319
XMLTYPE

`XMLType` is a system-defined opaque type for handling XML data. It as predefined member functions on it to extract XML nodes and fragments.

You can create columns of `XMLType` and insert XML documents into it. You can also generate XML documents as `XMLType` instances dynamically using the `SYS_XMLAGG SQL` function.

This chapter contains the following topics:

- Summary of XMLType Subprograms

> **See Also:**
>
> – *Oracle XML DB Developer's Guide*

Summary of XMLType Subprograms

This table summarizes functions and procedures of `XMLType`.

Table 319-1 XMLTYPE Subprograms

Method	Description
CREATENONSCHEMABASEDXML	Creates a non schema based XML from the input schema based instance.
CREATESCHEMABASEDXML	Creates a schema based `XMLType` instance from the non-schema based instance using the input schema URL.
CREATEXML	Static function for creating and returning an `XMLType` instance.
EXISTSNODE	Takes a `XMLType` instance and a `XPath` and returns 1 or 0 indicating if applying the `XPath` returns a non-empty set of nodes.
EXTRACT	Takes a `XMLType` instance and an `XPath`, applies the XPath expression and returns the results as an `XMLType`.
GETBLOBVAL	Returns the value of the `XMLType` instance as a `BLOB`
GETCLOBVAL	Returns the value of the `XMLType` instance as a `CLOB`.
GETNAMESPACE	Returns the namespace for the top level element in a schema based document.
GETNUMBERVAL	Returns the value of the `XMLType` instance as a `NUMBER`. This is only valid if the input `XMLType` instance contains a simple text node and is convertible to a number.
GETROOTELEMENT	Returns the root element of the input instance. Returns `NULL` if the instance is a fragment
GETSCHEMAURL	Returns the XML schema URL if the input is an XML Schema based.

Table 319-1 (Cont.) XMLTYPE Subprograms

Method	Description
GETSTRINGVAL	Returns the value of the `XMLType` instance as a string.
ISFRAGMENT	Checks if the input `XMLType` instance is a fragment or not. A fragment is a XML instance, which has more than one root element.
ISSCHEMABASED	Returns 1 or 0 indicating if the input `XMLType` instance is a schema based one or not.
ISSCHEMAVALID	Checks if the input instance is schema valid according to the given schema URL.
ISSCHEMAVALIDATED	Checks if the instance has been validated against the schema.
SCHEMAVALIDATE	Validates the input instance according to the XML Schema. Raises error if the input instance is non-schema based.
SETSCHEMAVALIDATED	Sets the schema valid flag to avoid costly schema validation.
TOOBJECT	Converts the `XMLType` instance to an object type.
TRANSFORM	Takes an `XMLType` instance and an associated stylesheet (which is also an `XMLType` instance), applies the stylesheet and returns the result as XML.
XMLTYPE	Constructs an instance of the `XMLType` datatype. The constructor can take in the XML as a `CLOB`, `VARCHAR2` or take in a object type.

CREATENONSCHEMABASEDXML

This member function creates a non-schema based XML document from a schema based instance.

Syntax

```
MEMBER FUNCTION CREATENONSCHEMABASEDXML
return XMLType deterministic;
```

CREATESCHEMABASEDXML

This member function creates a schema based `XMLType` instance from a non-schema based `XMLType` value.

It uses either the supplied `SCHEMA` URL, or the `SCHEMALOCATION` attribute of the instance.

Syntax

```
MEMBER FUNCTION createSchemaBasedXML(
schema IN varchar2 := NULL)
return XMLType deterministic;
```

Table 319-2 CREATESCHEMABASEDXML Subprogram Parameters

Parameter	Description
`schema`	Optional XMLSchema URL used to convert the value to the specified schema.

CREATEXML

This static function creates and returns an `XMLType` instance. The string and `CLOB` parameters used to pass in the date must contain well-formed and valid XML documents.

The options are described in the following table.

Table 319-3 CREATEXML Subprograms

Syntax	Description

`STATIC FUNCTION createXML(`
`  xmlData IN varchar2)`
`RETURN XMLType deterministic;`
	Creates the `XMLType` instance from a string.

`STATIC FUNCTION createXML(`
`  xmlData IN clob)`
`RETURN XMLType deterministic;`
	Creates the `XMLType` instance from a `CLOB`.

`STATIC FUNCTION createXML (`
`  xmlData IN clob,`
`  schema IN varchar2,`
`  validated IN number := 0,`
`  wellformed IN number := 0 )`
`RETURN XMLType deterministic;`
	This static function creates a schema-based `XMLType` instance using the specified schema and xml data parameters.

`STATIC FUNCTION createXML (`
`  xmlData IN varchar2,`
`  schema IN varchar2,`
`  validated IN number := 0,`
`  wellformed IN number := 0)`
`RETURN XMLType deterministic;`
	This static function creates a schema-based `XMLType` instance using the specified schema and xml data parameters.

`STATIC FUNCTION createXML (`
`  xmlData IN "<ADT_1>",`
`  schema IN varchar2 := NULL,`
`  element IN varchar2 := NULL,`
`  validated IN NUMBER := 0)`
`RETURN XMLType deterministic;`
	Creates an XML instance from an instance of an user-defined type.

`STATIC FUNCTION createXML (`
`  xmlData IN SYS_REFCURSOR,`
`  schema in varchar2 := NULL,`
`  element in varchar2 := NULL,`
`  validated in number := 0)`
`RETURN XMLType deterministic;`
	Creates an XML instance from a cursor reference. You can pass in any arbitrary SQL query as a `CURSOR`.

Table 319-3 (Cont.) CREATEXML Subprograms

Syntax	Description

`STATIC FUNCTION createXML (`
`xmlData IN AnyData,`
`schema in varchar2 := NULL,`
`element in varchar2 := NULL,`
`validated in number := 0)`
`RETURN sys.XMLType deterministic`
`parallel_enable`
	Creates an XML instance from `ANYDATA`.If the `ANYDATA`instance contains an ADT, the `XMLType`returned is the same as would be returned for a call directly on the ADT. If the `ANYDATA`contains a scalar, the `XMLType` contains a leaf node with the scalar value. The element name for this node is taken from the optional element string if present, and is "`ANYDATA`" if it is not.

`STATIC FUNCTION createXML (`
`xmlData IN blob,`
`csid IN number,`
`schema IN varchar2,`
`validated IN number := 0,`
`wellformed IN number := 0)`
`return sys.XMLType deterministic`
	Creates an XML instance from a `BLOB`.

`STATIC FUNCTION createXML (`
`xmlData IN bfile,`
`csid IN number,`
`Schema IN varchar2,`
`validated IN number := 0,`
`wellformed IN number := 0)`
`return sys.XMLType deterministic`
	Creates an XML instance from a `BFILE`.

Table 319-4 CREATEXML Parameters

Parameter	Description
`xmlData`	The actual data in the form of a `BFILE`, `BLOB`, `CLOB`, `REF` cursor, `VARCHAR2` or object type.
`schema`	Optional Schema URL to be used to make the input conform to the given schema. **Caution**: Oracle does not support use of types generated by Schema Registration (see *Oracle XML DB Developer's Guide*).
`validated`	Flag to indicate that the instance is valid according to the given XML Schema. (Default is `0`)
`wellformed`	Flag to indicate that the input is well formed. If set, then the database would not do well formed check on the input instance. (Default is `0`)
`element`	Optional element name in the case of the `ADT_1` or `REF CURSOR` constructors. (Default is `NULL`). **Caution**: Oracle does not support use of types generated by Schema Registration (see *Oracle XML DB Developer's Guide*).
`csid`	The character set id of input XML data.

EXISTSNODE

This member function checks if the node exists.

If the XPath string is `NULL` or the document is empty, then a value of 0 is returned, otherwise returns 1.

The options are described in the following table.

Syntax	Description

`MEMBER FUNCTION existsNode(`
`  xpath IN varchar2)`
`RETURN number deterministic;`
	Given an XPath expression, checks if the XPath applied over the document can return any valid nodes.

`MEMBER FUNCTION existsNode(`
`  xpath in varchar2,`
`  nsmap in varchar2)`
`RETURN number deterministic;`
	This member function uses the XPath expression with the namespace information and checks if applying the XPath returns any nodes or not.

Table 319-5 EXISTSNODE Subprogram Parameters

Parameter	Description
`xpath`	The XPath expression to test.
`nsmap`	Optional namespace mapping.

EXTRACT

This member function extracts an `XMLType` fragment and returns an `XMLType` instance containing the result nodes. If the XPath does not result in any nodes, it then returns `NULL`.

The options are described in the following table.

Syntax	Description

`MEMBER FUNCTION extract(`
`  xpath IN varchar2)`
`RETURN XMLType deterministic;`
	Given an XPath expression, applies the XPath to the document and returns the fragment as an `XMLType`.

`MEMBER FUNCTION extract(`
`  xpath IN varchar2,`
`  nsmap IN varchar2)`
`RETURN XMLType deterministic;`
	This member function applies the XPath expression and namespace mapping, over the XML data to return a `XMLType` instance containing the resultant fragment.

Table 319-6 EXTRACT Subprogram Parameters

Parameter	Description
`xpath`	The XPath expression to apply.
`nsmap`	Optional prefix to namespace mapping information.

GETBLOBVAL

This member function returns a BLOB containing the serialized XML representation. If the BLOB returned is temporary, it must be freed after use.

Syntax

```
MEMBER FUNCTION getBlobVal(
   csid    IN    NUMBER)
 RETURN BLOB DETERMINISTIC;
```

Table 319-7 GETBLOBVAL Subprogram Parameters

Parameter	Description
csid	The desired character set ID of output BLOB

GETCLOBVAL

This member function returns a CLOB containing the serialized XML representation. If the CLOB returned is temporary, it must be freed after use. The CLOBs returned by this function are read-only.

Syntax

```
MEMBER FUNCTION getClobVal()
RETURN clob deterministic;
```

GETNAMESPACE

GETNAMESPACE is a member function. It returns the namespace of the top level element in the instance. It returns NULL if the input is a fragment or is a non-schema based instance.

Syntax

```
MEMBER FUNCTION getNamespace
return varchar2 deterministic;
```

GETNUMBERVAL

This is a member function. It returns a numeric value, formatted from the text value pointed to by the XMLType instance. The XMLType must point to a valid text node that contains a numerical value.

The options are described in the following table.

Syntax

```
MEMBER FUNCTION getNumberVal()
RETURN number deterministic;
```

GETROOTELEMENT

this member function gets the root element of the `XMLType` instance. It returns `NULL` if the instance is a fragment.

Syntax

```
MEMBER FUNCTION getRootElement
return varchar2 deterministic;
```

GETSCHEMAURL

This member function returns the XML Schema URL corresponding to the `XMLType` instance, if the `XMLType` instance is a schema-based document. Otherwise, it returns `NULL`.

Syntax

```
MEMBER FUNCTION getSchemaURL
return varchar2 deterministic;
```

GETSTRINGVAL

This member function returns the document as a string. It returns a string containing the seralized XML representation, or in the case of text nodes, the text itself.

If the XML document exceeds the `VARCHAR2` maximum size (4000), then an error is raised at run time.

Syntax

```
MEMBER FUNCTION getStringVal()
RETURN varchar2 deterministic;
```

ISFRAGMENT

`ISFRAGMENT` determines if the `XMLType` instance corresponds to a well-formed document, or a fragment. It returns 1 or 0 indicating if the `XMLType` instance contains a fragment or a well-formed document.

Syntax

```
MEMBER FUNCTION isFragment()
RETURN number deterministic;
```

ISSCHEMABASED

This member function determines whether the `XMLType` instance is schema-based or not. It returns 1 or 0 depending on whether the `XMLType` instance is schema-based.

Syntax

```
MEMBER FUNCTION isSchemaBased
return number deterministic;
```

ISSCHEMAVALID

This member function checks if the input instance conforms to a specified schema. I does not change the validation status of the XML instance.

If an XML Schema URL is not specified and the xml document is schema based, the conformance is checked against the `XMLType` instance's own schema.

Syntax

```
member function isSchemaValid(
schurl IN VARCHAR2 := NULL,
elem IN VARCHAR2 := NULL)
return NUMBER deterministic;
```

Table 319-8 ISSCHEMAVALID Subprogram Parameters

Parameter	IN / OUT	Description
`schurl`	`(IN)`	The URL of the XML Schema against which to check conformance.
`elem`	`(IN)`	Element of a specified schema, against which to validate. This is useful when we have a XML Schema which defines more than one top level element, and we want to check conformance against a specific one of these elements.

ISSCHEMAVALIDATED

This member function returns the validation status of the `XMLType` instance to tell if a schema-based instance has been actually validated against its schema. It returns 1 if the instance has been validated against the schema, 0 otherwise.

Syntax

```
MEMBER FUNCTION isSchemaValidated
return NUMBER deterministic;
```

SCHEMAVALIDATE

This member procedure validates the XML instance against its schema, if it has not already been done.

For non-schema based documents an error is raised. If validation fails an error is raised; else, the document's status is changed to validated.

Syntax

```
MEMBER PROCEDURE schemaValidate(
   self IF OUT NOCOPY XMLType);
```

Table 319-9 SCHEMAVALIDATE Subprogram Parameters

Parameter	IN / OUT	Description
`self`	`(OUT)`	XML instance being validated against the schema.

SETSCHEMAVALIDATED

This member function sets the `VALIDATION` state of the input XML instance.

Syntax

```
MEMBER PROCEDURE setSchemaValidated(
self IF OUT NOCOPY XMLType,
   flag IN BINARY_INTEGER := 1);
```

Table 319-10 SERTSSCHEMAVALIDATED Subprogram Parameters

Parameter	IN / OUT	Description
`self`	`(OUT)`	XML instance.
`flag`	`(IN)`	0 - `NOT VALIDATED`; 1 - `VALIDATED` (Default)

TOOBJECT

This member procedure converts the XML value to an object type using the `XMLSCHEMA` mapping, if available. If a `SCHEMA` is not supplied or the input is a non-schema based XML, the procedure uses cannonical mapping between elements and object type attributes.

Syntax

```
MEMBER PROCEDURE toObject(
SELF in XMLType,
object OUT "<ADT_1>",
schema in varchar2 := NULL,
element in varchar2 := NULL);
```

Table 319-11 TOOBJECT Subprogram Parameters

Parameter	IN / OUT	Description
`SELF`	`(IN)`	Instance to be converted. Implicit if used as a member procedure.
`object`	`(IN)`	Converted object. An object instance of the required type may be passed in to this function
`schema`	`(IN)`	Schema URL. The mapping of the `XMLType` instance to the converted object instance may be specified using a schema. **Caution**: Oracle does not support use of types generated by Schema Registration (see *Oracle XML DB Developer's Guide*).
`element`	`(IN)`	Top-level element name. An XML Schema document does not specify the top-level element for a conforming XML instance document without this parameter. **Caution**: Oracle does not support use of types generated by Schema Registration (see *Oracle XML DB Developer's Guide*).

TRANSFORM

This member function transforms the XML data using the XSL stylesheet argument and the top-level parameters passed as a string of name=value pairs

If any of the arguments other than the parammap is `NULL`, then a `NULL` is returned.

Syntax

```
MEMBER FUNCTION transform(
xsl IN XMLType,
parammap in varchar2 := NULL)
RETURN XMLType deterministic;
```

Table 319-12 TRANSFORM Subprogram Parameters

Parameter	IN / OUT	Description
xsl	(IN)	The XSL stylesheet describing the transformation
parammap	(IN)	Top level parameters to the XSL - string of name=value pairs

XMLTYPE

This is an `XMLType` constructor.

The options are described in the following table.

Table 319-13 XMLTYPE Member Subprogram Parameters

Syntax	Description

`constructor function XMLType(`
`xmlData IN clob,`
`schema IN varchar2 := NULL,`
`validated IN number := 0,`
`wellformed IN Number := 0)`
`return self as result deterministic;`
	This constructor function creates an optionally schema-based `XMLType` instance using the specified schema and xml data parameters.

`constructor function XMLType(`
`xmlData IN varchar2,`
`schema IN varchar2 := NULL,`
`validated IN number := 0,`
`wellformed IN number := 0)`
`return self as result deterministic;`
	This constructor function creates an optionally schema-based `XMLType` instance using the specified schema and xml data parameters.

`constructor function XMLType (`
`xmlData IN "w<ADT_1>",`
`schema IN varchar2 := NULL,`
`element IN varchar2 := NULL,`
`validated IN number := 0)`
`return self as result deterministic;`
	This constructor function creates an optionally schema-based `XMLType` instance from the specified object type parameter.

`constructor function XMLType(`
`xmlData IN SYS_REFCURSOR,`
`schema in varchar2 := NULL,`
`element in varchar2 := NULL,`
`validated in number := 0)`
`return self as result deterministic;`
	This constructor function creates an optionally schema-based `XMLType` instance from the specified `REF CURSOR` parameter.

Table 319-13 (Cont.) XMLTYPE Member Subprogram Parameters

Syntax	Description

`constructor function XMLType(`
`  xmlData IN AnyData,`
`  schema IN varchar2 := NULL,`
`  element IN varchar2 := NULL,`
`  validated IN number := 0)`
`return self as result deterministic`
`  parallel_enable`
	This constructor function creates an optionally schema-based `XMLType` instance from the specified `ANYDATA` parameter.If the `ANYDATA` instance contains an ADT, the `XMLType` returned is the same as would be returned for a call directly on the ADT. If the `ANYDATA` contains a scalar, the `XMLType` contains a leaf node with the scalar value. The element name for this node is taken from the optional element string if present, and is "`ANYDATA`" if it is not.

`constructor function XMLType(`
`  xmlData IN blob, csid IN number,`
`  schema IN varchar2 := NULL,`
`  validated IN number := 0,`
`  wellformed IN number := 0)`
`return self as result deterministic`
	This constructor function creates an optionally schema-based `XMLType` instance from the specified `BLOB` parameter.

`constructor function XMLType(`
`  xmlData IN bfile,`
`  csid IN number,`
`  schema IN varchar2 := NULL,`
`  validated IN number := 0,`
`  wellformed IN number := 0)`
`return self as result deterministic`
	This constructor function creates an optionally schema-based `XMLType` instance from the specified `BFILE` parameter.

Table 319-14 XMLTYPE Constructor Subprogram Parameters

Parameter	Description
`xmlData`	The data in the form of a `BFILE`, `BLOB`, `CLOB`, `REF`s, `VARCHAR2` or object type.
`schema`	Optional Schema URL to be used to make the input conform to the given schema. **Caution**: Oracle does not support use of types generated by Schema Registration (see *Oracle XML DB Developer's Guide*).
`validated`	Indicates that the instance is valid to the given XML Schema.
`wellformed`	Indicates that the input is well formed. If set, then the database would not do well formed check on the input instance.
`element`	Optional element name in the case of the `ADT_1` or `REF CURSOR` constructors. (Default is `NULL`). **Caution**: Oracle does not support use of types generated by Schema Registration (see *Oracle XML DB Developer's Guide*).
`csid`	The character set id of input XML data.