D

Indexing XML Data for Full-Text Queries (pre-23ai)

When you need full-text search over XML data, Oracle recommends that you store your `XMLType` data as binary XML and you use XQuery Full Text (XQFT). You use an XML search index for this. This is the topic of this section.

XML search indexes can be created using the `SEARCH INDEX FOR XML` syntax or by creating an XQFT enabled `CONTEXT` index. Oracle recommends that you create XML search indexes by using the `SEARCH INDEX FOR XML`.

If portability and standardized code are not a concern, or if your `XMLType` data is stored object-relationally, then you can alternatively use the Oracle-specific full-text constructs and syntax provided by Oracle Text, specifically Oracle SQL function `contains`.

You can perform XQuery Full Text (XQFT) queries on `XMLType` data that is stored as binary XML. If you use an XQFT full-text predicate in an `XMLExists` expression within a SQL `WHERE` clause, then you must create an **XML search index**. This section describes the creation and use of such an index.

- Creating and Using an XML Search Index
  An XQuery Full Text query can use an XML search index to improve performance.
- What To Do If an XML Search Index Is Not Picked Up
  You can modify your query to ensure that certain conditions are satisfied, so its evaluation picks up an XML search index.
- Pragma ora:no_schema: Using XML Schema-Based Data with XQuery Full Text
  Oracle recommends in general that you use *non* XML Schema-based `XMLType` data when you use XQuery Full Text and an XML search index. But you can in some circumstances use XML Schema-based `XMLType` data that is stored as binary XML. Oracle XQuery pragma `ora:no_schema` can be useful in this context.
- Pragma ora:use_xmltext_idx: Forcing the Use of an XML Search Index
  You can use XQuery pragma `ora:use_xmltext_idx` to force the use of an XML search index.
- Migrating from Using Oracle Text Index to XML Search Index
  If you have legacy queries for `XMLType` data stored as binary XML that use SQL function `CONTAINS` and an Oracle Text index that is not XML-enabled, then consider using XQuery Full Text constructs instead.

Related Topics

- Support for XQuery Full Text
  Oracle XML DB supports XQuery Full Text for `XMLType` data that is stored as binary XML. Oracle Text technology provides the full-text indexing and search that is the basis of this support.

> **See Also:**
>
> Example D-6

Creating and Using an XML Search Index

An XQuery Full Text query can use an XML search index to improve performance.

To create an XML search index you must be granted database role `CTXAPP`. More generally, this role is needed to create Oracle Text indexes, to set Oracle Text index preferences, or to use Oracle Text PL/SQL packages.

Before creating the index, you must create an Oracle Text path section group and set its `XML_ENABLE` attribute to `t`. This makes the path section group XML-aware.

For best performance, create an index preference of type `BASIC_STORAGE` in the Oracle Text data dictionary, specifying the following attributes:

- `D_TABLE_CLAUSE` – Specify `SECUREFILE` storage for column `DOC` of index data table `$D`, which contains information about the structure of your XML documents. Specify *caching* and medium *compression*.
- `I_TABLE_CLAUSE` – Specify `SECUREFILE` storage for column `TOKEN_INFO` of index data table `$I`, which contains information about full-text tokens and their occurrences in the indexed documents. Specify *caching* (but not compression).

This is illustrated in Example D-1, which uses a non XML-schema-based `XMLType` table, `po_binxml` (which has the same data as table `purchaseorder` in standard database schema `OE`).

Index preference `BASIC_STORAGE` specifies the tablespace and creation parameters for the database tables and indexes that constitute an Oracle Text index.

> **See Also:**
>
> - *Oracle Text Reference* for information about section groups
> - *Oracle Text Reference* for information about procedure `CTX_DDL.set_sec_grp_attr`
> - *Oracle Text Reference* for information about procedure `CTX_DDL.create_preference`
> - *Oracle Text Reference* for information about procedure `CTX_DDL.set_attribute`
> - *Oracle Text Reference* for information about preference `BASIC_STORAGE`, `D_TABLE_CLAUSE`, and `I_TABLE_CLAUSE`

Example D-2 queries the data to retrieve the `Description` elements whose text contains both `Big` and `Street`, in that order.

Example D-3 shows the execution plan for the query, which indicates that index `po_ctx_idx` is picked up.

Example D-1 Creating an XML Search Index

```
BEGIN
  CTX_DDL.create_section_group('mysecgroup', 'PATH_SECTION_GROUP');
  CTX_DDL.set_sec_grp_attr('mysecgroup', 'XML_ENABLE', 'T');

  CTX_DDL.create_preference('mypref', 'BASIC_STORAGE');
  CTX_DDL.set_attribute('mypref',
                        'D_TABLE_CLAUSE',
                        'TABLESPACE my_ts
                         LOB(DOC) STORE AS SECUREFILE
                         (TABLESPACE my_ts COMPRESS MEDIUM CACHE)');
  CTX_DDL.set_attribute('mypref',
                        'I_TABLE_CLAUSE',
                        'TABLESPACE my_ts
                         LOB(TOKEN_INFO) STORE AS SECUREFILE
                         (TABLESPACE my_ts NOCOMPRESS CACHE)');
END;
/

CREATE INDEX po_ctx_idx ON po_binxml(OBJECT_VALUE)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS('storage mypref section group mysecgroup');
```

Example D-2 XQuery Full Text Query

```
SELECT XMLQuery('for $i in /PurchaseOrder/LineItems/LineItem/Description
                   where $i[. contains text "Big" ftand "Street"]
                return <Title>{$i}</Title>'
                PASSING OBJECT_VALUE RETURNING CONTENT)
  FROM po_binxml
  WHERE XMLExists('/PurchaseOrder/LineItems/LineItem/Description
                   [. contains text "Big" ftand "Street"]'
                  PASSING OBJECT_VALUE);
```

Example D-3 Execution Plan for XQuery Full Text Query

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |     1 |  2014 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| PO_BINXML  |     1 |  2014 |     4   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | PO_CTX_IDX |       |       |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CTXSYS"."CONTAINS"(SYS_MAKEXML(0,"XMLDATA"),'<query><textquery
              grammar="CONTEXT" lang="english"> ( ( {Big} ) and ( {Street} ) )  INPATH
              (/PurchaseOrder/LineItems/LineItem/Description)</textquery></query>')>0)

Note
-----
   - dynamic sampling used for this statement (level=2)
   - Unoptimized XML construct detected (enable XMLOptimizationCheck for more information)

21 rows selected.
```

What To Do If an XML Search Index Is Not Picked Up

You can modify your query to ensure that certain conditions are satisfied, so its evaluation picks up an XML search index.

If you use an XQuery full-text predicate in an `XMLExists` expression within a SQL `WHERE` clause, but you do not create an XML search index or the index cannot be used for some reason, then compile-time error ORA-18177 is raised.

If this error is raised then your execution plan does *not* indicate that the index is picked up. In the plan you do not see operation `DOMAIN INDEX` followed by the name of the index.

In that case, try to change your query to enable the index to be used. The following conditions must both apply for the index to be picked up:

- The expression that computes the XML nodes for the search context must be an XPath expression whose steps are only along *forward* and *descendent axes*.
- You can pass only one `XMLType` instance as a SQL expression in the `PASSING` clause of SQL/XML function `XMLExists`, and each of the other, non-`XMLType` SQL expressions in that clause must be either a *compile-time constant* of a SQL built-in data type or a *bind variable* that is bound to an instance of such a data type.

Pragma ora:no_schema: Using XML Schema-Based Data with XQuery Full Text

Oracle recommends in general that you use *non* XML Schema-based `XMLType` data when you use XQuery Full Text and an XML search index. But you can in some circumstances use XML Schema-based `XMLType` data that is stored as binary XML. Oracle XQuery pragma `ora:no_schema` can be useful in this context.

By default, when an XML search index is used to evaluate XML Schema-based data, compile-time error ORA-18177 is raised. This is because the full-text indexing itself makes no use of the associated XML schema: it is not type-aware. It treats all of the text that it applies to as untyped. This error is raised even if you type-cast data appropriately and thus do not depend on the XML schema to cast types implicitly. Example D-4 illustrates this.

The error raised draws this to your attention, in case you might be expecting a full-text condition in your query to depend on XML Schema types and typed operations.

In order to use a condition that depends on types you must explicitly cast the relevant XQuery expressions to the appropriate types. Do not expect Oracle XML DB to use the XML schema to perform implicit type casting. Failure to type-cast appropriately can lead to results that you might not expect.

Example D-5 shows a query of XML Schema-based data that uses explicit type-casting to ensure that the proper condition is evaluated.

However, most uses of XQuery Full Text expressions, even with XML Schema-based data, do not involve data that is typed. Just remember that if you do use a condition that makes use of typed data then you must cast to the proper type.

In sum, if you are sure that your query does not involve typed data, or if you judge that it is all right to treat particular typed data as if it were untyped, or if you explicitly type-cast any data that needs to be typed, then you can use Oracle XQuery pragma `ora:no_schema` in your query to inhibit raising the error and allow evaluation of the query using an XML search index.

Example D-4 XQuery Full Text Query with XML Schema-Based Data: Error ORA-18177

```
SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem'
                PASSING OBJECT_VALUE RETURNING CONTENT)
  FROM oe.purchaseorder
  WHERE XMLExists('/PurchaseOrder
                   [LineItems/LineItem/@ItemNumber > xs:integer("20")
                    and Actions/Action/User contains text "KPARTNER"]'
                  PASSING OBJECT_VALUE);
  FROM oe.purchaseorder
         *
ERROR at line 3:
ORA-18177: XQuery full text expression '/PurchaseOrder
[LineItems/LineItem/@ItemNumber > xs:integer("20")
and Actions/Action/User contains text "KPARTNER"]'
cannot be evaluated using XML text index
```

Example D-5 Using XQuery Pragma ora:no_schema with XML Schema-Based Data

```
SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem'
                PASSING OBJECT_VALUE RETURNING CONTENT)
  FROM oe.purchaseorder
  WHERE XMLExists('(# ora:no_schema #)
                   {/PurchaseOrder
                    [LineItems/LineItem/@ItemNumber > xs:integer("20")
                     and Actions/Action/User contains text "KPARTNER"]}'
                  PASSING OBJECT_VALUE);
```

Pragma ora:use_xmltext_idx: Forcing the Use of an XML Search Index

You can use XQuery pragma `ora:use_xmltext_idx` to force the use of an XML search index.

A given query involving XML data can be evaluated in various ways, depending on the existence of different indexes and other factors. Sometimes the default evaluation method is not the most performant and it would be more efficient to force the use of an existing XML search index. You can use XQuery pragma `ora:use_xmltext_idx` to do this. (An XML search index applies only to `XMLType` data stored as binary XML.)

For example, a `WHERE` clause might include two `XMLExists` expressions, only one of which involves an XQuery full-text condition, and you might have an `XMLIndex` index that applies to the `XMLExists` expression that has no full-text condition. With such a query it is typically more efficient to use an XML search index to evaluate the entire `WHERE` clause.

Even in some cases where there is no full-text condition in the query, the use of an XML search index can provide the most efficient query evaluation.

The query in Example D-6 illustrates the use of pragma `ora:use_xmltext_idx`. Only the first of the `XMLExists` clauses uses a full-text condition. Because of the pragma, the full-text index (`po_ctx_idx`, created in Example D-1) is used for both `XMLExists` clauses.

Example D-6 Full-Text Query with XQuery Pragma ora:use_xmltext_idx

```
SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem'
                PASSING OBJECT_VALUE RETURNING CONTENT)
  FROM po_binxml
  WHERE XMLExists('/PurchaseOrder/LineItems/LineItem
                   [Description contains text "Picnic"]' PASSING OBJECT_VALUE)
```

```
AND XMLExists('(# ora:use_xmltext_idx #) {/PurchaseOrder[User="SBELL"]}'
              PASSING OBJECT_VALUE);
```

Migrating from Using Oracle Text Index to XML Search Index

If you have legacy queries for `XMLType` data stored as binary XML that use SQL function `CONTAINS` and an Oracle Text index that is not XML-enabled, then consider using XQuery Full Text constructs instead.

The XQuery and XPath Full Text (XQFT) standard is supported by Oracle XML DB starting with Oracle Database 12c Release 1 (12.1). This support applies only to `XMLType` data stored as binary XML. Prior to that release, for full-text querying of XML data you could use only an Oracle Text index that was not XML-enabled (not an XML search index), and your full-text queries necessarily used Oracle-specific constructs: SQL function `CONTAINS`.

If you have legacy code that does this, Oracle recommends that you migrate that code to use XQFT. This section provides information about which XQFT constructs you can use to replace the use of `CONTAINS` .

This use of an Oracle Text index can also be replaced by the use of an XML search index. To replace a query that uses `HASPATH` by one that uses a simple XQuery expression, you use Oracle XQuery pragma `ora:use_xmltext_idx` to specify that the XML search index is to be picked up. This section also illustrates this.

Table D-1 provides a mapping from typical queries that use Oracle-specific constructs to queries that use XQuery Full Text.

Table D-1 Migrating Oracle-Specific XML Queries to XQuery Full Text

Original Example	Replacement Example

`CONTAINS(t.x, 'HASPATH (/P/LIs/LI/Description[1])') > 0`	`XMLExists('(# ora:use_xmltext_idx #) {$d/P/LIs/LI/Description[1]}' PASSING t.x AS "d")`
	
	Or if the data is XML Schema-based:
	
	`XMLExists('(# ora:use_xmltext_idx #) {(# ora:no_schema #) {$d/P/LIs/LI/Description[1]}}' PASSING t.x AS "d")`

Table D-1 (Cont.) Migrating Oracle-Specific XML Queries to XQuery Full Text

Original Example	Replacement Example

`CONTAINS(t.x, 'Big INPATH (/P/LIs/LI/Description)') > 0`	`XMLExists('$d/P/LIs/LI/Description [. contains text "Big"]' PASSING t.x AS "d")`
	
	Or if the data is XML Schema-based:
	
	`XMLExists('(# ora:no_schema #) {$d/P/LIs/LI/Description [. contains text "Big"]}' PASSING t.x AS "d")`

`CONTAINS(t.x, '(Big) AND (Street) INPATH (/P/LIs/LI/Description)') > 0`	`XMLExists('$d/P/LIs/LI/Description [. contains text "Big" ftand "Street"]' PASSING t.x AS "d")`
`CONTAINS(t.x, '(Big) OR (Street) INPATH (/P/LIs/LI/Description)') > 0`	`XMLExists('$d/P/LIs/LI/Description [. contains text "Big" ftor "Street"]' PASSING t.x AS "d")`
`CONTAINS(t.x, '({Big}) NOT ({Street}) INPATH (/P/LIs/LI/Description)') > 0`	`XMLExists('$d/P/LIs/LI/Description [. contains text "Big" ftand ftnot "Street"]' PASSING t.x AS "d")`
`CONTAINS(t.x, '({Street}) MNOT ({Big Street}) INPATH (/P/LIs/LI/Description)') > 0`	`XMLExists('$d/P/LIs/LI/Description [. contains text "Street" not in "Big Street"]' PASSING t.x AS "d")`
`CONTAINS(t.x, '(NEAR (({Big}, {Street}), 3) INPATH (/P/LIs/LI/Description)') > 0`	`XMLExists('$d/P/LIs/LI/Description [. contains text "Big" ftand "Street" window 3 words]' PASSING t.x AS "d")`

Table D-1 (Cont.) Migrating Oracle-Specific XML Queries to XQuery Full Text

Original Example	Replacement Example
(Not applicable – Oracle Text queries are not XML namespace aware.)	`XMLExists('declare namespace ipo="http://www.example.com/IPO"; /ipo:P/ipo:LIs/ipo:LI/ipo:Description [. contains text "Big"]' PASSING t.x AS "d")`

[1] The path test can contain a predicate expression, which is the same for both the original query (with `HASPATH`) and its replacement. For example: `/PurchaseOrder/LineItems/LineItem/Part[@Id < "31415927"]`.