

Chapter 8

How to work with data types

Objectives

Applied

1. Code queries that convert data from one data type to another.

Knowledge

1. Describe the data that can be stored in any of the character, numeric, date/time, and large object data types.
2. Describe ENUM and SET data types.

MySQL data type categories

- Character
- Numeric
- Date and time
- Binary
- Large Object (LOB)
- Spatial
- JSON

The character types

Type	Bytes
CHAR (M)	Mx4
VARCHAR (M)	L+1

How the character types work with utf8mb4

Data type	Original value	Value stored	Bytes used
CHAR (2)	'CA'	'CA'	8
CHAR (10)	'CA'	'CA'	40
VARCHAR (10)	'CA'	'CA'	3
VARCHAR (20)	'California'	'California'	11
VARCHAR (20)	'New York'	'New York'	9
VARCHAR (20)	"Murach's MySQL"	"Murach's MySQL"	15

Terms to know about character types

- Latin1 character set
- utf8mb3 character set
- utf8mb4 character set
- Unicode standard

The integer types

Type	Bytes
BIGINT	8
INT	4
MEDIUMINT	3
SMALLINT	2
TINYINT	1

How the UNSIGNED attribute works

Data type	Original value	Value stored	Value displayed
INT	99	99	99
INT	-99	-99	-99
INT UNSIGNED	99	99	99
INT UNSIGNED	-99	None	None

The fixed-point type

Type	Bytes
DECIMAL (M, D)	Vary

The floating-point types

Type	Bytes
DOUBLE	8
FLOAT	4

How the fixed-point and floating-point types work

Data type	Original value	Value stored	Bytes used
DECIMAL (9, 2)	1.2	1.20	5
DECIMAL (9, 2)	1234567.89	1234567.89	5
DECIMAL (9, 2)	-1234567.89	-1234567.89	5
DECIMAL (18, 9)	1234567.89	1234567.890000000	8
DOUBLE	1234567.89	1234567.89	8
FLOAT	1234567.89	1234570	4

Terms to know about numeric data types

- Real number
- Precision
- Scale
- Exact numeric type
- Floating-point number
- Approximate numeric type

The date and time types

Type	Bytes
DATE	3
TIME	3
DATETIME	8
TIMESTAMP	4
YEAR	1

How MySQL interprets literal date/time values

Literal value	Value stored in DATE column
'2022-08-15'	2022-08-15
'2022-8-15'	2022-08-15
'22-8-15'	2022-08-15
'20220815'	2022-08-15
20220815	2022-08-15
'8-15-22'	(error)
'2022-02-31'	(error)

How MySQL interprets literal date/time values (continued)

Literal value

Value stored in TIME column

`'7:32'`

`07:32:00`

`'19:32:11'`

`19:32:11`

`'193211'`

`19:32:11`

`193211`

`19:32:11`

`'19:61:11'`

`(error)`

Literal Value

Value stored in DATETIME or TIMESTAMP column

`'2022-08-15 19:32:11'`

`2022-08-15 19:32:11`

`'2022-08-15'`

`2022-08-15 00:00:00`

The ENUM and SET types

Type	Bytes
<code>ENUM(val1, val2, ...)</code>	1-2
<code>SET(val1, val2, ...)</code>	1-8

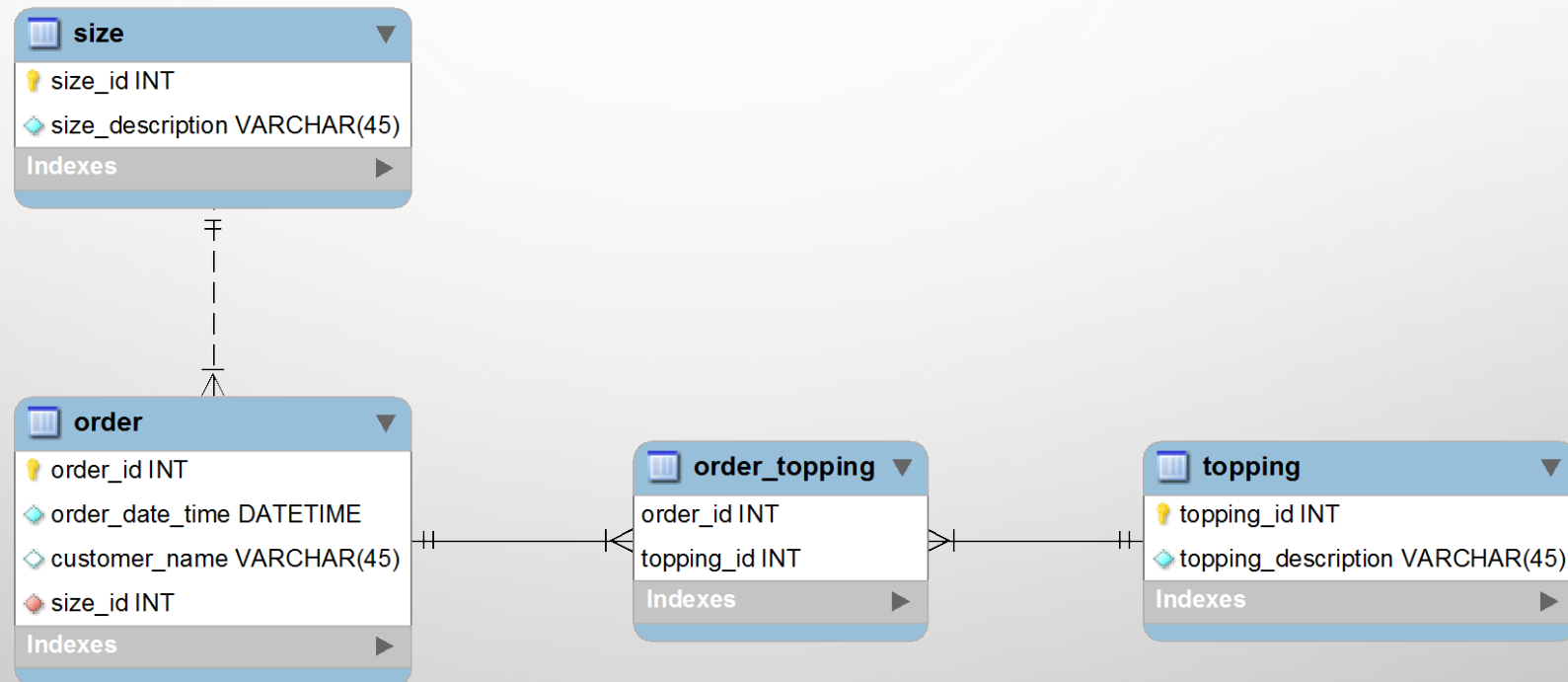
How an ENUM('Yes','No','Maybe') column works

Value	Value stored	Value displayed
'Yes'	1	'Yes'
'No'	2	'No'
'Maybe'	3	'Maybe'
'Possibly'	(error)	
' '	(error)	

How a SET('Pepporoni','Mushrooms','Olives') column works

Value	Valued stored (binary)	Value displayed
'Pepperoni'	1 (00000001)	'Pepperoni'
'Mushrooms'	2 (00000010)	'Mushrooms'
'Olives'	4 (00000100)	'Olives'
'Olives,Pepperoni'	5 (00000101)	'Pepperoni,Olives'
'Olives,Olives,Mushrooms'	6 (00000110)	'Mushrooms,Olives'
''	0 (00000000)	''
'Pepperoni,Sausage'	(error)	

RELATIONAL ALTERNATIVES TO ENUM AND SET



The binary data types

Type	Bytes
BINARY (M)	M
VARBINARY (M)	L+1

The large object types

Type	Bytes
LOB	
LONGBLOB	L+4
MEDIUMBLOB	L+3
BLOB	L+2
TINYBLOB	L+1
LONGTEXT	L+4
MEDIUMTEXT	L+3
TEXT	L+2
TINYTEXT	L+1

Terms to know about large objects

- BLOB (binary large object) types
- CLOB (character large object) types

Implicitly convert a number to a string

```
SELECT invoice_total, CONCAT('$', invoice_total)
FROM invoices
```

	invoice_total	CONCAT('\$', invoice_total)
▶	3813.33	\$3813.33
	40.20	\$40.20
	138.75	\$138.75
	144.70	\$144.70

Implicitly convert a string to a number

```
SELECT invoice_number, 989319/invoice_number
FROM invoices
```

	invoice_number	989319/invoice_number
▶	989319-457	1
	263253241	0.0037580505988908225
	963253234	0.0010270601385803393
	2-000-2993	494659.5
	963253251	0.0010270601204542418

Implicitly convert a date to a number

```
SELECT invoice_date, invoice_date + 1  
FROM invoices
```

	invoice_date	invoice_date + 1
▶	2022-08-02	20220803
	2022-08-01	20220802
	2022-07-31	20220732
	2022-07-30	20220731
	2022-07-28	20220729

The syntax of the CAST function

```
CAST(expression AS cast_type)
```

The syntax of the CONVERT function

```
CONVERT(expression, cast_type)
```

Cast types you can use in these functions

```
CHAR[ (N) ]
```

```
DATE
```

```
DATETIME
```

```
TIME
```

```
SIGNED [INTEGER]
```

```
UNSIGNED [INTEGER]
```

```
DECIMAL[ (M[,D]) ]
```

A statement that uses the CAST function

```
SELECT invoice_id, invoice_date, invoice_total,  
       CAST(invoice_date AS CHAR(10)) AS char_date,  
       CAST(invoice_total AS SIGNED) AS integer_total  
FROM invoices
```

	invoice_id	invoice_date	invoice_total	char_date	integer_total
▶	1	2022-04-08	3813.33	2022-04-08	3813
	2	2022-04-10	40.20	2022-04-10	40
	3	2022-04-13	138.75	2022-04-13	139
	4	2022-04-16	144.70	2022-04-16	145
	5	2022-04-16	15.50	2022-04-16	16

A statement that uses the CONVERT function

```
SELECT invoice_id, invoice_date, invoice_total,  
       CONVERT(invoice_date, CHAR(10)) AS char_date,  
       CONVERT(invoice_total, SIGNED) AS integer_total  
FROM invoices
```

	invoice_id	invoice_date	invoice_total	char_date	integer_total
▶	1	2022-04-08	3813.33	2022-04-08	3813
	2	2022-04-10	40.20	2022-04-10	40
	3	2022-04-13	138.75	2022-04-13	139
	4	2022-04-16	144.70	2022-04-16	145
	5	2022-04-16	15.50	2022-04-16	16

The FORMAT and CHAR functions

`FORMAT (number, decimal)`

`CHAR (value1 [, value2] ...)`

FORMAT function examples

Function	Result
<code>FORMAT (1234567.8901, 2)</code>	<code>1,234,567.89</code>
<code>FORMAT (1234.56, 4)</code>	<code>1,234.5600</code>
<code>FORMAT (1234.56, 0)</code>	<code>1,235</code>

CHAR function examples for common control characters

Function	Control character
----------	-------------------

CHAR(9)	Tab
---------	-----

CHAR(10)	Line feed
----------	-----------

CHAR(13)	Carriage return
----------	-----------------

A statement that uses the CHAR function

```
SELECT CONCAT(vendor_name, CHAR(13,10), vendor_address1,  
             CHAR(13,10), vendor_city, ', ', vendor_state, ' ',  
             vendor_zip_code) AS "Mailing Label"  
FROM vendors  
WHERE vendor_id = 1
```

Result Grid		 Filter Rows:	Export: 	Wrap Cell Content: 
	Mailing Label			
▶	US Postal Service Attn: Supt. Window Services Madison, WI 53707			