

# dbshow 宏包\* $\Rightarrow$ English Version

李昌锴 <lichangkai225@qq.com>

2022 年 1 月 7 日

## Contents

<b>1</b>	<b>引言</b>	<b>2</b>
<b>2</b>	<b>接口文档</b>	<b>2</b>
2.1	\dbNewDatabase . . . . .	2
2.2	\dbNewStyle 和样式选项 . . . . .	3
2.3	使用 \dbNewReviewPoints 定义复习点 . . . . .	5
2.4	在 dbFilters 环境中定义过滤器 . . . . .	5
2.5	使用 dbitem 环境存储数据 . . . . .	6
2.6	\dbsave 和 \dbuse . . . . .	6
2.7	使用 \dbshow 展示数据库 . . . . .	6
2.8	条件判别式 . . . . .	7
2.9	特殊命令 . . . . .	7
<b>3</b>	<b>错题本示例</b>	<b>8</b>
<b>4</b>	<b>Introduction</b>	<b>9</b>
<b>5</b>	<b>Interfaces</b>	<b>9</b>
5.1	\dbNewDatabase . . . . .	9
5.2	\dbNewStyle and Style Options . . . . .	10
5.3	Use \dbNewReviewPoints to Define Review Points . . . . .	12
5.4	Define Filters inside dbFilters Environment . . . . .	12
5.5	Store Data with dbitem Environment . . . . .	13
5.6	\dbsave and \dbuse . . . . .	13
5.7	Use \dbshow to Display the Database . . . . .	14
5.8	Conditionals . . . . .	14
5.9	Special Macros . . . . .	15

---

\*<https://github.com/ZhiyuanLck/dbshow>



$\langle attr \rangle = \langle type \rangle$  将  $\langle attr \rangle$  声明为  $\langle type \rangle$  类型

$\langle attr \rangle = \langle type \rangle | \langle default \rangle$  将  $\langle attr \rangle$  声明为  $\langle type \rangle$  类型，并且将默认值设置为  $\langle default \rangle$ 。

**NOTE:** 每个数据库都有一个默认的属性 `id` 用来存储数据的索引。

下面是定义一个错题数据库的示例，`question` 和 `answer` 属性用来存储问题和答案，`date` 属性存储日期，`info` 属性存储额外信息，`labels` 存储题目标签。

```
\dbNewDatabase{ques}{
  question = tl,
  answer = tl,
  date = date,
  info = tl,
  labels = clist
}
```

## 2.2 \dbNewStyle 和样式选项

---

`\dbNewStyle`    `\dbNewStyle` [ $\langle base\ styles \rangle$ ] [ $\langle style \rangle$ ] [ $\langle database \rangle$ ] [ $\langle opts \rangle$ ]

---

为  $\langle database \rangle$  定义一个新的样式  $\langle style \rangle$ ，该样式可以基于已有的样式  $\langle base\ styles \rangle$ ，比如 `\dbNewStyle[base1, base2]{new-style}{ques}`。

**filter** =  $\langle filter \rangle$  (initially -none-)

为当前样式设置由 `\dbCombineFilters` 所定义的过滤器

**raw-filter** =  $\langle cond\ combination \rangle$  (no value)

用已定义的条件为当前样式设置匿名过滤器。下面两个例子的作用是等同的。

```
% method 1
\begin{dbFilters}{db}
  \dbNewConditional{cond1}{int-attr}{\rval > 1}
  \dbNewConditional*{cond2}{str-attr}{\d+}
\end{dbFilters}
\dbNewStyle{style}{db}{raw-filter={cond1 && cond2}}

% method 2
\begin{dbFilters}{db}
  \dbNewConditional{cond1}{int-attr}{\rval > 1}
  \dbNewConditional*{cond2}{str-attr}{\d+}
  \dbCombineFilters{filter}{cond1 && cond2}
\end{dbFilters}
\dbNewStyle{style}{db}{filter=filter}
```

**sort** = {  $\langle attr\ spec1 \rangle$ ,  $\langle attr\ spec2 \rangle$ , ... } (no value)

为当前样式设置排序规则。支持根据 `str`, `date`, `int`, `fp` 类型的数据进行排序，支持多级排序。 $\langle attr \rangle$  表示增序， $\langle attr \rangle^*$  表示降序。下面例子中，使用 `sort-style` 展示数据时的顺序为先按 `level` 降序，`level` 相同的再按出生日期 `birth` 增序，以此类推。

```

\dbNewDatabase{sort-example}{
    name = str,
    birth = date,
    level = int,
    weight = fp,
}
\dbNewStyle{sort-style}{sort-example}{
    sort = { level*, birth, name, weight }
}

```

**before-code** = *<before code>* (no value)

该选项用来设置在展示整个数据库之前需要执行的代码。

**after-code** = *<after code>* (no value)

该选项用来设置在展示整个数据库之后需要执行的代码。

**item-code** = *<item code>* (no value)

该选项用来设置展示数据库中每条记录的代码。你可以使用 `\dbuse` 来展示属性的值。

**<attr>/before-code** = *<before code>* (no value)

该选项用来设置展示数据库中属性 *<attr>* 对应数据之前需要执行的代码。`\dbuse` 会在展示属性数据前执行此代码。

**<attr>/after-code** = *<after code>* (no value)

该选项用来设置展示数据库中属性 *<attr>* 对应数据之后需要执行的代码。`\dbuse` 会在展示属性数据后执行此代码。

**<attr>/sep** = *<sep spec>* (initially ,~)

**<attr>/sep** = *<separator>*

**<attr>/sep** = {

*<separator between two>*,

*<separator between more than two>*,

*<separator between final two>*

}

该选项只适用于类型为 `clist` 的属性，用来设置列表间元素的间隔。第一个版本接受一个参数，将所有的元素间隔设置为 *<separator>*。第二个版本接受逗号分隔的三个参数，分别用来设置只有两个元素时的分隔符 *<separator between two>*，超过两个元素时的分隔符 *<separator between more than two>*，和最后两个元素之间的分隔符 *<separator between final two>*。

**item-before-code** = *<before code>* (no value)

该选项只适用于类型为 `clist` 的属性，用来设置展示列表每个元素前需要执行的代码。

**item-after-code** = *<after code>* (no value)

该选项只适用于类型为 `clist` 的属性，用来设置展示列表每个元素后需要执行的代码。

## 2.3 使用 \dbNewReviewPoints 定义复习点

---

```
\dbNewReviewPoints \dbNewReviewPoints {<name>} {<points>}
```

---

定义名为  $\langle name \rangle$  的复习点。这是专门为错题本或复习所定制的功能， $\langle points \rangle$  是一系列整数，现在假设每道错题你都将写错时的日期记录在了 `date` 属性中，并且你希望每隔 2, 5, 15 天复习一次。下面的代码给出了一个实现示例。

```
\dbNewReviewPoints{review-point}{2, 5, 15} % 定义复习点
\begin{dbFilters}
  \dbNewConditional{cond1}{date}{review-point|\Today} % 定义复习条件
  \dbCombineConditionals{filter1}{cond1} % 定义过滤器
\end{dbFilters}
\dbNewStyle{review-style}{ques}{filter=filter1} % 定义展示样式
```

## 2.4 在 dbFilters 环境中定义过滤器

```
dbFilters \begin{dbFilters}{<database>}
  <code>
\end{dbFilters}
```

`dbFilters` 用来定义过滤器，此环境中定义了 `\dbNewConditional` 命令用来定义条件和 `\dbCombineConditionals` 命令用来组合条件定义过滤器。过滤器独立于每个  $\langle database \rangle$ ，这意味着你可以在不同数据库中定义名称相同的过滤条件和过滤器。

---

```
\dbNewConditional \dbNewConditional {<name>} {<attr>} {<cond spec>}
\dbNewConditional* \dbNewConditional* {<name>} {<attr>} {<cond spec>}
\dbNewConditional {<name>} {<int/fp attr>} {<relation>}
\dbNewConditional* {<name>} {<int/fp attr>} {<relation>}
\dbNewConditional {<name>} {<str/tl attr>} {<regex expr>}
\dbNewConditional* {<name>} {<str/tl attr>} {<regex expr>}
\dbNewConditional {<name>} {<clist attr>} {<val list>}
\dbNewConditional* {<name>} {<clist attr>} {<val list>}
\dbNewConditional {<name>} {<date attr>} {<review points>|<date>}
\dbNewConditional* {<name>} {<date attr>} {<relation>}
```

---

`\dbNewConditional` 用来定义名为  $\langle name \rangle$  的条件， $\langle attr \rangle$  指定条件所绑定的属性，在  $\langle cond spec \rangle$  中可以用 `\dbval` 指代属性的值。

对于类型为 `int` 和 `fp` 的属性，两个版本的定义是一致的， $\langle relation \rangle$  可以是单个关系式，比如 `\dbval > 3`，也可以是组合关系式，比如 `\dbval > 3 && \dbval < 10.2`。支持的操作符有 `<`, `>`, `=`, `==`, `!=`, `>=`, `<=`, `!`。

对于类型为 `str` 和 `tl` 的属性， $\langle regex \rangle$  为正则表达式，`\dbNewConditional` 表示部分匹配，`\dbNewConditional*` 表示整体匹配。

```
\dbNewConditional{cond1}{str-attr}{abc} % 匹配 abc, abcd, 1abc, =abc= 等
\dbNewConditional*{cond2}{str-attr}{abc} % 只匹配 abc
```

对于类型为 `clist` 的属性，使用 `\dbNewConditional` 定义的条件只要  $\langle val list \rangle$  中的任意一个元素在属性值（列表）中则条件成立；使用 `\dbNewConditional*` 定义的条件只有  $\langle val list \rangle$  中每一个值都在属性值（列表）中条件才成立。

```
\dbNewConditional{cond1}{clist-attr}{a, b, c} % a, b, d 满足条件
\dbNewConditional*{cond2}{clist-attr}{a, b, c} % a, b, d 不满足条件
```

对于类型为 `date` 的属性, `\dbNewConditional` 使用复习点来定义过滤条件, `\dbNewReviewPoints` 定义的复习点, `\dbNewConditional*` 使用单个关系式来定义过滤条件。支持的操作符有 `<`, `>`, `=`, `==`, `!=`, `>=`, `<=`。

---

```
\dbCombineConditionals \dbCombineConditionals {<name>} {<cond combination>} [<info>]
```

---

`\dbCombineConditionals` 定义名为 `{<name>}` 的过滤器, 并将 `\dbNewConditional` 定义的条件组合起来, 比如 `\dbCombineConditionals{filter}{(cond1 && cond2) || !cond3}`。`<cond combination>` 中可以使用的关系操作符为 `&&`, `||`, `!`。可以将 `filter` 选项设置为 `<name>` 来应用过滤器。`<info>` 为过滤器的相关信息, 在展示数据库的时候可以用 `\dbFilterInfo` 指代。

## 2.5 使用 dbitem 环境存储数据

```
dbitem \begin{dbitem}{<database>}[
    <attr1> = <val1>,
    <attr2> = <val2>,
    ...
]
    <code>
\end{dbitem}
```

`dbitem` 环境用来存储数据。有两种存储数据的方法, 较短的数据可以在选项列表中通过键值对设置值, 较长的数据可以在 `<code>` 中使用 `\dbsave` 存储。`\dbsave` 会覆盖选项中设置的值。没有设置的值将会被设置为全局默认值, 下面给出一个存储示例。

```
\begin{dbitem}[date=2022-01-01, info= 测试]
    \dbsave{question}{这是一个测试问题}
    \dbsave{answer}{这是一个测试答案}
\end{dbitem}
```

## 2.6 \dbsave 和 \dbuse

---

```
\dbsave \dbsave {<attr>} {<data>}
\dbuse \dbuse {<attr>}
```

---

`\dbsave` 用来存储数据, 只能在 `item` 环境中使用。`\dbuse` 用来使用数据, 只能在 `item-code` 选项中使用。

## 2.7 使用 \dbshow 展示数据库

---

```
\dbshow \dbshow {<style>} {<database>}
    使用 <style> 样式来展示 <database>。
```

---

## 2.8 条件判别式

---

<code>\dbIfEmptyT</code>	<code>\dbIfEmptyTF {<i>\langle true code \rangle</i>} {<i>\langle false code \rangle</i>}</code>
<code>\dbIfEmptyF</code>	<code>\dbIfEmptyT {<i>\langle true code \rangle</i>}</code>
<code>\dbIfEmptyTF</code>	<code>\dbIfEmptyF {<i>\langle false code \rangle</i>}</code>

---

该判别式用来判断当前数据库是否为空。下面的示例展示了如何预防空的列表环境。

```
\dbNewStyle{style-cond1}{database-test}{
  before-code = {\dbIfEmptyF{\begin{enumerate}}},
  after-code = {\dbIfEmptyF{\end{enumerate}}},
  item-code = {\item \dbuse{attr-test}}
}
```

---

<code>\dbItemIfEmptyT</code>	<code>\dbItemIfEmptyTF {<i>\langle true code \rangle</i>} {<i>\langle false code \rangle</i>}</code>
<code>\dbItemIfEmptyF</code>	<code>\dbItemIfEmptyT {<i>\langle true code \rangle</i>}</code>
<code>\dbItemIfEmptyTF</code>	<code>\dbItemIfEmptyF {<i>\langle false code \rangle</i>}</code>

---

该判别式用来判断当前元素是否为空。下面的示例展示了如何在展示 `database-test` 数据库中 `text` 属性的元素时在元素非空的时候前后都加上 `*`。

```
\dbNewStyle{style-cond2}{database-test}{
  text/before-code = {\dbItemIfEmptyF{*}},
  text/after-code = {\dbItemIfEmptyF{*}},
}
```

---

<code>\dbClistItemIfEmptyT</code>	<code>\dbClistItemIfEmptyTF {<i>\langle true code \rangle</i>} {<i>\langle false code \rangle</i>}</code>
<code>\dbClistItemIfEmptyF</code>	<code>\dbClistItemIfEmptyT {<i>\langle true code \rangle</i>}</code>
<code>\dbClistItemIfEmptyTF</code>	<code>\dbClistItemIfEmptyF {<i>\langle false code \rangle</i>}</code>

---

该判别式用来判断列表属性中的元素是否为空。下面的示例展示了如何在展示 `database-test` 数据库中 `labels` 属性（标签列表）的元素时在标签非空的时候前后都加上 `*`。

```
\dbNewDatabase{database-test}{labels=clist}
\dbNewStyle{style-cond3}{database-test}{
  labels/item-before-code = {\dbClistItemIfEmptyF{*}},
  labels/item-after-code = {\dbClistItemIfEmptyF{*}},
}
```

## 2.9 特殊命令

`dbshow` 定义了一些特殊的命令，会根据语境展开为不同的内容。

<code>\dbval</code>	当前属性的值，只能在 <code>\dbNewConditional</code> 中使用
<code>\dbDatabase</code>	数据库名称
<code>\dbFilterName</code>	过滤器名称
<code>\dbFilterInfo</code>	过滤器信息
<code>\dbIndex</code>	原始数据索引，等同于 <code>\dbuse{id}</code> 但是 <code>\dbuse</code> 是不可展开的

<code>\dbarabic</code>	用数字表示的查询集数据计数
<code>\dbalph</code>	用小写字母表示的查询集数据计数
<code>\dbAlph</code>	用小写字母表示的查询集数据计数
<code>\dbroman</code>	用小写罗马数字表示的查询集数据计数
<code>\dbRoman</code>	用大写罗马数字表示的查询集数据计数

### 3 错题本示例

见第 6 节。

⇒ 中文版本

2022/01/07

The initial motivation to write this package is that I want to write a template, which can collect questions you gave the wrong answer and can display those questions you would like to review by some conditionals, such as questions with certain label, questions you have answered uncorrectly for certain times or questions having not been reviewed for certain days. So this package provides a database to do such thing.

The package constructs 6 types based on the internal typed of `expl3`:

**clist** comma list, default empty.

## 5.1 \dbNewDatabase

\*<https://github.com/ZhiyuanLck/dbshow>

Create a new database named  $\langle database \rangle$ , unstarred form provides the optional  $\langle base\ database \rangle$  from which current database inherit the attributes settings. The unstarred form always replace the old definition, while starred form appends the new options.

$\langle attr \rangle = \langle type \rangle$

$\langle attr \rangle = \langle type \rangle | \langle default \rangle$

The first form defines the  $\langle attr \rangle$  as  $\langle type \rangle$ , and the second also sets the default value.

**NOTE:** Every database has a default attribute `id` to store the index of the item.

The example below define a database named `ques`.

```
\dbNewDatabase{ques}{
  question = tl, % store question
  answer = tl, % store corresponding answer
  date = date, % store the date when you were wrong
  info = tl, % store extra info
  labels = clist % store question labels
}
```

## 5.2 \dbNewStyle and Style Options

---

**\dbNewStyle**     $\backslash dbNewStyle$  [ $\langle base\ styles \rangle$ ] [ $\langle style \rangle$ ] [ $\langle database \rangle$ ] [ $\langle opts \rangle$ ]

---

Define a new  $\langle style \rangle$  that binds to  $\langle database \rangle$ . The style can inherit from a list of  $\langle base\ styles \rangle$  such as  $\backslash dbNewStyle[base1, base2]\{new-style\}\{ques\}\}$ .

**filter** =  $\langle filter \rangle$  (initially -none-)

Set the  $\langle filter \rangle$  defined by  $\backslash dbCombineFilters$ .

**raw-filter** =  $\langle cond\ combination \rangle$  (no value)

Set anonymous with defined conditionals. Two examples below perform the same function.

```
% method 1
\begin{dbFilters}{db}
  \dbNewConditional{cond1}{int-attr}{\rval > 1}
  \dbNewConditional*{cond2}{str-attr}{\d+}
\end{dbFilters}
\dbNewStyle{style}{db}{raw-filter={cond1 && cond2}}

% method 2
\begin{dbFilters}{db}
  \dbNewConditional{cond1}{int-attr}{\rval > 1}
  \dbNewConditional*{cond2}{str-attr}{\d+}
  \dbCombineFilters{filter}{cond1 && cond2}
\end{dbFilters}
\dbNewStyle{style}{db}{filter=filter}
```

**sort** = {  $\langle attr\ spec1 \rangle$ ,  $\langle attr\ spec2 \rangle$ , ... } (no value)

Set sorting rules. Attributes of type `str`, `date`, `int`, `fp` is supported to sort. Multi-level sort is allowed.  $\langle attr \rangle$  represents for ascending order, and  $\langle attr \rangle^*$  represents for descending

order. The example below use four fields to determine the order of the records. It sorts on `level` in descending order first and if two `levels` are same then sorts on `birth` in ascending order and so on.

```
\dbNewDatabase{sort-example}{
    name = str,
    birth = date,
    level = int,
    weight = fp,
}
\dbNewStyle{sort-style}{sort-example}{
    sort = { level*, birth, name, weight }
}
```

**before-code** = *<before code>* (no value)

Set the *<before code>* that is executed before displaying the database.

**after-code** = *<after code>* (no value)

Set the *<after code>* that is executed after displaying the database.

**item-code** = *<item code>* (no value)

Set the code that show a single record. You can use `\dbuse` to display certian attribute.

**<attr>/before-code** = *<before code>* (no value)

Set the *<before code>* that is executed by `\dbuse` before displaying certain attribute.

**<attr>/after-code** = *<after code>* (no value)

Set the *<after code>* that is executed by `\dbuse` after displaying certain attribute.

**<attr>/sep** = *<sep spec>* (initially ,~)

**<attr>/separator** = *<separator>*

```
<attr>/separator = {
    <separator between two>,
    <separator between more than two>,
    <separator between final two>
}
```

Only for attributes of type `clist`. Set the separator between clist items. The first form accept one arguments and set the seperator as *<sep>*. The second form is more complicated, the following documentation is quoted from `interface3`:

If the comma list has more than two items, the *<separator between more than two>* is placed between each pair of items except the last, for which the *<separator between final two>* is used. If the comma list has exactly two items, then they are placed in the input stream separated by the *<separator between two>*. If the comma list has a single item, it is placed in the input stream, and a comma list with no items produces no output.

**item-before-code** = *<before code>* (no value)

Only for attributes of type `clist`. Set the *<after code>* that is excuted before displaying the item of the comma list.

**item-after-code** =  $\langle \text{after code} \rangle$  (no value)

Only for attributes of type `clist`. Set the  $\langle \text{after code} \rangle$  that is executed after displaying the item of the comma list.

### 5.3 Use `\dbNewReviewPoints` to Define Review Points

---

**\dbNewReviewPoints**    `\dbNewReviewPoints { $\langle \text{name} \rangle$ } { $\langle \text{points} \rangle$ }`

---

Define the new  $\langle \text{points} \rangle$  that is specially designed for reviewing something.  $\langle \text{points} \rangle$  is a list of integers. Suppose you record the date when you did not answer correctly and you plan to review every 2, 5 and 15 days. The following code give what you want.

```
\dbNewReviewPoints{review-point}{2, 5, 15}           % define points
\begin{dbFilters}
  \dbNewConditional{cond1}{date}{review-point|\Today} % define conditional
  \dbCombineConditionals{filter1}{cond1}              % define filter
\end{dbFilters}
\dbNewStyle{review-style}{ques}{filter=filter1}      % define style
```

### 5.4 Define Filters inside `dbFilters` Environment

**dbFilters**    `\begin{dbFilters}{ $\langle \text{database} \rangle$ }`  
                $\langle \text{code} \rangle$   
               `\end{dbFilters}`

Filters are defined inside `dbFilters` environment, inside which, `\dbNewConditional` is defined to declare conditionals and `\dbCombineConditionals` is defined to combine conditionals. Filters are independent in different databases, which means the same name of filters is allowed in different databases.

---

**\dbNewConditional**    `\dbNewConditional { $\langle \text{name} \rangle$ }            { $\langle \text{attr} \rangle$ } { $\langle \text{cond spec} \rangle$ }`  
**\dbNewConditional\*** `\dbNewConditional* { $\langle \text{name} \rangle$ }            { $\langle \text{attr} \rangle$ } { $\langle \text{cond spec} \rangle$ }`

---

`\dbNewConditional { $\langle \text{name} \rangle$ } { $\langle \text{int/fp attr} \rangle$ } { $\langle \text{relation} \rangle$ }`  
`\dbNewConditional* { $\langle \text{name} \rangle$ } { $\langle \text{int/fp attr} \rangle$ } { $\langle \text{relation} \rangle$ }`  
`\dbNewConditional { $\langle \text{name} \rangle$ } { $\langle \text{str/tl attr} \rangle$ } { $\langle \text{regex expr} \rangle$ }`  
`\dbNewConditional* { $\langle \text{name} \rangle$ } { $\langle \text{str/tl attr} \rangle$ } { $\langle \text{regex expr} \rangle$ }`  
`\dbNewConditional { $\langle \text{name} \rangle$ } { $\langle \text{clist attr} \rangle$ } { $\langle \text{val list} \rangle$ }`  
`\dbNewConditional* { $\langle \text{name} \rangle$ } { $\langle \text{clist attr} \rangle$ } { $\langle \text{val list} \rangle$ }`  
`\dbNewConditional { $\langle \text{name} \rangle$ } { $\langle \text{date attr} \rangle$ } { $\langle \text{review points} \rangle$ | $\langle \text{date} \rangle$ }`  
`\dbNewConditional* { $\langle \text{name} \rangle$ } { $\langle \text{date attr} \rangle$ } { $\langle \text{relation} \rangle$ }`

Define the conditional named  $\langle \text{name} \rangle$  that binds to  $\langle \text{attr} \rangle$ . `\dbval` is replaced with the real value of the attribute inside the  $\langle \text{cond spec} \rangle$ .

For attributes of type `int` and `fp`, two forms have the same definition.  $\langle \text{relation} \rangle$  can be a single relation formula, such as `\dbval > 3`, or the combination of several relation formulas, such as `\dbval > 3 && \dbval < 10.2`. Supported operators are `<`, `>`, `=`, `==`, `!=`, `>=`, `<=`, `!`.

For attribute of type `str` and `tl`, unstarred form matches any part while starred form matches the whole part with the  $\langle \text{regex expr} \rangle$ .

```
\dbNewConditional{cond1}{str-attr}{abc} % match abc, abcd, 1abc, =abc=, etc
\dbNewConditional*{cond2}{str-attr}{abc} % only match abc
```

For attributes of type `clist`, the conditional defined by unstarred form is true if any item of  $\langle val\ list \rangle$  is in the comma list. While the conditional defined by starred form is true only if every item of  $\langle val\ list \rangle$  is in the comma list. As is showed below, for `cond1`, `a` is in  $\{a, b, d\}$  so `cond1` is true. While `c` is not in  $\{a, b, d\}$  so `cond2` is false.

```
\dbNewConditional{cond1}{clist-attr}{a, b, c} % {a, b, d} -> true
\dbNewConditional*{cond2}{clist-attr}{a, b, c} % {a, b, d} -> false
```

For attributes of type `date`, unstarred form uses  $\langle review\ points \rangle$  to define the conditional and  $\langle date \rangle$  is the date to be compared. The starred form define the conditional with **single** relation formula. Supported operators are `<`, `>`, `=`, `==`, `!=`, `>=`, `<=`.

---

### `\dbCombineConditionals`

---

```
\dbCombineConditionals { $\langle name \rangle$ } { $\langle cond\ combination \rangle$ } [ $\langle info \rangle$ ]
```

Define the filter  $\langle name \rangle$ , which combine the conditionals and store the extra  $\langle info \rangle$  into `\dbFilterInfo`. So you can write something as

```
\dbCombineConditionals{filter}{(cond1 && cond2) || !cond3}.
```

Supported operators are `&&`, `||`, `!`. You can set the option `filter` to  $\langle name \rangle$  to apply the filter when you display the database.

## 5.5 Store Data with `dbitem` Environment

```
dbitem \begin{dbitem}{ $\langle database \rangle$ }[
     $\langle attr1 \rangle$  =  $\langle val1 \rangle$ ,
     $\langle attr2 \rangle$  =  $\langle val2 \rangle$ ,
    ...
]
 $\langle code \rangle$ 
\end{dbitem}
```

The data are stored with `dbitem` environment in two ways. Smaller data can be stored in the option list and the bigger data can be stored by `\dbsave`, which will suppress the value set by the option list. An example code is showned below.

```
\begin{dbitem}[date=2022-01-01, info=test]
    \dbsave{question}{This is a test question.}
    \dbsave{answer}{This is the correct answer of the question.}
\end{dbitem}
```

## 5.6 `\dbsave` and `\dbuse`

---

```
\dbsave \dbsave { $\langle attr \rangle$ } { $\langle data \rangle$ }
\dbuse \dbuse { $\langle attr \rangle$ }

```

---

Date is stored with `\dbsave` and is displayed with `\dbuse`. `\dbsave` can be used only inside the `dbitem` environment and `\dbuse` can be only used inside the option `item-code`.

## 5.7 Use \dbshow to Display the Database

---

<code>\dbshow</code>	<code>\dbshow {&lt;style&gt;} {&lt;database&gt;}</code>
----------------------	---

---

Show the `<database>` with `<style>`.

## 5.8 Conditionals

---

<code>\dbIfEmptyT</code>	<code>\dbIfEmptyTF {&lt;true code&gt;} {&lt;false code&gt;}</code>
<code>\dbIfEmptyF</code>	<code>\dbIfEmptyT {&lt;true code&gt;}</code>
<code>\dbIfEmptyTF</code>	<code>\dbIfEmptyF {&lt;false code&gt;}</code>

---

Test if the database is empty. The example below shows how to avoid an empty list environment.

```
\dbNewStyle{style-cond1}{database-test}{
  before-code = {\dbIfEmptyF{\begin{enumerate}}},
  after-code = {\dbIfEmptyF{\end{enumerate}}},
  item-code = {\item \dbuse{attr-test}}
}
```

---

<code>\dbItemIfEmptyT</code>	<code>\dbItemIfEmptyTF {&lt;true code&gt;} {&lt;false code&gt;}</code>
<code>\dbItemIfEmptyF</code>	<code>\dbItemIfEmptyT {&lt;true code&gt;}</code>
<code>\dbItemIfEmptyTF</code>	<code>\dbItemIfEmptyF {&lt;false code&gt;}</code>

---

Test if the value of the attribute is empty. The example belows shows how to surround the non-empty `text` attribute with the symbol `*`.

```
\dbNewStyle{style-cond2}{database-test}{
  text/before-code = {\dbItemIfEmptyF{*}},
  text/after-code = {\dbItemIfEmptyF{*}},
}
```

---

<code>\dbClistItemIfEmptyT</code>	<code>\dbClistItemIfEmptyTF {&lt;true code&gt;} {&lt;false code&gt;}</code>
<code>\dbClistItemIfEmptyF</code>	<code>\dbClistItemIfEmptyT {&lt;true code&gt;}</code>
<code>\dbClistItemIfEmptyTF</code>	<code>\dbClistItemIfEmptyF {&lt;false code&gt;}</code>

---

Test if the item of comma list is empty. The example belows shows how to surround the non-empty label with the symbol `*`.

```
\dbNewDatabase{database-test}{labels=clist}
\dbNewStyle{style-cond3}{database-test}{
  labels/item-before-code = {\dbClistItemIfEmptyF{*}},
  labels/item-after-code = {\dbClistItemIfEmptyF{*}},
}
```

## 5.9 Special Macros

Some special macros are defined to expand to different contents according to context.

<code>\dbval</code>	Attribute value, only accessible in <code>\dbNewConditional</code> .
<code>\dbDatabase</code>	Database name.
<code>\dbFilterName</code>	Filter name.
<code>\dbFilterInfo</code>	Filter information.
<code>\dbIndex</code>	Index of original records, which is equal to <code>\rvuse{id}</code> but <code>\rvuse</code> is nonexpandable.
<code>\dbarabic</code>	Show the counter of query set as digits.
<code>\dbalph</code>	Show the counter of query set as lowercase letters.
<code>\dbAlph</code>	Show the counter of query set as uppercase letters.
<code>\dbroman</code>	Show the counter of query set as roman numerals.
<code>\dbRoman</code>	Show the counter of query set as Roman numerals.

## 6 Example of Flaw Sweeper Template

```
\documentclass{article}
\usepackage{amsmath, physics}
\usepackage{geometry}
\usepackage{dbshow}
\usepackage{tikz}
\usepackage{tcolorbox}
\tcbuselibrary{skins}
\usetikzlibrary{shadings}
\usepackage[hidelinks]{hyperref}

\geometry{
  margin=2cm
}

% #1 link node #2 target node #3 text to show
\NewDocumentCommand \linktarget { m m m } {%
  \hyperlink{#1}{#3}%
  \raisebox{1em}{\hypertarget{#2}{}}%
}

% question box
\tcbset{
  base/.style={
    empty,
    frame engine=path,
    colframe=yellow!10,
    coltitle=red!70,
    fonttitle=\bfseries\sffamily,
    sharp corners,
    left=4pt,
```

```

        right=4pt,
        drop fuzzy shadow,
        drop fuzzy shadow,
        borderline west={3pt}{-3pt}{red!80},
    }
}

\newtcolorbox{mybox}[1]{%
    base, title = {#1}
}

\dbNewReviewPoints{review}{1, 3, 7, 15, 30, 60}

\dbNewDatabase{ques-book}{
    ques = tl,
    answer = tl,
    count = int|1,
    labels = clist,
    date = date,
}

\begin{dbFilters}{ques-book}
    \dbNewConditional{hard}{labels}{hard}
    \dbNewConditional{bad}{count}{\dbval > 1}
    \dbNewConditional{review}{date}{review|2022-01-07}
\end{dbFilters}

% show all questions with hyperlink to answers
\dbNewStyle{ques}{ques-book}{
    before-code = {\section{Questions}},
    item-code = {
        \begin{mybox}{%
            \linktarget{answer_\dbIndex}{ques_\dbIndex}{%
                Question \dbarabic%
                \hspace{2em}\dbuse{date}%
                \hspace{2em}\dbuse{labels}%
                \hfill\dbuse{count}%
            }%
        }
        \dbuse{ques}%
    }
    \end{mybox}
},
    labels/sep = /,
}

% show all questions and answers with hyperlink to questions
\dbNewStyle{answer}{ques-book}{
    before-code = {\section{Questions and Answers}},
    item-code = {
        \begin{mybox}{%
            \linktarget{ques_\dbIndex}{answer_\dbIndex}{%

```

```

        Question \dbarabic%
        \hspace{2em}\dbuse{date}%
        \hspace{2em}\dbuse{labels}%
        \hfill\dbuse{count}%
    }%
}
    \dbuse{ques}\tcbsubtitle{Answer}\dbuse{answer}%
    \end{mybox}
},
labels/sep = /,
}

% show all hard questions with hyperlink to answers
\dbNewStyle{hard}{ques-book}{
    before-code = {\section{Hard Questions}},
    item-code = {
        \begin{mybox}{%
            \hyperlink{answer_\dbIndex}{%
                Question \dbarabic%
                \hspace{2em}\dbuse{date}%
                \hspace{2em}\dbuse{labels}%
                \hfill\dbuse{count}%
            }%
        }
        \dbuse{ques}%
        \end{mybox}
    },
    raw-filter = hard,
    labels/sep = /,
}

% show all hard questions that have answered incorrectly for more than one
% time with hyperlink to answers
\dbNewStyle[hard]{bad}{ques-book}{
    before-code = {\section{Bad Questions}},
    raw-filter = {bad && hard},
}

% show all questions that plan to be reviewed on 2022-01-07 with hyperlink to
% answers
\dbNewStyle[hard]{review}{ques-book}{
    before-code = {\section{Questions to be Reviewed}},
    raw-filter = {review},
}

\AtEndDocument{
    \dbshow{review}{ques-book}
    \dbshow{hard}{ques-book}
    \dbshow{bad}{ques-book}
    \dbshow{ques}{ques-book}
    \dbshow{answer}{ques-book}
}

```

```

\begin{document}

\begin{dbitem}{ques-book}[
  date=2022-01-01,
  labels={math, equation, easy},
  count=2
]
\dsave{ques}{%
  Solve the linear equation:  $x + 16 = 31$ .
}
\dsave{answer}{%
   $x = 31 - 16 = 15$ 
}
\end{dbitem}

\begin{dbitem}{ques-book}[
  date=2022-01-01,
  labels={math, equation, hard},
  count=3
]
\dsave{ques}{%
  Solve the linear equation:  $2y = 16$ .
}
\dsave{answer}{%
   $y = 16 / 2 = 8$ 
}
\end{dbitem}

\begin{dbitem}{ques-book}[
  date=2022-01-04,
  labels={math, integral, hard},
  count=1
]
\dsave{ques}{%
  Find the indefinite integral:  $\int 2x \, dx$ .
}
\dsave{answer}{%
   $\int 2x \, dx = x^2$ 
}
\end{dbitem}

\end{document}

```

<p>1 Questions to be Reviewed</p> <p>Question 1 2022-01-04 math/integral/hard 1</p> <p>Find the indefinite integral: <math>\int 2x dx</math>.</p> <p>2 Hard Questions</p> <p>Question 1 2022-01-01 math/equation/hard 3</p> <p>Solve the linear equation: <math>2y = 16</math>.</p> <p>Question 2 2022-01-04 math/integral/hard 1</p> <p>Find the indefinite integral: <math>\int 2x dx</math>.</p> <p>3 Bad Questions</p> <p>Question 1 2022-01-01 math/equation/hard 3</p> <p>Solve the linear equation: <math>2y = 16</math>.</p> <p>4 Questions</p> <p>Question 1 2022-01-01 math/equation/easy 2</p> <p>Solve the linear equation: <math>x + 16 = 31</math>.</p> <p>Question 2 2022-01-01 math/equation/hard 3</p> <p>Solve the linear equation: <math>2y = 16</math>.</p> <p>Question 3 2022-01-04 math/integral/hard 1</p> <p>Find the indefinite integral: <math>\int 2x dx</math>.</p> <p>5 Questions and Answers</p> <p>Question 1 2022-01-01 math/equation/easy 2</p> <p>Solve the linear equation: <math>x + 16 = 31</math>.</p> <p>Answer</p> <p><math>x = 31 - 16 = 15</math></p> <p>1</p>	<p>Question 2 2022-01-01 math/equation/hard 3</p> <p>Solve the linear equation: <math>2y = 16</math>.</p> <p>Answer</p> <p><math>y = 16/2 = 8</math></p> <p>Question 3 2022-01-04 math/integral/hard 1</p> <p>Find the indefinite integral: <math>\int 2x dx</math>.</p> <p>Answer</p> <p><math>\int 2x dx = x^2</math></p> <p>2</p>
--	--

# Change History

Date	Ver.	Describe
2022-01-05	1.0	First version Add macro: <code>\dbarabic</code> , <code>\dbalph</code> , <code>\dbAlpha</code> , <code>\dbroman</code> , <code>\dbRoman</code>
2022-01-06		Fix bug: <code>\dbIndex</code> undefined Add style option: <code>raw-filter</code>
2022-01-07	1.1	Improve example

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<code>&lt;attr&gt;/after-code</code> (option) . . . . .	<i>4, 11</i>
<code>&lt;attr&gt;/before-code</code> (option) . . . . .	<i>4, 11</i>
<code>&lt;attr&gt;/sep</code> (option) . . . . .	<i>4, 11</i>
<b>A</b>	
<code>after-code</code> (option) . . . . .	<i>4, 11</i>
<b>B</b>	
<code>before-code</code> (option) . . . . .	<i>4, 11</i>
<b>D</b>	
<code>\dbAlpha</code> . . . . .	<i>8, 15</i>
<code>\dbAlph</code> . . . . .	<i>19</i>
<code>\dbalph</code> . . . . .	<i>8, 15</i>
<code>\dbalph</code> . . . . .	<i>19</i>
<code>\dbarabic</code> . . . . .	<i>8, 15</i>
<code>\dbarabic</code> . . . . .	<i>19</i>
<code>\dbClistItemIfEmptyF</code> . . . . .	<i>7, 14</i>
<code>\dbClistItemIfEmptyT</code> . . . . .	<i>7, 14</i>
<code>\dbClistItemIfEmptyTF</code> . . . . .	<i>7, 14</i>
<code>\dbCombineConditionals</code> . . . . .	<i>5, 6, 12, 13</i>
<code>\dbCombineFilters</code> . . . . .	<i>3, 10</i>
<code>\dbDatabase</code> . . . . .	<i>7, 15</i>
<code>\dbFilterInfo</code> . . . . .	<i>7, 15</i>
<code>\dbFilterInfo</code> . . . . .	<i>6, 13</i>
<code>\dbFilterName</code> . . . . .	<i>7, 15</i>
<code>dbFilters</code> (environment) . . . . .	<i>5, 12</i>
<code>\dbIfEmptyF</code> . . . . .	<i>7, 14</i>
<code>\dbIfEmptyT</code> . . . . .	<i>7, 14</i>
<code>\dbIfEmptyTF</code> . . . . .	<i>7, 14</i>
<code>\dbIndex</code> . . . . .	<i>7, 15</i>
<code>\dbIndex</code> . . . . .	<i>19</i>
<code>dbitem</code> (environment) . . . . .	<i>6, 13</i>
<code>\dbItemIfEmptyF</code> . . . . .	<i>7, 14</i>
<code>\dbItemIfEmptyT</code> . . . . .	<i>7, 14</i>
<code>\dbItemIfEmptyTF</code> . . . . .	<i>7, 14</i>
<code>\dbNewConditional</code> . . . . .	<i>5–7, 12, 15</i>
<code>\dbNewConditional*</code> . . . . .	<i>5, 6, 12</i>
<code>\dbNewDatabase</code> . . . . .	<i>2, 9</i>
<code>\dbNewDatabase*</code> . . . . .	<i>2, 9</i>
<code>\dbNewReviewPoints</code> . . . . .	<i>5, 6, 12</i>
<code>\dbNewStyle</code> . . . . .	<i>3, 10</i>
<code>\dbRoman</code> . . . . .	<i>8, 15</i>
<code>\dbRoman</code> . . . . .	<i>19</i>
<code>\dbroman</code> . . . . .	<i>8, 15</i>
<code>\dbroman</code> . . . . .	<i>19</i>
<code>\dbsave</code> . . . . .	<i>6, 13</i>
<code>\dbshow</code> . . . . .	<i>6, 14</i>
<code>\dbuse</code> . . . . .	<i>4, 6, 7, 11, 13</i>
<code>\dbval</code> . . . . .	<i>7, 15</i>
<code>\dbval</code> . . . . .	<i>5, 12</i>
<b>E</b>	
environments:	
<code>dbFilters</code> . . . . .	<i>5, 12</i>
<code>dbitem</code> . . . . .	<i>6, 13</i>
<b>F</b>	
<code>filter</code> (option) . . . . .	<i>3, 10</i>
<code>filter</code> . . . . .	<i>6, 13</i>
<b>I</b>	
<code>item-after-code</code> (option) . . . . .	<i>4, 12</i>
<code>item-before-code</code> (option) . . . . .	<i>4, 11</i>
<code>item-code</code> (option) . . . . .	<i>4, 11</i>
<code>item-code</code> . . . . .	<i>6, 13</i>
<b>O</b>	
options:	
<code>&lt;attr&gt;/after-code</code> . . . . .	<i>4, 11</i>
<code>&lt;attr&gt;/before-code</code> . . . . .	<i>4, 11</i>
<code>&lt;attr&gt;/sep</code> . . . . .	<i>4, 11</i>
<code>after-code</code> . . . . .	<i>4, 11</i>
<code>before-code</code> . . . . .	<i>4, 11</i>
<code>filter</code> . . . . .	<i>3, 10</i>
<code>item-after-code</code> . . . . .	<i>4, 12</i>
<code>item-before-code</code> . . . . .	<i>4, 11</i>
<code>item-code</code> . . . . .	<i>4, 11</i>
<code>raw-filter</code> . . . . .	<i>3, 10</i>
<code>sort</code> . . . . .	<i>3, 10</i>
<b>R</b>	
<code>raw-filter</code> (option) . . . . .	<i>3, 10</i>
<code>raw-filter</code> . . . . .	<i>19</i>
<code>\rvuse</code> . . . . .	<i>15</i>
<b>S</b>	
<code>sort</code> (option) . . . . .	<i>3, 10</i>
<b>T</b>	
<code>\Today</code> . . . . .	<i>2, 9</i>