Tabularray	Typeset Tabulars and Arrays with LaTeX3
Author	Jianrui Lyu (tolvjr@163.com)
Version	2025@ (2025-03-02)
Code	https://github.com/lvjr/tabularray
Code	https://bitbucket.org/lvjr/tabularray
Support	https://github.com/lvjr/tabularray/discussions
Support	https://topanswers.xyz/tex
Issue	https://github.com/lvjr/tabularray/issues

```
\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} \\
  Support    & \url{https://github.com/lvjr/tabularray/discussions} \\
  Support    & \url{https://topanswers.xyz/tex} \\
  Issue      & \url{https://github.com/lvjr/tabularray/issues} \\
\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{\ch{+}}},
  vline{4} = {1}{text=\clap{\ch{->}}},
  vline{5} = {1}{text=\clap{\ch{+}}},
}
  Equation & \ch{CH4} & \ch{2 O2} & \ch{CO2} & \ch{2 H2O} \\
  Initial  & $n_1$    & $n_2$     & 0        & 0 \\
  Final    & $n_1-x$  & $n_2-2x$  & $x$      & $2x$ \\
\end{tblr}
```

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

You need to load `chemmacros` package for the `\ch` command.

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 postive 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}
```

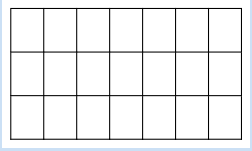
Row1	1
Text	Text
Row3	3
Text	Text
Row5	5

```
\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 instersection point of `hborder{<i>}` and `vborder{1}`
corner node `v<j>`	coordinate node at the instersection 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 instersection point of `horder{<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 writng 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 curent 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