


# QB Searched Case Expression

 Apr 06, 2024 09:02

## Supported Clients

SmartClient WebClient NGClient

## Property Summary

QBColumn	abs	Create abs(column) expression
QBSort	asc	Create an ascending sort expression
QBColumn	avg	Create an aggregate expression.
QBColumn	bit_length	Create bit_length(column) expression
QBColumn	ceil	Create ceil(column) expression
QBColumn	count	Create an aggregate expression.
QBColumn	day	Extract day from date
QBSort	desc	Create an descending sort expression
QBColumn	floor	Create floor(column) expression
QBColumn	hour	Extract hour from date
QBCondition	isNull	Compare column with null.
QBColumn	len	Create length(column) expression
QBColumn	lower	Create lower(column) expression
QBColumn	max	Create an aggregate expression.
QBColumn	min	Create an aggregate expression.
QBColumn	minute	Extract minute from date
QBColumn	month	Extract month from date
QBColumn	not	Create a negated condition.
QBTableClause	parent	Get query builder parent table clause, this may be a query or a join clause.
QBSelect	root	Get query builder parent.
QBColumn	round	Create round(column) expression
QBColumn	second	Extract second from date
QBColumn	sqrt	Create sqrt(column) expression
QBColumn	sum	Create an aggregate expression.
QBColumn	trim	Create trim(column) expression
QBColumn	upper	Create upper(column) expression
QBColumn	year	Extract year from date

## Methods Summary

QBCondition	between(value1, value2)	Compare column to a range of 2 values or other columns.
QBColumn	cast(type)	Create cast(column, type) expression
QBColumn	concat(arg)	Concatenate with value
QBColumn	divide(arg)	Divide by value
QBCondition	eq(value)	Compare column with a value or another column.
QBCondition	ge(value)	Compare column with a value or another column.
Number	getFlags()	The flags are a bit pattern consisting of 1 or more of the following bits: - JSColumn.
String	getTypeAsString()	Column type as a string
QBCondition	gt(value)	Compare column with a value or another column.
QBCondition	isin(query)	Compare column with subquery result.
QBCondition	isin(values)	Compare column with values.
QBCondition	isin(customQuery, args)	Compare column with custom query result.
QBCondition	le(value)	Compare column with a value or another column.
QBCondition	like(pattern)	Compare column with a value or another column.
QBCondition	like(pattern, escape)	Compare column with a value or another column.
QBColumn	locate(arg)	Create locate(arg) expression
QBColumn	locate(arg, start)	Create locate(arg, start) expression
QBCondition	lt(value)	Compare column with a value or another column.
QBColumn	minus(arg)	Subtract value
QBColumn	mod(arg)	Create mod(arg) expression
QBColumn	multiply(arg)	Multiply with value
QBColumn	nullif(arg)	Create nullif(arg) expression
QBColumn	plus(arg)	Add up value
QBColumn	substring(pos)	Create substring(pos) expression
QBColumn	substring(pos, len)	Create substring(pos, len) expression

## Property Details

**abs**

Create abs(column) expression

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.abs)
```

**asc**

Create an ascending sort expression

**Returns**

[QBSort](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
query.sort
.add(query.joins.orders_to_order_details.columns.quantity.asc)
.add(query.columns.companyid)
foundset.loadRecords(query)
```

**avg**

Create an aggregate expression.

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
.root.having.add(query.joins.orders_to_order_details.columns.quantity.avg.eq(1))
foundset.loadRecords(query)
```

**bit\_length**

Create bit\_length(column) expression

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.bit_length)
```

**ceil**

Create ceil(column) expression

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.ceil)
```

**count**

Create an aggregate expression.

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
.root.having.add(query.joins.orders_to_order_details.columns.quantity.count.eq(0))
foundset.loadRecords(query)
```

**day**

Extract day from date

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mydatecol.day)
```

**desc**

Create an descending sort expression

**Returns**

[QBSort](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
query.sort
.add(query.joins.orders_to_order_details.columns.quantity.desc)
.add(query.columns.companyid)
foundset.loadRecords(query)
```

**floor**

Create floor(column) expression

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.floor)
```

**hour**

Extract hour from date

---

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mydatecol.hour)
```

**isNull**

Compare column with null.

**Returns**[QBCondition](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.isNull)
```

**len**

Create length(column) expression

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.len)
```

**lower**

Create lower(column) expression

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.lower)
```

**max**

Create an aggregate expression.

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
.root.having.add(query.joins.orders_to_order_details.columns.quantity.count.max(10))
foundset.loadRecords(query)
```

**min**

Create an aggregate expression.

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
    query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
    .root.having.add(query.joins.orders_to_order_details.columns.quantity.count.min(10))
    foundset.loadRecords(query)
```

**minute**

Extract minute from date

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mydatecol.minute)
```

**month**

Extract month from date

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mydatecol.month)
```

**not**

Create a negated condition.

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.not.eq(1))
```

**parent**

Get query builder parent table clause, this may be a query or a join clause.

**Returns**[QBTableClause](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.person.createSelect();
    query.where.add(query.joins.person_to_parent.joins.person_to_parent.columns.name.eq('john'))
    foundset.loadRecords(query)
```

**root**

---

Get query builder parent.

**Returns**[QBSelect](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var subquery = datasources.db.example_data.order_details.createSelect();

    var query = datasources.db.example_data.orders.createSelect();
    query.where.add(query
        .or
            .add(query.columns.order_id.not.isin([1, 2, 3]))
            .add(query.exists(
                subquery.where.add(subquery.columns.orderid.eq(query.columns.order_id)).
root
            ))
        )

    foundset.loadRecords(query)
```

**round**

Create round(column) expression

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.round)
```

**second**

Extract second from date

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mydatecol.second)
```

**sqrt**

Create sqrt(column) expression

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.sqrt)
```

**sum**

Create an aggregate expression.

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = datasources.db.example_data.orders.createSelect();
    query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
    .root.having.add(query.joins.orders_to_order_details.columns.quantity.count.sum(10))
    foundset.loadRecords(query)
```

**trim**

Create trim(column) expression

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.trim)
```

**upper**

Create upper(column) expression

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.custname.upper)
```

**year**

Extract year from date

**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mydatecol.year)
```

**Methods Details****between(value1, value2)**

Compare column to a range of 2 values or other columns.

**Parameters**[Object](#) value1;[Object](#) value2;**Returns**[QBCondition](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.between(0, 5))
```

**cast(type)**

Create `cast(column, type)` expression

**Parameters**

[String](#) type string type, see `QUERY_COLUMN_TYPES`

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient, WebClient, NGClient

**Sample**

```
query.result.add(query.columns.mycol.cast(QUERY_COLUMN_TYPES.TYPE_INTEGER))
```

**concat(arg)**

Concatenate with value

**Parameters**

[Object](#) arg value to concatenate with

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient, WebClient, NGClient

**Sample**

```
query.result.add(query.columns.firstname.concat(' ').concat(query.columns.lastname))
```

**divide(arg)**

Divide by value

**Parameters**

[Object](#) arg nr to divide by

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient, WebClient, NGClient

**Sample**

```
query.result.add(query.columns.mycol.divide(2))
```

**eq(value)**

Compare column with a value or another column.  
Operator: equals

**Parameters**

[Object](#) value ;

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient, WebClient, NGClient



---

**Sample**

```
query.where.add(query.columns.flag.eq(1))
```

**ge(value)**

Compare column with a value or another column.  
Operator: `greaterThanOrEqual`

**Parameters**

[Object](#) value ;

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.ge(2))
```

**getFlags()**

The flags are a bit pattern consisting of 1 or more of the following bits:

- `JSColumn.UUID_COLUMN`
- `JSColumn.EXCLUDED_COLUMN`
- `JSColumn.TENANT_COLUMN`

**Returns**

[Number](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample****getTypeAsString()**

Column type as a string

**Returns**

[String](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample****gt(value)**

Compare column with a value or another column.  
Operator: `greaterThan`

**Parameters**

[Object](#) value ;

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.gt(0))
```

**isin(query)**

Compare column with subquery result.

**Parameters**

[QBPart](#) query subquery

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.isin(query2))
```

**isin(values)**

Compare column with values.

**Parameters**

[Array](#) values array of values

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.isin([1, 5, 99]))
```

**isin(customQuery, args)**

Compare column with custom query result.

**Parameters**

[String](#) customQuery custom query

[Array](#) args query arguments

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.ccy.isin("select ccycode from currencies c where c.category = " + query.getTableAlias() + ".currency_category and c.flag = ?", ['T']))
```

**le(value)**

Compare column with a value or another column.  
Operator: lessThanOrEqual

**Parameters**

[Object](#) value;

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.le(2))
```

**like(pattern)**

Compare column with a value or another column.  
Operator: like

---

**Parameters**

[String](#) pattern the string value of the pattern

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.companyname.like('Serv%'))
```

**like(pattern, escape)**

Compare column with a value or another column.  
Operator: like, with escape character

**Parameters**

[String](#) pattern the string value of the pattern

[Number](#) escape the escape char

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.companyname.like('X_%', '_'))
```

**locate(arg)**

Create locate(arg) expression

**Parameters**

[Object](#) arg string to locate

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.locate('sample'))
```

**locate(arg, start)**

Create locate(arg, start) expression

**Parameters**

[Object](#) arg string to locate

[Number](#) start start pos

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.locate('sample', 5))
```

**lt(value)**

Compare column with a value or another column.  
Operator: lessThan

---

**Parameters**

[Object](#) value ;

**Returns**

[QBCondition](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.where.add(query.columns.flag.lt(99))
```

**minus(arg)**

Subtract value

**Parameters**

[Object](#) arg nr to subtract

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.minus(2))
```

**mod(arg)**

Create mod(arg) expression

**Parameters**

[Object](#) arg mod arg

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.mod(2))
```

**multiply(arg)**

Multiply with value

**Parameters**

[Object](#) arg nr to multiply with

**Returns**

[QBColumn](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.multiply(2))
```

**nullif(arg)**

Create nullif(arg) expression

**Parameters**

[Object](#) arg object to compare

**Returns**

[QBColumn](#)

---

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.nullif('none'))
```

**plus(arg)**

Add up value

**Parameters**[Object](#) arg nr to add**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.plus(2))
```

**substring(pos)**

Create substring(pos) expression

**Parameters**[Number](#) pos;**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.substring(3))
```

**substring(pos, len)**

Create substring(pos, len) expression

**Parameters**[Number](#) pos;[Number](#) len ;**Returns**[QBColumn](#)**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
query.result.add(query.columns.mycol.substring(3, 2))
```