Welcome to this course!

SQL SERVER FUNCTIONS FOR MANIPULATING DATA

Ana Voicu
Data Engineer

Topics covered

- The most important data types
- Functions for these types:
    - Date and time functions
    - String functions
    - Functions for numeric operations

Categories of data types

- Exact numerics
- Approximate numerics
- Date and time
- Character strings
- Unicode character strings
- Binary strings
- *Other data types*

Exact numerics

- Whole numbers
  - smallint
  - tinyint
  - int
  - bigint

- Decimal numbers
  - numeric
  - decimal
  - money
  - smallmoney

Exact numerics - integers

Numbers without a decimal point

Data type	**Storage**
bigint	8 Bytes
int	4 Bytes
smallint	2 Bytes
tinyint	1 Byte

Exact numerics - decimals

$$\underbrace{1234567.\underbrace{675}_{\text{S (scale) = number of decimal digits stored to the right of the decimal point}}}_{\text{P (precision) = maximum total number of decimal digits that will be stored}}$$

Precision	Storage
1 – 9	5 Bytes
10 – 19	9 Bytes
20 – 28	13 Bytes
29 - 38	17 Bytes

Approximate numerics

- Float
- Real
- Store **approximate** numeric values

Date and time data types

Data type	Format	Accuracy
time	hh:mm:ss[.nnnnnnn]	100 nanoseconds
date	YYYY-MM-DD	1 day
smalldatetime	YYYY-MM-DD hh:mm:ss	1 minute
datetime	YYYY-MM-DD hh:mm:ss[.nnn]	0.00333 second
datetime2	YYYY-MM-DD hh:mm:ss[.nnnnnnn]	100 nanoseconds

Date has the smallest accuracy while datetime2 is the most exact.

Character and Unicode character data types

Character data types store character strings (ASCII)

- char
- varchar
- text

Unicode data types are used for storing Unicode data (non-ASCII)

- nchar
- nvarchar
- ntext

With Unicode types, you can store characters from all languages around the world.

Other data types

- binary
- image
- cursor
- rowversion
- uniqueidentifier
- xml
- Spatial Geometry / Geography Types

Let's see what you know!

SQL SERVER FUNCTIONS FOR MANIPULATING DATA

Implicit conversion

SQL SERVER FUNCTIONS FOR MANIPULATING DATA

Ana Voicu
Data Engineer

Data comparison

Keep in mind: for comparing two values, they need to be of the same type.

Otherwise:

- SQL Server converts from one type to another (**IMPLICIT**)
- The developer explicitly converts the data (**EXPLICIT**)

Example of possible conversions

```
SELECT
    company
    bean_type,
    cocoa_percent
FROM ratings;
```

Example of possible conversions

```
SELECT
    company
    bean_type,
    cocoa_percent
FROM ratings
WHERE cocoa_percent > 0.5;
```

```
| company | bean_type   | cocoa_percent |
|---------|------------|---------------|
| Amedei  | Blend       | 0.7000        |
| Bonnat  | Trinitario | 0.7500        |
| ...     | ...         | ...           |
```

Example of possible conversions

```
SELECT
    company
    bean_type,
    cocoa_percent
FROM ratings
WHERE cocoa_percent > -2;
```

```
| company | bean_type   | cocoa_percent |
|---------|-------------|---------------|
| Amedei  | Blend       | 0.7000        |
| Bonnat  | Trinitario  | 0.7500        |
| ...     | ...         | ...           |
```

Example of possible conversions

```
SELECT
    company
    bean_type,
    cocoa_percent
FROM ratings
WHERE cocoa_percent > GETDATE();
```

It will work and the ‘cocoa_percent’ will be implicitly converted from decimal to datetime.
The value 0 corresponds to the date ’01-01-1900’.
Adding 1 to the decimal number will add one day when it’s converted to a datetime.
Converting the number 365 to datetime will produce the date ’01-01-1901’.

```
| company | bean_type   | cocoa_percent |
|---------|------------|---------------|
| ...     | ...        | ...           |
```

Example of possible conversions

```sql
SELECT
    company
    bean_type,
    cocoa_percent
FROM ratings
WHERE cocoa_percent > 'A';
```

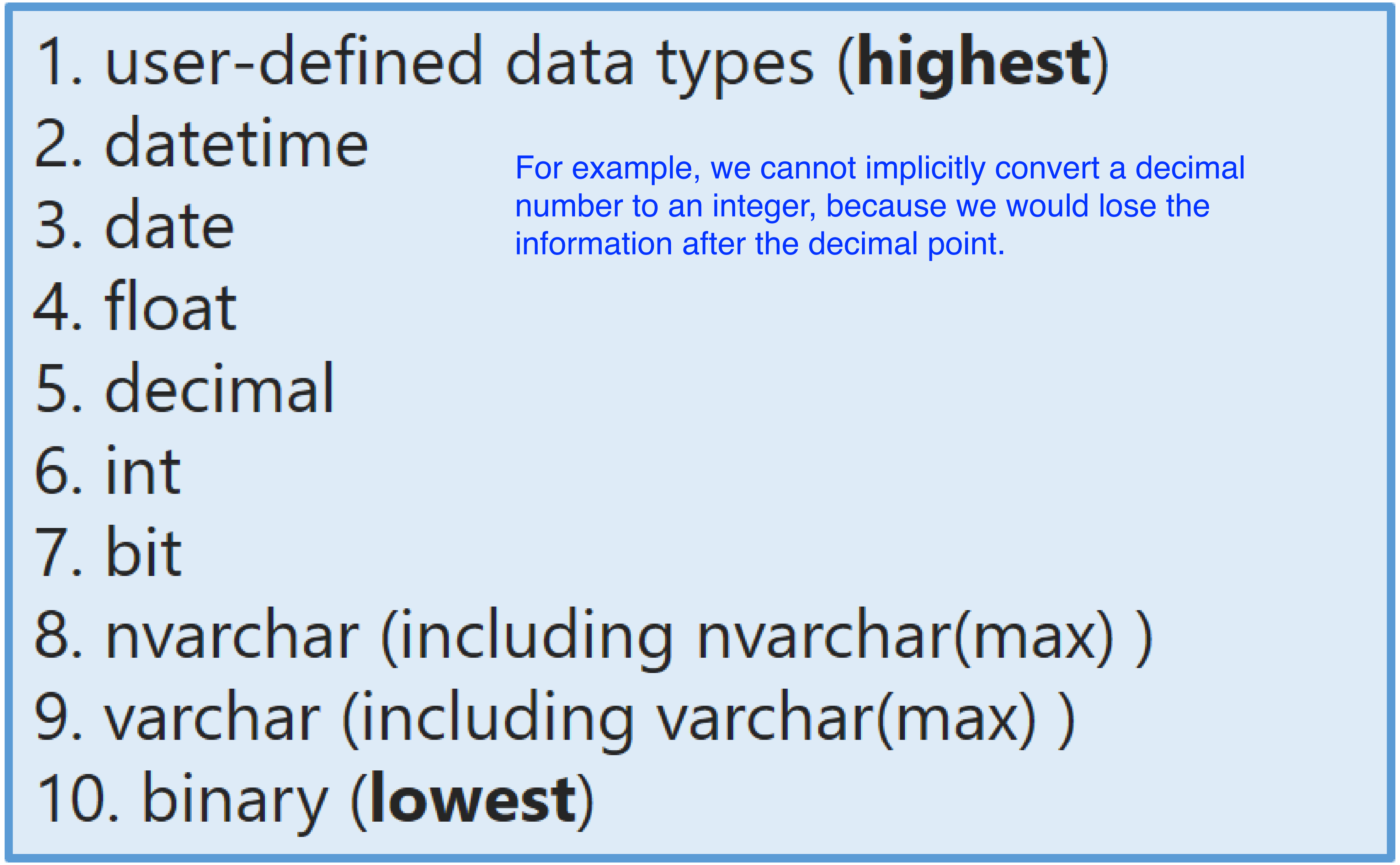
```
| result                                          |
|-------------------------------------------------|
| Error converting data type varchar to numeric. |
```

Example of possible conversions

```
SELECT
    company
    bean_type,
    cocoa_percent
FROM ratings
WHERE cocoa_percent > '0.5';
```

```
| company | bean_type   | cocoa_percent |
|---------|-------------|---------------|
| Amedei  | Blend       | 0.7000        |
| Bonnat  | Trinitario  | 0.7500        |
| ...     | ...         | ...           |
```

Data type precedence



1. user-defined data types (**highest**)
2. datetime
3. date
4. float
5. decimal
6. int
7. bit
8. nvarchar (including nvarchar(max))
9. varchar (including varchar(max))
10. binary (**lowest**)

For example, we cannot implicitly convert a decimal number to an integer, because we would lose the information after the decimal point.

Data type precedence

![Data type precedence, from highest to lowest

Implicit conversion between data types

To → From ↓	DATETIME	FLOAT	DECIMAL	INT	BIT	NVARCHAR	VARCHAR
DATETIME		X	X	X	X	V	V
FLOAT	V		V	V	V	V	V
DECIMAL	V	V		V	V	V	V
INT	V	V	V		V	V	V
BIT	V	V	V	V		V	V
NVARCHAR	V	V	V	V	V		V
VARCHAR	V	V	V	V	V	V	

Performance impact of implicit conversion

- Implicit conversion is done for each row of the query
- Implicit conversion can be prevented with a good database schema design.

Let's practice!

SQL SERVER FUNCTIONS FOR MANIPULATING DATA

Explicit conversion

SQL SERVER FUNCTIONS FOR MANIPULATING DATA

Ana Voicu
Data Engineer

Implicit and explicit conversion

- **IMPLICIT** - performed automatically, behind the scenes
- **EXPLICIT** - performed with the functions `CAST()` and `CONVERT()`
  - `CAST()` and `CONVERT()` are used to convert from one data type to another

CAST()

```
CAST(expression AS data_type [(length)])
```

```
SELECT
     CAST(3.14 AS int) AS DECIMAL_TO_INT,
     CAST('3.14' AS decimal(3,2)) AS STRING_TO_DECIMAL,
     CAST(GETDATE() AS nvarchar(20)) AS DATE_TO_STRING,
     CAST(GETDATE() AS float) AS DATE_TO_FLOAT;
```

```
| DECIMAL_TO_INT | STRING_TO_DECIMAL | DATE_TO_STRING      | DATE_TO_FLOAT      |
|----------------|--------------------|--------------------|-------------------|
| 3              | 3.14               | Apr 11 2019 1:01PM | 43531.7052687886 |
```

CONVERT()

```
CONVERT(data_type [(length)], expression [,style])
```

```
SELECT
     CONVERT(int, 3.14) AS DECIMAL_TO_INT,
     CONVERT(decimal(3,2), '3.14') AS STRING_TO_DECIMAL,
     CONVERT(nvarchar(20), GETDATE(), 104) AS DATE_TO_STRING,
     CONVERT(float, GETDATE()) AS DATE_TO_FLOAT;
```

style parameter

```
| DECIMAL_TO_INT | STRING_TO_DECIMAL | DATE_TO_STRING | DATE_TO_FLOAT     |
|----------------|-------------------|----------------|-------------------|
| 3              | 3.14              | 11.04.2019     | 43531.7052687886 |
```

CAST() vs. CONVERT()

- `CAST()` comes from the SQL standard and `CONVERT()` is SQL Server specific
- `CAST()` is available in most database products
- `CONVERT()` performs slightly better in SQL Server

Let's practice!

SQL SERVER FUNCTIONS FOR MANIPULATING DATA