修改@column 解析，支持字符串传参

目录

修改的主要类是 AbstractSqlConfig 中的 getColumnString

1 支持了单引号字符串。

添加了支持解析单引号，并加了字符串正则：
PATTERN_STRING = Pattern.compile("^[,#;\"`]+$");防止 sql 注入
下面是列举常见难以解析的格式的函数 ， 带有字符串的:

```
1. CONCAT(s1, s2...sn)    字符串 s1,s2 等多个字符串合并为一个字符串
2. CONCAT_WS(x, s1, s2...sn)   同 CONCAT(s1, s2 ...) 函数，但是每个字符串之间要加上 x，x 可以是分隔符
3.  INSERT(s1, x, len, s2) 字符串 s2 替换 s1 的 x 位置开始长度为 len 的字符串
4. FIELD(s, s1, s2...) 返回第一个字符串 s 在字符串列表 (s1, s2...)中的位置
5. LPAD(s1, len, s2)    在字符串 s1 的开始处填充字符串 s2，使字符串长度达到 len
6. STRCMP(s1, s2) 比较字符串 s1 和 s2，如果 s1 与 s2 相等返回 0 ，如果 s1>s2 返回 1，如果 s1<s2 返回 -1
7. toUnixTimestamp
8. CONCAT_WS(x, s1, s2...sn)   同 CONCAT(s1, s2 ...) 函数，但是每个字符串之间要加上 x，x 可以是分隔符
9. STRCMP(s1, s2)  比较字符串 s1 和 s2，如果 s1 与 s2 相等返回 0 ，如果 s1>s2 返回 1，如果 s1<s2 返回 -1
10. CAST (expression AS data_type) cast 函数
11.  POSITION(s, s1);   从字符串 s 中获取 s1 的开始位置
12. LPAD(s1, len, s2)   在字符串 s1 的开始处填充字符串 s2，使字符串长度达到 len
13.  EXTRACT(type FROM d)   从日期 d 中获取指定的值，type 指定返回的值
14. mysql 和 clickhosue 中大部分的 json 函数都是两个字符串以上的参数。
```

一般来说，函数只需要传入一个字符串。而且这个字符串一般是字段，直接写入函数就行了。但遇到这样的情况，一个函数需要传入两个字符串的，或者需要从前端传入字符串的，这种

情况就需要将一个字符串传入@column 中交给 APIJSON 解析，但是 APIJSON 中会将字符串解析为字段，没有去支持传入一个字符串这样的情况 。
于是就有了这样的一个需求： 修改@column ，使之支持字符串的传入。具体代码见 6,、

2 修改了 isNumber 正则，

因为一些函数中
mysql ceil（'23.321'）可以
clickhosue ceil('4324')不可以
只要字段满足下面两个 就可以

```
PATTERN_INTEGER = Pattern.compile("^-?\\d+$");
PATTERN_FLOAT = Pattern.compile("^(-?\\d+)(\\.\\d+)?$");
```

3.添加了解析以空格分隔 的参数

比如 下面这样的参数:

```
1.  cast(timestamp AS DateTime):datetime
2.  cast(now() AS DATETIME)
3.  timestamp_add(date, INTERVAL 1 MONTH)
4.  rank() OVER (PARTITION BY name ORDER BY id DESC):ranking
5.  toDate(now())
6.  count(DISTINCT id,name)
```

以上举出了一些以空格分隔的参数，APIJSON 原来的情况下会把”timestamp as DateTime” 看做一个整体 ，比如第一行的样例，会解析成：

```
7.  cast(`timestamp AS DateTime`):datetime
```

把函数内部的参数当做了一个字段，而我们希望她能够解析成：

```
1.  cast(`timestamp` AS DateTime):datetime
```

所以，有了这样的 一个需求 ，不仅需要解析字段，还需要解析参数中以空格分隔的参数 ，把以空格分隔的参数一一解析出来。
类似： now() as date ，把相关的关键字都加入 RAW_MAP 中

4. 修改了 PATTERN_FUNCTION 正则 ，加了括号匹配

5. 备注：

方法时间复杂度 和原来一样。
主要添加了一些功能的代码，把主要功能部分提取了出来，其他的代码大部分没变

疑问？
1.所有的关键字都统一大写吗？　　统一
2.可以 DISTINCT 加入 RAW_MAP 中，不用再单独判断 ，可以

6.主要 修改代码

```
1.  /**
2.           * 解析@column 中以“;”分隔的表达式（"@column":"expression1;expression2;expression2;...."）中的 expression
3.           *
4.           * @param expression
5.           * @return
6.           */
7.          public String getColumnPrase(String expression) {
8.                  String quote = getQuote();
9.                  int start = expression.indexOf('(');
10.                 if (start < 0) {
11.                         //没有函数 ,可能是字段,也可能是 DISTINCT
12.                         String cks[] = parseArgsSplitWithComma(expression, true);
13.                         expression = StringUtil.getString(cks);
14.                 } else {
15.                         //有函数,但不是窗口函数
16.                         if (expression.indexOf("OVER") < 0) {
17.                                 int end = expression.lastIndexOf(")");
18.                                 if (start >= end) {
19.                                         throw new IllegalArgumentException("字符 " + expression + " 不合法！"
20.                                                         + "@column:value 中 value 里的 SQL 函数必须为 function(arg0,arg1,...) 这种格式！");
21.                                 }
22.                                 String fun = expression.substring(0, start);
```

```
23.                             if (fun.isEmpty() == false) {
24.                                 if (FunctionsAndRaws.SQL_FUNCTION_MAP == null || FunctionsAndRaws.SQL_FUNCTION_MAP.isEmpty()) {
25.                                     if (StringUtil.isName(fun) == false) {
26.                                         throw new IllegalArgumentException("字符 " + fun + " 不合法!"
27.                                                 + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
28.                                                 + " 中 function 必须符合小写英文单词的 SQL 函数名格式!");
29.                                     }
30.                                 } else if (FunctionsAndRaws.SQL_FUNCTION_MAP.containsKey(fun) == false) {
31.                                     throw new IllegalArgumentException("字符 " + fun + " 不合法!"
32.                                             + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
33.                                             + " 中 function 必须符合小写英文单词的 SQL 函数名格式!且必须是后端允许调用的 SQL 函数!");
34.                                 }
35.                             }
36.
37.                             String s = expression.substring(start + 1, end);
38.                             // 解析函数内的参数
39.                             String ckeys[] = parseArgsSplitWithComma(s, false);
40.
41.                             String suffix = expression.substring(end + 1, expression.length()); //:contactCount
42.                             int index = suffix.lastIndexOf(":");
43.                             String alias = index < 0 ? "" : suffix.substring(index + 1); //contactCount
44.                             suffix = index < 0 ? suffix : suffix.substring(0, index);

45.                             if (alias.isEmpty() == false && StringUtil.isName(alias) == false) {
46.                                 throw new IllegalArgumentException("字符串 " + alias + " 不合法!"
47.                                         + "预编译模式下 @column:value 中 value 里面用 ; 分割的每一项"
48.                                         + " function(arg0,arg1,...):alias 中 alias 必须是 1 个单词! 并且不要有多余的空格!");
49.                             }
50.
51.
```

```
52.                             if (suffix.isEmpty() == false && (((String) suffix).contains("--
") || ((String) suffix).contains("/*") || PATTERN_RANGE.matcher((String) suffix).matches() ==
 false)) {
53.                                 throw new UnsupportedOperationException("字符
串 " + suffix + " 不合法！"
54.                                         + "预编译模式
下 @column:\"column?value;function(arg0,arg1,...)?value...\""
55.                                         + " 中 ?value 必须符合正则表达
式 " + PATTERN_RANGE + " 且不包含连续减号 -- 或注释符 /* ！不允许多余的空格！");
56.                             }
57.
58.                             String origin = fun + "(" + StringUtil.getString(ckeys) + ")
" + suffix;
59.                             expression = origin + (StringUtil.isEmpty(alias, true) ? "" :
 " AS " + quote + alias + quote);
60.
61.                         } else {
62.                             //是窗口函数      fun(arg0,agr1) OVER (agr0 agr1 ...)
63.                             int overindex = expression.indexOf("OVER"); // OVER 的位置
64.                             String s1 = expression.substring(0, overindex); // OVER 前半部
分
65.                             String s2 = expression.substring(overindex); // OVER 后半部
分
66.
67.                             int index1 = s1.indexOf("("); //    函数 "(" 的起始位置
68.                             String fun = s1.substring(0, index1); // 函数名称
69.                             int end = s2.lastIndexOf(")"); // 后半部分 ")" 的位置
70.
71.                             if (index1 >= end) {
72.                                 throw new IllegalArgumentException("字
符 " + expression + " 不合法！"
73.                                         + "@column:value 中 value 里的 SQL 函数
必须为 function(arg0,arg1,...) 这种格式！");
74.                             }
75.                             if (fun.isEmpty() == false) {
76.                                 if (FunctionsAndRaws.SQL_FUNCTION_MAP == null || Functio
nsAndRaws.SQL_FUNCTION_MAP.isEmpty()) {
77.                                     if (StringUtil.isName(fun) == false) {
78.                                         throw new IllegalArgumentException("字
符 " + fun + " 不合法！"
79.                                                 + "预编译模式
下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
```

```
                                                    + " 中 function 必须符合小写英文单词的 SQL 函数名格式！");
                                            }
                                    } else if (FunctionsAndRaws.SQL_FUNCTION_MAP.containsKey(fun) == false) {
                                            throw new IllegalArgumentException("字符 " + fun + " 不合法！"
                                                    + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
                                                    + " 中 function 必须符合小写英文单词的 SQL 函数名格式！且必须是后端允许调用的 SQL 函数!");
                                    }
                            }

                            // 获取前半部分函数的参数解析 fun(arg0,agr1)
                            String agrsString1[] = parseArgsSplitWithComma(s1.substring(index1 + 1, s1.lastIndexOf(")")), false);


                            int index2 = s2.indexOf("("); // 后半部分 “(”的起始位置
                            String argString2 = s2.substring(index2 + 1, end); // 后半部分的参数
                            // 别名
                            String alias = s2.lastIndexOf(":") < 0 ? null : s2.substring(s2.lastIndexOf(":") + 1);
                            // 获取后半部分的参数解析 (agr0 agr1 ...)
                            String argsString2[] = parseArgsSplitWithComma(argString2,false);

                            expression = fun + "(" + StringUtil.getString(agrsString1) + ")" + " OVER " + "(" + StringUtil.getString(argsString2) + ")" + (StringUtil.isEmpty(alias, true) ? "" : " AS " + quote + alias + quote);
                        }
                }
                return expression;

        }

        /**
         * 解析函数参数或者字段,此函数对于解析字段 和 函数内参数通用
         *
         * @param param
         * @param isColumn true:不是函数参数。false:是函数参数
         * @return
         */
```

```
113.        private String[] parseArgsSplitWithComma(String param, boolean isColumn) {
114.                // 以"," 分割参数
115.                String quote = getQuote();
116.                String tableAlias = getAliasWithQuote();
117.                String ckeys[] = StringUtil.split(param); // 以","分割参数
118.                if (ckeys != null && ckeys.length > 0) {
119.                        String origin;
120.                        String alias;
121.                        int index;
122.                        for (int i = 0; i < ckeys.length; i++) {
123.                                // 如果参数包含 "'" ,解析字符串
124.                                if (ckeys[i].contains("'")) {
125.                                        int count = 0;
126.                                        for (int j = 0; j < ckeys[i].length(); j++) {
127.                                                if (ckeys[i].charAt(j) == '\'') count++;
128.                                        }
129.                                        // 排除字符串中参数中包含 ' 的情况和不以' 开头和结尾的情况，同时排除 cast('s' as ...) 以空格分隔的参数中包含字符串的情况
130.                                        if (count != 2 || !(ckeys[i].startsWith("'") && ckeys[i].endsWith("'"))) {
131.                                                throw new IllegalArgumentException("字符串 " + ckeys[i] + " 不合法！"
132.                                                        + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
133.                                                        + " 中字符串参数不合法，必须以 ' 开头, ' 结尾,字符串中不能包含 ' ");
134.                                        }
135.                                        //sql 注入判断 判断
136.                                        origin = (ckeys[i].substring(1, ckeys[i].length() - 1));
137.                                        if (origin.contains("--") || PATTERN_STRING.matcher(origin).matches() == true) {
138.                                                throw new IllegalArgumentException("字符 " + ckeys[i] + " 不合法！"
139.                                                        + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
140.                                                        + " 中所有字符串 arg 都必须不符合正则表达式 " + PATTERN_STRING + " 且不包含连续减号 -- ！");
141.                                        }
142.
143.                                        // 1.字符串不是字段也没有别名,所以不解析别名 2. 是字符串，进行预编译，使用 getValue()
144.                                        ckeys[i] = getValue(ckeys[i].substring(1, ckeys[i].length() - 1)).toString();
```

```
145.
146.                                 } else {
147.                                     // 参数不包含",",即不是字符串
148.                                     // 解析参数:1. 字段 ,2. 是以空格分隔的参数 eg: cast(now() as date)
149.                                     index = ckeys[i].lastIndexOf(":"); //StringUtil.split 返回数组中，子项不会有 null
150.                                     origin = index < 0 ? ckeys[i] : ckeys[i].substring(0, index); //获取 : 之前的
151.                                     alias = index < 0 ? null : ckeys[i].substring(index + 1);
152.                                     if (isPrepared()) {
153.                                         if (isColumn) {
154.                                             if (StringUtil.isName(origin) == false || (alias != null && StringUtil.isName(alias) == false)) {
155.                                                 throw new IllegalArgumentException("字符 " + ckeys[i] + " 不合法！"
156.                                                         + "预编译模式下 @column:value 中 value 里面用 , 分割的每一项"
157.                                                         + " column:alias 中 column 必须是 1 个单词！如果有 alias，则 alias 也必须为 1 个单词！"
158.                                                         + "关键字必须全大写，且以空格分隔的参数，空格必须只有 1 个！其它情况不允许空格！");
159.                                             }
160.                                         } else {
161.                                             if (origin.startsWith("_") || origin.contains("--") || PATTERN_FUNCTION.matcher(origin).matches() == false) {
162.                                                 throw new IllegalArgumentException("字符 " + ckeys[i] + " 不合法！"
163.                                                         + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
164.                                                         + " 中所有 arg 都必须是 1 个不以 _ 开头的单词 或者符合正则表达式 " + PATTERN_FUNCTION + " 且不包含连续减号 -- ！DISTINCT 必须全大写，且后面必须有且只有 1 个空格！其它情况不允许空格！");
165.                                             }
166.                                         }
167.                                     }
168.                                     // 以空格分割参数
169.                                     String mkes[] = StringUtil.split(ckeys[i], " ", true);
170.
171.                                     boolean isName = false;
172.                                     //如果参数中含有空格(少数情况) 比如 fun(arg1 arg2 arg3 ,arg4) 中的 arg1 arg2 arg3，比如 DISTINCT id
```

```
173.                                     if (mkes != null && mkes.length >= 2) {
174.                                         ckeys[i] = praseArgsSplitWithSpace(mkes);
175.                                     } else {
176.                                         // 如果参数没有空格
177.                                         if ("".equals(FunctionsAndRaws.RAW_MAP.get(origin)))
    {
178.                                             // do nothing , 比
    如 toDate(now()) ,
179.                                         } else if (StringUtil.isNumer(origin)) {
180.                                             //do nothing
181.                                         } else if (StringUtil.isName(origin)) {
182.                                             origin = quote + origin + quote;
183.                                             isName = true;
184.                                         } else {
185.                                             origin = getValue(origin).toString();
186.                                         }
187.                                         if (isName && isKeyPrefix()) {
188.                                             ckeys[i] = tableAlias + "." + origin;

189.                                             if (isColumn && StringUtil.isEmpty(alias,
    true) == false) {
190.                                                 ckeys[i] += " AS " + quote +
    alias + quote;
191.                                             }
192.                                         } else {
193.                                             ckeys[i] = origin + (StringUtil.isEmpty(a
    lias, true) ? "" : " AS " + quote + alias + quote);
194.                                         }
195.                                     }
196.
197.                                 }
198.                             }
199.                 }
200.                 return ckeys;
201.         }
202.
203.
204.         /**
205.          * 只解析以空格分隔的参数
206.          *
207.          * @param mkes
208.          * @return
209.          */
210.         private String praseArgsSplitWithSpace(String mkes[]) {
```

```
211.                String quote = getQuote();
212.                String tableAlias = getAliasWithQuote();
213.                boolean isName = false;
214.                String origin;
215.                // 包含空格的参数  肯定不包含别名  不用处理别名
216.                if (mkes != null && mkes.length > 0) {
217.                        for (int j = 0; j < mkes.length; j++) {
218.                                // now()/AS/ DISTINCT/VALUE 等等放在 RAW_MAP 中
219.                                if ("".equals(FunctionsAndRaws.RAW_MAP.get(mkes[j]))) {
220.                                        continue;
221.                                } else if (StringUtil.isNumer(mkes[j])) {
222.                                        // do nothing
223.                                } else {
224.                                        //这里为什么还要做一次判断  是因为解析窗口函数调用的时候会判断一次
225.                                        if (isPrepared()) {
226.                                                if (mkes[j].startsWith("_") || mkes[j].contains("--") || PATTERN_FUNCTION.matcher(mkes[j]).matches() == false) {
227.                                                        throw new IllegalArgumentException("字符 " + mkes[j] + " 不合法！"
228.                                                                + "预编译模式下 @column:\"column0,column1:alias;function0(arg0,arg1,...);function1(...):alias...\""
229.                                                                + " 中所有 arg 都必须是 1 个不以 _ 开头的单词 或者符合正则表达式 " + PATTERN_FUNCTION + " 且不包含连续减号 -- ！DISTINCT 必须全大写，且后面必须有且只有 1 个空格！其它情况不允许空格！");
230.                                                }
231.                                        }
232.                                        mkes[j] = quote + mkes[j] + quote;
233.                                        isName = true;
234.                                }
235.                                if (isName && isKeyPrefix()) {
236.                                        mkes[j] = tableAlias + "." + mkes[j];
237.                                }
238.                        }
239.                }
240.                // 返回重新以" "拼接后的参数
241.                return StringUtil.join(mkes, " ");
242.        }
```

7. 在 RAW_MAP 中添加的代码

```
1.  RAW_MAP = new LinkedHashMap<>();    // 保证顺序，避免配置冲突等意外情况
```

```
2.
3.    //  必 mysql 关键字
4.    RAW_MAP.put("AS","");
5.    RAW_MAP.put("VALUE","");
6.    RAW_MAP.put("DISTINCT","");
7.
8.    //时间
9.    RAW_MAP.put("DATE","");
10.   RAW_MAP.put("now()","");
11.   RAW_MAP.put("DATETIME","");
12.   RAW_MAP.put("DateTime","");
13.   RAW_MAP.put("SECOND","");
14.   RAW_MAP.put("MINUTE","");
15.   RAW_MAP.put("HOUR","");
16.   RAW_MAP.put("DAY","");
17.   RAW_MAP.put("WEEK","");
18.   RAW_MAP.put("MONTH","");
19.   RAW_MAP.put("QUARTER","");
20.   RAW_MAP.put("YEAR","");
21.   RAW_MAP.put("json","");
22.   RAW_MAP.put("unit","");
23.
24.   //MYSQL  数据类型  BINARY，CHAR，DATETIME，TIME，DECIMAL，SIGNED，UNSIGNED
25.   RAW_MAP.put("BINARY","");
26.   RAW_MAP.put("SIGNED","");
27.   RAW_MAP.put("DECIMAL","");
28.   RAW_MAP.put("BINARY","");
29.   RAW_MAP.put("UNSIGNED","");
30.   RAW_MAP.put("CHAR","");
31.   RAW_MAP.put("TIME","");
32.
33.   //窗口函数关键字
34.   RAW_MAP.put("OVER","");
35.   RAW_MAP.put("INTERVAL","");
36.   RAW_MAP.put("ORDER","");
37.   RAW_MAP.put("BY","");
38.   RAW_MAP.put("PARTITION","");  //往前
39.   RAW_MAP.put("DESC","");
40.   RAW_MAP.put("ASC","");
41.   RAW_MAP.put("FOLLOWING","");//往后
42.   RAW_MAP.put("BETWEEN","");
43.   RAW_MAP.put("AND","");
44.   RAW_MAP.put("ROWS","");
```