4: dimen register for total width; #5: dimen register for colsep right.
%% We can use \l__tblr_col_item_skip_size_prop which was made before
%% But when hspan=even or hspan=minimal, there are no itemskip in the prop list.
%% Therefore we need to calculate them from the sizes of items and skips
\cs_new_protected:Npn \__tblr_get_span_horizontal_sizes:NNNNN #1 #2 #3 #4 #5
  {
    \dim_set:Nn #3
      { \__tblr_data_item:nen { column } {#2} { leftsep } }
    \dim_zero:N #4
    \dim_add:Nn #4
      { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[#2] } }
    \int_step_inline:nnn { #2 + 1 } { #2 + \lTblrCellColSpanInt - 1 }
      {
        \dim_add:Nn #4
          {
            \prop_item:Ne \l__tblr_col_item_skip_size_prop { skip[##1] }
            +
            \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[##1] }
          }
      }
    \dim_set:Nn #5
      {
        \__tblr_data_item:nen { column }
          { \int_eval:n {#2 + \lTblrCellColSpanInt - 1} } { rightsep }
      }
    %\tl_log:e { cell[#1][#2] ~:~ \dim_use:N #3, \dim_use:N #4, \dim_use:N #5 }
  }
```

9.37 Tracing tabularray

```
\NewDocumentCommand \SetTblrTracing { m }
  {
    \__tblr_keys_set:nn { tracing/set } {#1}
  }


\bool_new:N \g__tblr_tracing_text_bool
\bool_new:N \g__tblr_tracing_command_bool
\bool_new:N \g__tblr_tracing_option_bool
\bool_new:N \g__tblr_tracing_theme_bool
\bool_new:N \g__tblr_tracing_outer_bool
\bool_new:N \g__tblr_tracing_inner_bool
\bool_new:N \g__tblr_tracing_column_bool
\bool_new:N \g__tblr_tracing_row_bool
\bool_new:N \g__tblr_tracing_cell_bool
\bool_new:N \g__tblr_tracing_vline_bool
\bool_new:N \g__tblr_tracing_hline_bool
\bool_new:N \g__tblr_tracing_colspec_bool
\bool_new:N \g__tblr_tracing_rowspec_bool
\bool_new:N \g__tblr_tracing_target_bool
\bool_new:N \g__tblr_tracing_cellspan_bool
\bool_new:N \g__tblr_tracing_intarray_bool
\bool_new:N \g__tblr_tracing_page_bool
\bool_new:N \g__tblr_tracing_step_bool


\__tblr_keys_define:nn { tracing/set }
  {
    +text .code:n = \bool_gset_true:N \g__tblr_tracing_text_bool,
    -text .code:n = \bool_gset_false:N \g__tblr_tracing_text_bool,
    +command .code:n = \bool_gset_true:N \g__tblr_tracing_command_bool,
    -command .code:n = \bool_gset_false:N \g__tblr_tracing_command_bool,
    +option .code:n = \bool_gset_true:N \g__tblr_tracing_option_bool,
    -option .code:n = \bool_gset_false:N \g__tblr_tracing_option_bool,
    +theme .code:n = \bool_gset_true:N \g__tblr_tracing_theme_bool,
    -theme .code:n = \bool_gset_false:N \g__tblr_tracing_theme_bool,
    +outer .code:n = \bool_gset_true:N \g__tblr_tracing_outer_bool,
    -outer .code:n = \bool_gset_false:N \g__tblr_tracing_outer_bool,
    +inner .code:n = \bool_gset_true:N \g__tblr_tracing_inner_bool,
    -inner .code:n = \bool_gset_false:N \g__tblr_tracing_inner_bool,
    +column .code:n = \bool_gset_true:N \g__tblr_tracing_column_bool,
    -column .code:n = \bool_gset_false:N \g__tblr_tracing_column_bool,
    +row .code:n = \bool_gset_true:N \g__tblr_tracing_row_bool,
    -row .code:n = \bool_gset_false:N \g__tblr_tracing_row_bool,
    +cell .code:n = \bool_gset_true:N \g__tblr_tracing_cell_bool,
    -cell .code:n = \bool_gset_false:N \g__tblr_tracing_cell_bool,
    +vline .code:n = \bool_gset_true:N \g__tblr_tracing_vline_bool,
    -vline .code:n = \bool_gset_false:N \g__tblr_tracing_vline_bool,
    +hline .code:n = \bool_gset_true:N \g__tblr_tracing_hline_bool,
    -hline .code:n = \bool_gset_false:N \g__tblr_tracing_hline_bool,
    +colspec .code:n = \bool_gset_true:N \g__tblr_tracing_colspec_bool,
    -colspec .code:n = \bool_gset_false:N \g__tblr_tracing_colspec_bool,
    +rowspec .code:n = \bool_gset_true:N \g__tblr_tracing_rowspec_bool,
    -rowspec .code:n = \bool_gset_false:N \g__tblr_tracing_rowspec_bool,
    +target .code:n = \bool_gset_true:N \g__tblr_tracing_target_bool,
    -target .code:n = \bool_gset_false:N \g__tblr_tracing_target_bool,
    +cellspan .code:n = \bool_gset_true:N \g__tblr_tracing_cellspan_bool,
    -cellspan .code:n = \bool_gset_false:N \g__tblr_tracing_cellspan_bool,
```

```
    +intarray .code:n = \bool_gset_true:N \g__tblr_tracing_intarray_bool,
    -intarray .code:n = \bool_gset_false:N \g__tblr_tracing_intarray_bool,
    +page .code:n = \bool_gset_true:N \g__tblr_tracing_page_bool,
    -page .code:n = \bool_gset_false:N \g__tblr_tracing_page_bool,
    +step .code:n = \bool_gset_true:N \g__tblr_tracing_step_bool,
    -step .code:n = \bool_gset_false:N \g__tblr_tracing_step_bool,
    all .code:n = \__tblr_enable_all_tracings:,
    none .code:n = \__tblr_disable_all_tracings:,
  }

\cs_new_protected_nopar:Npn \__tblr_enable_all_tracings:
  {
    \bool_gset_true:N \g__tblr_tracing_text_bool
    \bool_gset_true:N \g__tblr_tracing_command_bool
    \bool_gset_true:N \g__tblr_tracing_option_bool
    \bool_gset_true:N \g__tblr_tracing_theme_bool
    \bool_gset_true:N \g__tblr_tracing_outer_bool
    \bool_gset_true:N \g__tblr_tracing_inner_bool
    \bool_gset_true:N \g__tblr_tracing_column_bool
    \bool_gset_true:N \g__tblr_tracing_row_bool
    \bool_gset_true:N \g__tblr_tracing_cell_bool
    \bool_gset_true:N \g__tblr_tracing_vline_bool
    \bool_gset_true:N \g__tblr_tracing_hline_bool
    \bool_gset_true:N \g__tblr_tracing_colspec_bool
    \bool_gset_true:N \g__tblr_tracing_rowspec_bool
    \bool_gset_true:N \g__tblr_tracing_target_bool
    \bool_gset_true:N \g__tblr_tracing_cellspan_bool
    \bool_gset_true:N \g__tblr_tracing_intarray_bool
    \bool_gset_true:N \g__tblr_tracing_page_bool
    \bool_gset_true:N \g__tblr_tracing_step_bool
  }

\cs_new_protected_nopar:Npn \__tblr_disable_all_tracings:
  {
    \bool_gset_false:N \g__tblr_tracing_text_bool
    \bool_gset_false:N \g__tblr_tracing_command_bool
    \bool_gset_false:N \g__tblr_tracing_option_bool
    \bool_gset_false:N \g__tblr_tracing_theme_bool
    \bool_gset_false:N \g__tblr_tracing_outer_bool
    \bool_gset_false:N \g__tblr_tracing_inner_bool
    \bool_gset_false:N \g__tblr_tracing_column_bool
    \bool_gset_false:N \g__tblr_tracing_row_bool
    \bool_gset_false:N \g__tblr_tracing_cell_bool
    \bool_gset_false:N \g__tblr_tracing_vline_bool
    \bool_gset_false:N \g__tblr_tracing_hline_bool
    \bool_gset_false:N \g__tblr_tracing_colspec_bool
    \bool_gset_false:N \g__tblr_tracing_rowspec_bool
    \bool_gset_false:N \g__tblr_tracing_target_bool
    \bool_gset_false:N \g__tblr_tracing_cellspan_bool
    \bool_gset_false:N \g__tblr_tracing_intarray_bool
    \bool_gset_false:N \g__tblr_tracing_page_bool
    \bool_gset_false:N \g__tblr_tracing_step_bool
  }

\NewDocumentCommand \LogTblrTracing { m }
  {
    \__tblr_keys_set:nn { tracing/log } {#1}
```

```
  }

\__tblr_keys_define:nn { tracing/log }
  {
    step .code:n = \__tblr_log_tracing_step:n {#1},
    unknown .code:n = \__tblr_log_tracing:N \l_keys_key_str
  }

\cs_new_protected:Npn \__tblr_log_tracing:N #1
  {
    \bool_if:cT { g__tblr_tracing_ #1 _bool }
      { \cs:w __tblr_log_tracing _ #1 : \cs_end: }
  }

\cs_new_protected:Npn \__tblr_log_tracing_text:
  {
    \__tblr_spec_log:n { text }
  }

\cs_new_protected:Npn \__tblr_log_tracing_command:
  {
    \__tblr_prop_log:n { command }
  }

\cs_new_protected:Npn \__tblr_log_tracing_option:
  {
    \__tblr_prop_log:n { note }
    \__tblr_prop_log:n { remark }
    \__tblr_prop_log:n { more }
  }

\cs_new_protected:Npn \__tblr_log_tracing_theme:
  {
    \__tblr_style_log:
  }

\cs_new_protected:Npn \__tblr_log_tracing_outer:
  {
    \__tblr_spec_log:n { outer }
  }

\cs_new_protected:Npn \__tblr_log_tracing_inner:
  {
    \__tblr_prop_log:n { inner }
  }

\cs_new_protected:Npn \__tblr_log_tracing_column:
  {
    \__tblr_data_log:n { column }
  }

\cs_new_protected:Npn \__tblr_log_tracing_row:
  {
    \__tblr_data_log:n { row }
  }
```

```
\cs_new_protected:Npn \__tblr_log_tracing_cell:
  {
    \__tblr_data_log:n { cell }
  }

\cs_new_protected:Npn \__tblr_log_tracing_vline:
  {
    \__tblr_spec_log:n { vline }
  }

\cs_new_protected:Npn \__tblr_log_tracing_hline:
  {
    \__tblr_spec_log:n { hline }
  }

\cs_new_protected:Npn \__tblr_log_tracing_colspec:
  {
    \tl_if_eq:NnT \g__tblr_column_or_row_tl { Column }
      { \tl_log:N \g__tblr_expanded_colrow_spec_tl }
  }

\cs_new_protected:Npn \__tblr_log_tracing_rowspec:
  {
    \tl_if_eq:NnT \g__tblr_column_or_row_tl { Row }
      { \tl_log:N \g__tblr_expanded_colrow_spec_tl }
  }

\cs_new_protected:Npn \__tblr_log_tracing_target:
  {
    \dim_log:N \l__tblr_column_target_dim
    \prop_log:N \l__tblr_column_coefficient_prop
    \prop_log:N \l__tblr_column_natural_width_prop
    \prop_log:N \l__tblr_column_computed_width_prop
  }

\cs_new_protected:Npn \__tblr_log_tracing_cellspan:
  {
    \prop_log:N \l__tblr_col_item_skip_size_prop
    \prop_log:N \l__tblr_col_span_size_prop
    \prop_log:N \l__tblr_row_item_skip_size_prop
    \prop_log:N \l__tblr_row_span_size_prop
    \prop_log:N \l__tblr_row_span_to_row_prop
  }

\cs_new_protected:Npn \__tblr_log_tracing_page:
  {
    \tl_log:e
      {
        goal = \dim_use:N \pagegoal ,~ total = \dim_use:N \pagetotal,~
        stretch = \dim_use:N \pagestretch,~ shrink = \dim_use:N \pageshrink
      }
  }

\cs_new_protected:Npn \__tblr_log_tracing_step:n #1
  {
    \bool_if:NT \g__tblr_tracing_step_bool { \tl_log:e {Step :~ #1} }
```

```
  }

\cs_new_protected:Npn \__tblr_do_if_tracing:nn #1 #2
  {
    \bool_if:cT { g__tblr_tracing_ #1 _bool } {#2}
  }
```

9.38 Tabularray libraries

```
%% \NewTblrLibrary and \UseTblrLibrary commands

\NewDocumentCommand \NewTblrLibrary { m m }
  {
    \cs_new_protected:cpn { __tblr_use_lib_ #1: } {#2}
  }

%% Note that \prg_do_nothing: is an existing command.
\NewDocumentCommand \UseTblrLibrary { m }
  {
    \clist_map_inline:nn {#1}
      {
        \cs_if_exist:cTF { __tblr_use_lib_ ##1: }
          {
            \use:c { __tblr_use_lib_ ##1: }
            \cs_gset_eq:cN { __tblr_use_lib_ ##1: } \prg_do_nothing:
          }
          {
            \RequirePackage { tblrlib##1 }
          }
      }
  }

\prg_set_conditional:Npnn \__tblr_lib_if_used:n #1 { p, T, F, TF }
  {
    \tl_if_eq:cNTF { __tblr_use_lib_ #1: } \prg_do_nothing:
      { \prg_return_true: } { \prg_return_false: }
  }

%% Library amsmath and environments +array, +matrix, +cases, ...

\NewTblrLibrary { amsmath }
  {
    \RequirePackage { amsmath }
    \NewTblrEnviron { +array }
    \SetTblrInner[+array]{colsep = 5pt}
    \NewDocumentEnvironment { +matrix } { O{} +b } {
      \begin{+array}{
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        cells = {c}, ##1
      }
        ##2
      \end{+array}
    } { }
    \NewDocumentEnvironment { +bmatrix } { O{} +b } {
      \begin{+array}{
```

```
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        cells = {c}, delimiter = {left = [, right = ]}, ##1
      }
        ##2
      \end{+array}
    } { }
    \NewDocumentEnvironment { +Bmatrix } { O{} +b } {
      \begin{+array} {
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        cells = {c}, delimiter = {left = \lbrace, right = \rbrace}, ##1
      }
        ##2
      \end{+array}
    } { }
    \NewDocumentEnvironment { +pmatrix } { O{} +b } {
      \begin{+array} {
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        cells = {c}, delimiter = {left = (, right = )}, ##1
      }
        ##2
      \end{+array}
    } { }
    \NewDocumentEnvironment { +vmatrix } { O{} +b } {
      \begin{+array} {
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        cells = {c}, delimiter = {left = \lvert, right = \rvert}, ##1
      }
        ##2
      \end{+array}
    } { }
    \NewDocumentEnvironment { +Vmatrix } { O{} +b } {
      \begin{+array} {
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        cells = {c}, delimiter = {left = \lVert, right = \rVert}, ##1
      }
        ##2
      \end{+array}
    } { }
    \NewDocumentEnvironment { +cases } { O{} +b } {
      \begin{+array} {
        column{1} = {leftsep = 0pt}, column{Z} = {rightsep = 0pt},
        colspec = {ll}, stretch = 1.2, delimiter = {left=\lbrace, right=.}, ##1
      }
        ##2
      \end{+array}
    } { }
  }

%% Library booktabs and commands \toprule, \midrule, \bottomrule

\NewTblrLibrary { booktabs }
  {
    % We only use dimensions \aboverulesep and \belowrulesep in booktabs package
    \RequirePackage { booktabs }
    \newcommand \tblr@booktabs@hline [1] [] { \hline [##1] }
    \newcommand \tblr@booktabs@oldhline [1] [] {
      \hline [##1]
      \hborder { abovespace = \aboverulesep, belowspace = \belowrulesep }
```

```
    }
    \newcommand \tblr@booktabs@cline [2] [] { \cline [##1] {##2} }
    \newcommand \tblr@booktabs@oldcline [2] [] {
      \cline [##1] {##2}
      \hborder { abovespace = \aboverulesep, belowspace = \belowrulesep }
    }
    \newcommand \tblr@booktabs@cline@more [2] [] { \SetHline [+] {##2} {##1} }
    \newcommand \tblr@booktabs@oldcline@more [2] [] {
      \SetHline [+] {##2} {##1}
      \hborder { abovespace = \aboverulesep, belowspace = \belowrulesep }
    }
    \NewTblrTableCommand \toprule [1] [] {
      \tblr@booktabs@hline [wd=\heavyrulewidth, ##1]
    }
    \NewTblrTableCommand \midrule [1] [] {
      \tblr@booktabs@hline [wd=\lightrulewidth, ##1]
    }
    \NewTblrTableCommand \bottomrule [1] [] {
      \tblr@booktabs@hline [wd=\heavyrulewidth, ##1]
    }
    \NewTblrTableCommand \cmidrule [2] [] {
      \tblr@booktabs@cline [wd=\cmidrulewidth, endpos, ##1] {##2}
    }
    \NewTblrTableCommand \cmidrulemore [2] [] {
      \tblr@booktabs@cline@more [wd=\cmidrulewidth, endpos, ##1] {##2}
    }
    \newcommand \tblr@booktabs@change@more [1] { \cmidrulemore }
    \NewTblrTableCommand \morecmidrules {
      \peek_meaning:NTF \cmidrule { \tblr@booktabs@change@more } { \relax }
    }
    \NewTblrEnviron { booktabs }
    \NewTblrEnviron { longtabs }
    \NewTblrEnviron { talltabs }
    \SetTblrInner [ booktabs ] { rowsep = 0pt }
    \SetTblrInner [ longtabs ] { rowsep = 0pt }
    \SetTblrInner [ talltabs ] { rowsep = 0pt }
    \SetTblrOuter [ longtabs ] { long }
    \SetTblrOuter [ talltabs ] { tall }
    \RequirePackage { etoolbox }
    \newcommand \tblr@booktabs@begin@hook
      {
        \let \tblr@booktabs@hline = \tblr@booktabs@oldhline
        \let \tblr@booktabs@cline = \tblr@booktabs@oldcline
        \let \tblr@booktabs@cline@more = \tblr@booktabs@oldcline@more
      }
    \AtBeginEnvironment { booktabs } { \tblr@booktabs@begin@hook }
    \AtBeginEnvironment { longtabs } { \tblr@booktabs@begin@hook }
    \AtBeginEnvironment { talltabs } { \tblr@booktabs@begin@hook }
    \NewTblrTableCommand \specialrule [3]
      { \hline [##1] \hborder { abovespace = ##2, belowspace = ##3 } }
    \NewTblrTableCommand \addrowspace [1] [\defaultaddspace]
      { \hborder { abovespace+ = (##1) / 2, belowspace+ = (##1) / 2 } }
    \NewTblrTableCommand \addlinespace [1] [\defaultaddspace]
      { \hborder { abovespace+ = (##1) / 2, belowspace+ = (##1) / 2 } }
  }

%% Library counter for resetting all counters
```

```
\tl_new:N \l__tblr_saved_trial_counters_tl
\tl_new:N \l__tblr_saved_cell_counters_tl

\cs_new_protected:Npn \__tblr_save_counters:n #1 { }
\cs_new_protected:Npn \__tblr_restore_counters:n #1 { }

%% We use code from tabularx package for resetting all LaTeX counters,
%% where internal macro \cl@@ckpt looks like the following:
%% \@elt{page} \@elt{equation} \@elt{enumi} \@elt{enumii} \@elt{enumiii} ...

\NewTblrLibrary { counter }
  {
    \cs_set_protected:Npn \__tblr_save_counters:n ##1
      {
        \def \@elt ####1 { \global\value{####1} = \the\value{####1} \relax }
        \tl_set:ce { l__tblr_saved_ ##1 _counters_tl } { \cl@@ckpt }
        \let \@elt = \relax
      }
    \cs_set_protected:Npn \__tblr_restore_counters:n ##1
      {
        \tl_use:c { l__tblr_saved_ ##1 _counters_tl }
      }
  }

%% Library diagbox and command \diagbox

\NewTblrLibrary { diagbox }
  {
    \RequirePackage{ diagbox }
    \cs_set_eq:NN \__tblr_lib_saved_diagbox:w \diagbox
    \NewTblrContentCommand \diagbox [3] []
      {
        \__tblr_lib_diagbox_fix:n
          {
            \__tblr_lib_saved_diagbox:w
              [ leftsep=\leftsep, rightsep=\rightsep, ##1 ]
              { \__tblr_lib_diagbox_math_or_text:n {##2} }
              { \__tblr_lib_diagbox_math_or_text:n {##3} }
          }
      }
    \NewTblrContentCommand \diagboxthree [4] []
      {
        \__tblr_lib_diagbox_fix:n
          {
            \__tblr_lib_saved_diagbox:w
              [ leftsep=\leftsep, rightsep=\rightsep, ##1 ]
              { \__tblr_lib_diagbox_math_or_text:n {##2} }
              { \__tblr_lib_diagbox_math_or_text:n {##3} }
              { \__tblr_lib_diagbox_math_or_text:n {##4} }
          }
      }
  }

\cs_new_protected:Npn \__tblr_lib_diagbox_math_or_text:n #1
  {
    \bool_if:NTF \l__tblr_cell_math_mode_bool {$#1$} {#1}
```

```
  }

\box_new:N \l__tblr_diag_box

\cs_new_protected:Npn \__tblr_lib_diagbox_fix:n #1
  {
    \hbox_set:Nn \l__tblr_diag_box {#1}
    \box_set_ht:Nn \l__tblr_diag_box { \box_ht:N \l__tblr_diag_box - \abovesep }
    \box_set_dp:Nn \l__tblr_diag_box { \box_dp:N \l__tblr_diag_box - \belowsep }
    \box_use:N \l__tblr_diag_box
  }

%% Library functional with evaluate and process options

\cs_set_eq:NN \__tblr_functional_calculation: \prg_do_nothing:

\NewTblrLibrary { functional }
  {
    \RequirePackage { functional }
    %% Add outer specification "evaluate"
    \__tblr_keys_define:nn { table/outer }
      { evaluate .code:n = \__tblr_outer_gput_spec:nn { evaluate } {##1} }
    \tl_new:N \l__tblr_evaluate_tl
    \cs_set_protected:Npn \__tblr_hook_split_before:
      {
        \tl_set:Ne \l__tblr_evaluate_tl
          { \__tblr_spec_item:nn { outer } { evaluate } }
        \tl_if_empty:NF \l__tblr_evaluate_tl
          {
            \tl_if_eq:NnTF \l__tblr_evaluate_tl { all }
              {
                \tlSet \l__tblr_body_tl { \evalWhole {\expValue \l__tblr_body_tl} }
              }
              {
                \exp_last_unbraced:NNV
                \__tblr_evaluate_table_body:NN \l__tblr_body_tl \l__tblr_evaluate_tl
              }
          }
      }
    %% Evaluate every occurrence of the specified function
    %% Note that functional package runs every return processor inside a group
    %% #1: tl with table content; #2: function to be evaluated
    \tl_new:N \g__tblr_functional_result_tl
    \cs_new_protected:Npn \__tblr_evaluate_table_body:NN ##1 ##2
      {
        \tl_gclear:N \g__tblr_functional_result_tl
        \cs_set_protected:Npn \__tblr_evaluate_table_body_aux:w ####1 ##2
          {
            \tl_gput_right:Nn \g__tblr_functional_result_tl {####1}
            \peek_meaning:NTF \q_stop { \use_none:n } {##2}
          }
        \fun_run_return_processor:nn
          {
            \exp_last_unbraced:NV \__tblr_evaluate_table_body_aux:w \gResultTl
          }
          {
            \exp_last_unbraced:NV
```

```
            \__tblr_evaluate_table_body_aux:w ##1 ##2 \q_stop
        }
      \tl_set_eq:NN ##1 \g__tblr_functional_result_tl
    }
  %% Add inner specification "process"
  \__tblr_keys_define:nn { table/inner }
    { process .code:n = \__tblr_keys_gput:nn { process } {##1} }
  \cs_set:Npn \__tblr_functional_calculation:
    {
      \LogTblrTracing { step = do ~ functional ~ calculation }
      \__tblr_prop_item:nn { inner } { process }
    }
  \prgNewFunction \cellGetText { m m }
    {
      \expWhole { \__tblr_spec_item:nn { text } { [##1][##2] } }
    }
  \prgNewFunction \cellSetText { m m m }
    {
      \__tblr_spec_gput:nnn { text } { [##1][##2] } {##3}
    }
  \prgNewFunction \cellSetStyle { m m m }
    {
      \tblr_set_cell:nnnn {##1} {##2} {} {##3}
    }
  \prgNewFunction \rowSetStyle { m m }
    {
      \tblr_set_row:nnn {##1} {} {##2}
    }
  \prgNewFunction \columnSetStyle { m m }
    {
      \tblr_set_column:nnn {##1} {} {##2}
    }
  %% Evaluate every function in inner specification
  %% But we need to protect the value of "process" key
  \clist_new:N \l__tblr_fun_keyvalue_clist
  \cs_new_protected:Npn \__tblr_fun_i:n ##1
    {
      \clist_put_right:Nn \l__tblr_fun_keyvalue_clist { ##1 }
    }
  \cs_new_protected:Npn \__tblr_fun_ii:nn ##1 ##2
    {
      \tl_if_eq:nnTF { ##1 } { process }
        {
          \clist_put_right:Nn \l__tblr_fun_keyvalue_clist
            { ##1 = {\evalNone{##2}} }
        }
        {
          \clist_put_right:Nn \l__tblr_fun_keyvalue_clist
            { ##1 = {##2} }
        }
    }
  \cs_set_protected:Npn \__tblr_hook_parse_inner_spec_before:
    {
      \clist_clear:N \l__tblr_fun_keyvalue_clist
      \keyval_parse:NNV \__tblr_fun_i:n \__tblr_fun_ii:nn \l__tblr_inner_spec_tl
      \tlSet \l__tblr_inner_spec_tl
        { \evalWhole { \expValue \l__tblr_fun_keyvalue_clist } }
    }
```

```
  }

%% Library hook provides some public hooks

\NewTblrLibrary { hook }
  {
    %% We need varwidth to keep \lTblrMeasuringBool correct
    \UseTblrLibrary { varwidth }
    \tl_set:Nn \l__tblr_inner_spec_measure_tl { vstore }
    \cs_set_eq:NN \__tblr_hook_use:n \__tblr_hook_use_true:n
    %% Be careful lthooks will not remove any spaces in hook paths
    \__tblr_hook_new_pair:nn { trial/before } { trial/after }
    \__tblr_hook_new_pair:nn { table/before } { table/after }
    \__tblr_hook_new_pair:nn {   row/before } {   row/after }
    \__tblr_hook_new_pair:nn {  cell/before } {  cell/after }
    \__tblr_hook_new_pair:nn { private/output/before } { private/output/after }
  }

%% Library html provides more public variables
%% These variables can be used by tagpdf, tex4ht and lwarp packages

\bool_new:N \l__tblr_html_variables_bool

\NewTblrLibrary { html }
  {
    \bool_set_true:N \l__tblr_html_variables_bool
  }

%% Library nameref and its caption-ref template

\NewTblrLibrary { nameref }
  {
    \RequirePackage { nameref }
    \clist_if_in:NnF \lTblrRefMoreClist { nameref }
      {
        \clist_put_right:Nn \lTblrRefMoreClist { nameref }
        \DeclareTblrTemplate { caption-ref }{ nameref }
          {
            \tl_if_eq:NnTF \lTblrEntryTl { none }
              { \exp_args:NV \GetTitleString \lTblrCaptionTl }
              {
                \tl_if_empty:NTF \lTblrEntryTl
                  { \exp_args:NV \GetTitleString \lTblrCaptionTl }
                  { \exp_args:NV \GetTitleString \lTblrEntryTl }
              }
            \tl_set_eq:NN \@currentlabelname \GetTitleStringResult
          }
      }
  }

%% Library siunitx and S columns

\NewTblrLibrary { siunitx }
  {
    \RequirePackage { siunitx }
```

```
    \NewTblrColumnType { S } [1] [] { Q[si = {##1}, c] }
    \NewTblrColumnType { s } [1] [] { Q[si = {##1}, c, cmd = \TblrUnit] }
    \__tblr_data_new_key:nnn { cell } { si } { str }
    \__tblr_keys_define:nn { column/inner }
      {
        si .code:n = \__tblr_siunitx_setcolumn:n {##1}
      }
    \cs_new_protected:Npn \__tblr_siunitx_setcolumn:n ##1
      {
        \__tblr_column_gput_cell:nn { si } {##1}
        \__tblr_column_gput_cell:nn { cmd } { \TblrNum }
      }
    \NewDocumentCommand \TblrNum { m }
      {
        \__tblr_siunitx_process:Nn \tablenum {##1}
      }
    \NewDocumentCommand \TblrUnit { m }
      {
        \__tblr_siunitx_process:Nn \si {##1}
      }
    \cs_new_protected:Npn \__tblr_siunitx_process:Nn ##1 ##2
      {
        \tl_if_head_is_group:nTF {##2}
          { ##2 }
          {
            \group_begin:
            \tl_set:Ne \l_tmpa_tl
              {
                \__tblr_data_item:neen { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { si }
              }
            \exp_args:NV \sisetup \l_tmpa_tl
            ##1 {##2}
            \group_end:
          }
      }
    \__tblr_keys_define:nn { cell/inner }   { guard .meta:n = { cmd = } }
    \__tblr_keys_define:nn { row/inner }    { guard .meta:n = { cmd = } }
    \__tblr_keys_define:nn { column/inner } { guard .meta:n = { cmd = } }
  }

%% Library tikz to add table nodes, cell nodes and corner nodes with tikz

\tl_new:N \g__tblr_tikz_below_code_tl
\tl_new:N \g__tblr_tikz_above_code_tl
\dim_new:N \l__tblr_cell_inner_wd_dim
\dim_new:N \l__tblr_cell_inner_dp_dim
\dim_new:N \l__tblr_cell_hanchor_dim
\dim_new:N \l__tblr_last_cell_table_ht_dim
\dim_new:N \l__tblr_last_cell_table_dp_dim
\dim_new:N \l__tblr_last_vline_wd_dim
\box_new:N \l__tblr_tikz_node_box

\NewTblrLibrary { tikz }
  {
    \UseTblrLibrary { hook }
    \RequirePackage { tikz }
```

```
\usetikzlibrary { calc }
\tikzset
  {
    tblr / cell ~ node / .style =
      {
        rectangle ,
        inner ~ sep = 0pt ,
        outer ~ sep = 0pt ,
        draw = none ,
        fill = none
      } ,
    tblr / cell / .style = { } ,
    tblr / table ~ node / .style =
      {
        rectangle ,
        inner ~ sep = 0pt ,
        outer ~ sep = 0pt ,
        draw = none ,
        fill = none
      } ,
    tblr / table / .style = { } ,
    tblr / overlay / .style = { }
  }
\NewDocumentEnvironment { tblrtikzbelow } { +b }
  {
    \tl_gset:Nn \g__tblr_tikz_below_code_tl
      {
        \begin{tikzpicture}
          [
            remember ~ picture ,
            overlay ,
            name ~ prefix = tblr \g__tblr_name_str - ,
            tblr / overlay
          ]
          ##1
        \end{tikzpicture}
      }
  } { }
\NewDocumentEnvironment { tblrtikzabove } { +b }
  {
    \tl_gset:Nn \g__tblr_tikz_above_code_tl
      {
        \begin{tikzpicture}
          [
            remember ~ picture ,
            overlay ,
            name ~ prefix = tblr \g__tblr_name_str - ,
            tblr / overlay
          ]
          ##1
        \end{tikzpicture}
      }
  } { }
\cs_new_protected:Npn \__tblr_tikz_make_cell_node:
  {
    \hbox_set:Nn \l__tblr_tikz_node_box
      {
        \begin{tikzpicture}[remember ~ picture, overlay]
```

```
            \coordinate
              (
                tblr \g__tblr_name_str -
                \int_use:N \c@rownum - \int_use:N \c@colnum -
                single
              )
              at ( 0pt , -\l__tblr_row_dp_dim );
            \node
              [
                tblr / cell ~ node ,
                tblr / cell ,
                anchor = center ,
                text ~ width = \l__tblr_cell_inner_wd_dim ,
                text ~ height = \l__tblr_row_ht_dim ,
                text ~ depth = \l__tblr_cell_inner_dp_dim
              ]
              (
                tblr \g__tblr_name_str -
                \int_use:N \c@rownum - \int_use:N \c@colnum
              )
              at
              (
                \l__tblr_cell_hanchor_dim - 0.5 \l__tblr_cell_inner_wd_dim
                ,
                0.5 \l__tblr_row_ht_dim - 0.5 \l__tblr_cell_inner_dp_dim
              ) { } ;
          \end{tikzpicture}
        }
    }
  \cs_new_protected:Npn \__tblr_tikz_make_table_node:
    {
      \hbox_set:Nn \l__tblr_tikz_node_box
        {
          \begin{tikzpicture}[remember ~ picture, overlay]
            \node
              [
                tblr / table ~ node ,
                tblr / table ,
                anchor = center ,
                text ~ width = \lTblrTableWidthDim ,
                text ~ height = \l__tblr_last_cell_table_ht_dim ,
                text ~ depth = \l__tblr_last_cell_table_dp_dim
              ]
              (
                tblr \g__tblr_name_str - table
              )
              at
              (
                \l__tblr_last_vline_wd_dim - 0.5 \lTblrTableWidthDim
                ,
                0.5 \l__tblr_last_cell_table_ht_dim
                - 0.5 \l__tblr_last_cell_table_dp_dim
              ) { } ;
          \end{tikzpicture}
        }
    }
  \cs_new_protected:Npn \__tblr_tikz_enable_cell_node:
    {
```

```
      \__tblr_get_cell_background_width:NNN
          \c@rownum \c@colnum \l__tblr_cell_inner_wd_dim
      \__tblr_get_cell_background_depth:NNN
          \c@rownum \c@colnum\l__tblr_cell_inner_dp_dim
      \dim_zero:N \l__tblr_cell_hanchor_dim
      \tl_set:Ne \l__tblr_c_tl
        {
          \__tblr_data_item:neen { cell }
            { \int_use:N \c@rownum } { \int_use:N \c@colnum } { colspan }
        }
      \int_compare:nNnT { \l__tblr_c_tl } > { 1 }
        {
          \int_step_inline:nn { \l__tblr_c_tl - 1 }
            {
              \dim_add:Nn \l__tblr_cell_hanchor_dim
                {
                  \__tblr_spec_item:ne { vline }
                    { [ \int_eval:n { \c@colnum + ####1 } ] / @vline-width }
                  +
                  \__tblr_data_item:nen { column }
                    { \int_eval:n { \c@colnum + ####1 } } { @col-width }
                  +
                  \__tblr_data_item:nen { column }
                    { \int_eval:n { \c@colnum + ####1 } } { leftsep }
                  +
                  \__tblr_data_item:nen { column }
                    { \int_eval:n { \c@colnum + ####1 } } { rightsep }
                }
            }
        }
      \__tblr_tikz_make_cell_node:
      \box_use:N \l__tblr_tikz_node_box
    }
  \cs_new_protected:Npn \__tblr_tikz_enable_table_node:
    {
      \int_compare:nNnT { \c@rownum } = { \c@rowcount }
        {
          \int_compare:nNnT { \c@colnum } = { \c@colcount }
            {
              \__tblr_get_table_width:
              \__tblr_get_table_height:
              \__tblr_get_table_depth:
              \dim_set:Nn \l__tblr_last_vline_wd_dim
                {
                  \__tblr_spec_item:ne { vline }
                    { [ \int_eval:n { \c@colcount + 1 } ] / @vline-width }
                }
              \__tblr_tikz_make_table_node:
              \box_use:N \l__tblr_tikz_node_box
            }
        }
    }
  \tl_new:N \l__tblr_tikz_corner_node_code_tl
  \cs_new_protected:Npn \__tblr_tikz_make_corner_node:
    {
      \tl_set:Nn \l__tblr_tikz_corner_node_code_tl
        { \coordinate (h1) at ($(table.north~west)!0.5!(1-1.north~west)$); }
      \int_step_inline:nnn { 2 } { \c@rowcount }
```

```
        {
          \tl_put_right:Ne \l__tblr_tikz_corner_node_code_tl
            {
              \exp_not:N \coordinate (h####1) at
                (
                  h1 |-
                  {
                    $(\int_eval:n{####1-1}-1-single)!0.5!(####1-1.north~west)$
                  }
                );
            }
        }
      \tl_put_right:Ne \l__tblr_tikz_corner_node_code_tl
        {
          \exp_not:N \coordinate (h\int_eval:n{\c@rowcount+1}) at
            (h1|-{$(\int_use:N\c@rowcount-1-single)!0.5!(table.south~east)$});
        }
      \tl_put_right:Nn \l__tblr_tikz_corner_node_code_tl
        { \coordinate (v1) at (h1); }
      \int_step_inline:nnn { 2 } { \c@colcount }
        {
          \tl_put_right:Ne \l__tblr_tikz_corner_node_code_tl
            {
              \exp_not:N \coordinate (v####1) at
                (
                  v1 -|
                  {
                    $(1-\int_eval:n{####1-1}-single)!0.5!(1-####1.north~west)$
                  }
                );
            }
        }
      \tl_put_right:Ne \l__tblr_tikz_corner_node_code_tl
        {
          \exp_not:N \coordinate (v\int_eval:n{\c@colcount+1}) at
            (v1-|{$(1-\int_use:N\c@colcount-single)!0.5!(table.south~east)$});
        }
    }
  \tl_const:Nn \c__tblr_portrait_long_tl { long }
  \cs_new_protected:Npn \__tblr_tikz_enable_corner_node:
    {
      \bool_lazy_or:nnT
        { ! \tl_if_eq_p:NN \lTblrPortraitTypeTl \c__tblr_portrait_long_tl }
        {
          \bool_lazy_and_p:nn
            { \int_compare_p:nNn { \lTblrRowFirstInt } = {1} }
            { \int_compare_p:nNn { \lTblrRowLastInt } = { \c@rowcount } }
        }
        {
          \__tblr_tikz_make_corner_node:
          \begin{tikzpicture}
            [
              remember ~ picture, overlay,
              name ~ prefix = tblr \g__tblr_name_str -
            ]
            \tl_use:N \l__tblr_tikz_corner_node_code_tl
          \end{tikzpicture}
        }
```

```
    }
  \AddToTblrHook { cell/after } [ tblrlibtikz ]
    {
      \bool_lazy_and:nnF
        { \tl_if_empty_p:N \g__tblr_tikz_below_code_tl }
        { \tl_if_empty_p:N \g__tblr_tikz_above_code_tl }
        {
          \__tblr_tikz_enable_cell_node:
          \__tblr_tikz_enable_table_node:
        }
    }
  \AddToTblrHook { private/output/before } [ tblrlibtikz ]
    {
      \bool_lazy_and:nnF
        { \tl_if_empty_p:N \g__tblr_tikz_below_code_tl }
        { \tl_if_empty_p:N \g__tblr_tikz_above_code_tl }
        {
          \__tblr_tikz_enable_corner_node:
          \noindent
          \tl_use:N \g__tblr_tikz_below_code_tl
        }
    }
  \AddToTblrHook { private/output/after } [ tblrlibtikz ]
    {
      \tl_use:N \g__tblr_tikz_above_code_tl
    }
  \AddToTblrHook { table/after } [ tblrlibtikz ]
    {
      %% We clear them here since they are used several times in longtblr
      \tl_gclear:N \g__tblr_tikz_below_code_tl
      \tl_gclear:N \g__tblr_tikz_above_code_tl
    }
}

\cs_new_protected:Npn \__tblr_get_table_height:
  {
    \dim_zero:N \l__tblr_last_cell_table_ht_dim
    \int_step_inline:nn { \c@rowcount - 1 }
      {
        \dim_add:Nn \l__tblr_last_cell_table_ht_dim
          { \__tblr_valign_get_hline_total:n {##1} }
        \dim_add:Nn \l__tblr_last_cell_table_ht_dim
          { \__tblr_valign_get_row_total:n {##1} }
      }
    \dim_add:Nn \l__tblr_last_cell_table_ht_dim
      { \__tblr_valign_get_hline_total:n { \int_use:N \c@rowcount } }
    \dim_add:Nn \l__tblr_last_cell_table_ht_dim
      { \__tblr_valign_get_row_height:n { \int_use:N \c@rowcount } }
  }

\cs_new_protected:Npn \__tblr_get_table_depth:
  {
    \dim_set:Nn \l__tblr_last_cell_table_dp_dim
      { \__tblr_valign_get_row_depth:n { \int_use:N \c@rowcount } }
    \dim_add:Nn \l__tblr_last_cell_table_dp_dim
      { \__tblr_valign_get_hline_total:n { \int_eval:n { \c@rowcount + 1 } } }
  }
```

```
%% Library varwidth and measure option

\NewTblrLibrary { varwidth }
  {
    \RequirePackage { varwidth }
    \tl_set:Nn \l__tblr_inner_spec_measure_tl { vbox }
    \__tblr_keys_define:nn { table/inner }
      { measure .tl_set:N = \l__tblr_inner_spec_measure_tl }
  }

%% Library zref and its caption-ref template

\NewTblrLibrary { zref }
  {
    \RequirePackage { zref-user }
    \clist_if_in:NnF \lTblrRefMoreClist { zref }
      {
        \clist_put_right:Nn \lTblrRefMoreClist { zref }
        \DeclareTblrTemplate { caption-ref }{ zref }
          {
            \exp_args:NV \zlabel \lTblrLabelTl
          }
      }
  }
```