

Expression Syntax

XINA expressions translate to MySQL expressions, which are evaluated as a query is executed.

All expressions have a **standard form** as a JSON object, with a `type` property specifying the expression type, and additional properties as needed by that expression type.

Additionally, certain expression types may be represented using a **short form**, which is formatted as a JSON object with a single property prefixed with the `$` character.

Literals

Literal expressions represent a single, discrete value.

Null

The MySQL `NULL` value. May also be specified with the JSON `null` value.

| Property | Value |
|-------------------|---------------------|
| <code>type</code> | <code>"null"</code> |

Example (as object)

```
{ "type": "null" }
```

Example (as JSON literal)

```
null
```

Number

A numeric literal value. The value may be provided as a native JSON number, or encoded as a string. May also be provided directly as a JSON `number` value.

| Property | Value |
|--------------------|--|
| <code>type</code> | <code>"number"</code> |
| <code>value</code> | <code>number</code> or <code>string</code> |

Example (as object)

```
{
  "type" : "number",
  "value" : 123
}
```

Example (as JSON literal)

```
123
```

String

A string literal value. May also be provided directly as a JSON `string`.

| Property | Value |
|--------------------|-----------------------|
| <code>type</code> | <code>"string"</code> |
| <code>value</code> | <code>string</code> |

Example (as object)

```
{
  "type" : "string",
  "value" : "foo"
}
```

Example (as JSON literal)

```
"foo"
```

Datetime

A datetime literal value. Interpreted by the database as Unix time in milliseconds.

| Property | Value |
|--------------------|--|
| <code>type</code> | <code>"datetime"</code> or <code>"dt"</code> |
| <code>value</code> | <code>integer</code> or <code>string</code> |

If the value provided is an `integer` it must be the number of milliseconds since the Unix epoch. If the value is a `string` it must be encoded according to the following syntax, taken from the ISO8601 standard:

```

date-opt-time  = date-element ['T' [time-element] [offset]]
date-element   = std-date-element | ord-date-element | week-date-element
std-date-element = yyyy ['- MM ['- dd]]
ord-date-element = yyyy ['- DDD]
week-date-element = xxxx '-W' ww ['- e]
time-element   = HH [minute-element] | [fraction]
minute-element = ':' mm [second-element] | [fraction]
second-element = ':' ss [fraction]
fraction       = ('.' | ',') digit [digit] [digit]
offset         = 'Z' | (('+' | '-') HH [':' mm [':' ss [( '.' | ',') SSS]])

```

If the offset is not provided the timezone will be assumed to be UTC.

Supports shorthand syntax with the `$dt` property.

| Property | Value |
|-------------------|-------------------|
| <code>\$dt</code> | integer or string |

Local Datetime

A local datetime literal value.

| Property | Value |
|--------------------|--------------------------|
| <code>type</code> | "localdatetime" or "ldt" |
| <code>value</code> | string |

The value must be encoded according to the same syntax as the datetime literal, except with the offset omitted.

Supports shorthand syntax with the `$ldt` property.

| Property | Value |
|--------------------|--------|
| <code>\$ldt</code> | string |

Local Date

A local date literal value.

| Property | Value |
|--------------------|---------------------|
| <code>type</code> | "localdate" or "ld" |
| <code>value</code> | string |

The value must be encoded according to the same syntax as the `date-element` in the datetime literal.

Supports shorthand syntax with the `$ld` property.

| Property | Value |
|-------------------|--------|
| <code>\$ld</code> | string |

Local Time

A local time literal value.

| Property | Value |
|----------|---------------------|
| type | "localtime" or "lt" |
| value | string |

The value must be encoded according to the same syntax as the `time-element` in the datetime literal.

Supports shorthand syntax with the `$lt` property.

| Property | Value |
|-------------------|--------|
| <code>\$lt</code> | string |

Columns

Column expressions specify a column of a table. Although each column type has a separate full syntax, there is a shorthand syntax with the `$col` property, which infers the column type based on the content.

| Property | Value |
|--------------------|---------------|
| <code>\$col</code> | string column |

```
column      = system-column | database-column
system-column = system-table-name '.' system-parameter-name
database-column = database-path ['@' database-table-name] '.' (parameter-name | attribute-name | field-name
| blob-attribute)
blob-attribute = blob-name ':' blob-attribute-name
```

Examples

- `user.email` : `email` parameter of the `user` system table
- `a.b.record_id` : `record_id` attribute of the `record` table of database `b` in group `a`
- `a.b@trash.record_id` : `record_id` attribute of the `trash` table of database `b` in group `a`
- `a.b.c` : `c` field of the `record` table of database `b` in group `a`

System Parameter Column

Specifies a column of a system table.

| Property | Value |
|----------|----------|
| type | "column" |
| table | string |
| column | string |

Database Parameter Column

Specifies a parameter column of a database table.

| Property | Value |
|----------|------------------------------------|
| type | "column" |
| database | database specifier |
| table | string table name |
| column | string parameter name |

Database Attribute Column

Specifies an attribute column of a database table.

| Property | Value |
|----------|------------------------------------|
| type | "column" |
| database | database specifier |
| table | string table name |
| column | string attribute name |

Database Field Column

Specifies a field column of a database table.

| Property | Value |
|----------|------------------------------------|
| type | "column" |
| database | database specifier |
| table | string table name |
| column | field specifier |

Alias

Although the alias is not technically a column, it can refer directly by name to any column in the source, or to an alias of a result column.

| Property | Value |
|----------|---------|
| type | "alias" |
| value | string |

Supports shorthand syntax with the `$alias` property.

| Property | Value |
|----------------------|--------|
| <code>\$alias</code> | string |

Evaluations

Evaluations are expressions evaluated by MySQL.

Between

Returns true if the expression `e` is between `min` and `max`.

| Property | Value |
|----------|------------|
| type | "between" |
| e | expression |
| min | expression |
| max | expression |

Supports shorthand syntax with the `$between` property. Takes a JSON array of exactly 3 expressions, in the order `e`, `min`, and `max`.

| Property | Value |
|------------------------|----------------------------|
| <code>\$between</code> | array of three expressions |

Binary

Binary operation, evaluated as `e1 op e2`.

| Property | Value |
|----------|----------|
| type | "binary" |

| Property | Value |
|----------|------------|
| op | string |
| e1 | expression |
| e2 | expression |

Valid binary operators are as follows:

| Operator | Description |
|----------|---|
| and | logical AND |
| or | logical OR |
| = | equal |
| != | not equal |
| > | greater |
| >= | greater or equal |
| < | less |
| <= | less or equal |
| is | test against NULL |
| like | simple pattern matching, see here |
| regexp | advanced pattern matching, see here |
| + | addition |
| - | subtraction |
| * | multiplication |
| / | division |
| % | modulus |
| & | bit-wise AND |
| | bit-wise OR |
| << | left shift |
| >> | right shift |

Supports shorthand syntax with any operator by prefixing it with `$`. Takes a JSON array of two or more expressions. If more than two expressions are provided, behavior depends on the operator type. Logic and math operators perform each binary operation in order of expressions. For example:

- `{"$and": [true, true, false]}` = `(true and true) and false` = `false`
- `{"$/": [12, 3, 2, 2]}` = `((12 / 3) / 2) / 2` = `1`

Comparison operators perform a logical AND of the comparisons of the first expression to each additional expression.

- `{"$=": [0, 1, 2]}` = `(0 = 1) and (0 = 2)` = `false`

Case

Case logic expression. If the `base` is provided, returns the `then` expression of the first case in which `when` = `base`. Otherwise returns the first case in which `when` is `true`. If no case is satisfied returns `else` if it is provided, or `NULL` otherwise.

| Property | Value |
|--------------------|-----------------------|
| <code>type</code> | "case" |
| <code>base</code> | expression (optional) |
| <code>cases</code> | array of case options |
| <code>else</code> | expression (optional) |

Case Option

| Property | Value |
|-------------------|------------|
| <code>when</code> | expression |
| <code>then</code> | expression |

Collate

Performs the MySQL `COLLATE` function.

| Property | Value |
|------------------------|------------|
| <code>type</code> | "collate" |
| <code>e</code> | expression |
| <code>collation</code> | string |

Count Rows

Performs the MySQL `COUNT(*)` function.

| Property | Value |
|-------------------|--------------|
| <code>type</code> | "count_rows" |

Example

```
{ "type": "count_rows" }
```

Exists

Returns true if the enclosed `SELECT` returns at least one row.

| Property | Value |
|---------------------|------------------------|
| <code>type</code> | "exists" |
| <code>select</code> | select |

Supports shorthand syntax with the `$exists` property.

| Property | Value |
|-----------------------|------------------------|
| <code>\$exists</code> | select |

Function

Performs a MySQL function. The number of arguments varies depending on the function.

| Property | Value |
|-----------------------|--------------------------------------|
| <code>type</code> | "function" |
| <code>function</code> | <code>string</code> |
| <code>args</code> | array of expressions |

Available functions are:

| Name | Args | Aggregate | Description |
|-----------------------------|----------------|-----------|---|
| <code>AVG</code> | 1 | yes | arithmetic average |
| <code>AVG_DISTINCT</code> | 1 | yes | arithmetic average of distinct values of argument |
| <code>BIT_AND</code> | 1 | yes | bit-wise AND |
| <code>BIT_OR</code> | 1 | yes | bit-wise OR |
| <code>BIT_XOR</code> | 1 | yes | bit-wise XOR |
| <code>CEIL</code> | 1 | yes | returns the smallest integer value not less than the argument |
| <code>COUNT</code> | 1 | yes | returns the number of rows in the which the argument is not <code>NULL</code> |
| <code>COUNT_DISTINCT</code> | <code>n</code> | yes | returns the number of distinct value(s) of the arguments |
| <code>FLOOR</code> | 1 | yes | returns the largest integer value not greater than the argument |
| <code>MAX</code> | 1 | yes | returns the maximum value of the argument |

| Name | Args | Aggregate | Description |
|--------------|------|-----------|---|
| MIN | 1 | yes | returns the minimum value of the argument |
| POW | 2 | no | |
| STDDEV_POP | 1 | yes | returns the population standard deviation of the argument |
| STDDEV_SAMP | 1 | yes | returns the sample standard deviation of the argument |
| SUM | 1 | yes | returns the sum of the argument |
| SUM_DISTINCT | 1 | yes | returns the sum of the distinct values of the argument |
| TRUNCATE | 2 | no | |
| VAR_POP | 1 | yes | returns the population variance of the argument |
| VAR_SAMP | 1 | yes | returns the sample variance of the argument |

Supports shorthand syntax by prefixing any function name with `$$`. For example, `{ "$$pow": [2, 3] }` evaluates to `8`.

In

Returns true if an expression is contained in a set of values. If an empty array is provided for `values`, will always return false.

| Property | Value |
|----------|----------------------|
| type | "in" |
| e | expression |
| values | array of expressions |

Supports shorthand syntax with the `$in` property. Takes an array of a single expression (`e`), followed by either an array of expression(s) (`values`) or a `SELECT` object.

| Property | Value |
|----------|--|
| \$in | array of one expression, then either an array of expressions or a select |

In Select

Returns true if `e` is in the result of the `select` query.

| Property | Value |
|----------|-------|
|----------|-------|

| | |
|--------|-------------|
| type | "in_select" |
| e | expression |
| select | select |

Supports shorthand syntax with the `$in` property (see [above](#)).

Select Expression

Returns the value of the first column in the first row of the result set of the query.

| Property | Value |
|----------|----------|
| type | "select" |
| select | select |

Supports shorthand syntax with the `$select` property.

| Property | Value |
|----------|--------|
| \$select | select |

Unary Expression

Unary operator expression, evaluated as `op e`.

| Property | Value |
|----------|------------|
| type | "unary" |
| op | string |
| e | expression |

Valid unary operators are:

| Operator | Description |
|----------|-------------|
| not | logical NOT |
| - | negate |
| ~ | bit invert |

Supports shorthand syntax with any operator by prefixing it with `$`. Takes a single expression as a value.

| Property | Value |
|----------|------------|
| \$ op | expression |

Revision #6

Created 9 June 2022 16:24:27 by Nick Dobson

Updated 15 January 2026 20:43:45 by Bradley Tse