6

SQL Semantics for LOBs

You can use various SQL mechanisms to operate on LOBs.

You can access `CLOB` and `NCLOB` data types using SQL `VARCHAR2` semantics, such as SQL string operators and functions. These techniques allow you to use LOBs directly in SQL code and provide an alternative to using LOB-specific APIs for some operations, and are beneficial in the following situations:

- When performing operations on LOBs that are relatively small in size, i.e., up to about 100K bytes
- After migrating your database from `LONG` columns to LOB data types, so that any SQL string functions contained in your existing PL/SQL application continue to work

SQL semantics are not recommended in the following situations, you must use LOB APIs instead:

- When using advanced features such as random access and piece-wise fetch.
- When performing operations on LOBs that are relatively large in size (greater than 1MB), because using SQL semantics can impact performance.

Note:

SQL semantics are used with persistent and temporary LOBs, and do not apply to BFILEs.

- SQL Functions and Operators Supported for Use with LOBs
  Many SQL operators and functions that take `VARCHAR2` columns as arguments, also accept LOB columns. The following list summarizes those categories of SQL functions and operators that are supported for use with LOBs.
- Detailed Semantics of SQL Operations on LOBs
  This section explains semantics of SQL operations on LOBs in details.
- Restrictions on SQL Operations on LOBs
  There are many SQL operations that are not supported on LOB columns. This section lists those operations.

6.1 SQL Functions and Operators Supported for Use with LOBs

Many SQL operators and functions that take `VARCHAR2` columns as arguments, also accept LOB columns. The following list summarizes those categories of SQL functions and operators that are supported for use with LOBs.

SQL Operations/ Functions	Support
Concatenation	Supported
Comparison	Some comparison functions are not supported for LOBs

SQL Operations/ Functions	Support
Character functions	Supported
Conversion	Some conversion functions are not supported for LOBs
Aggregate functions	Not supported
Unicode functions	Not supported

See Also:

Working with Remote LOBs in SQL and PL/SQL

The following table provides the details on each of the operations that accept VARCHAR2 types as operands or arguments, or return a VARCHAR2 value.

- The SQL column identifies the built-in functions and operators that are supported for CLOB and NCLOB data types. The LENGTH function is also supported for the BLOB data type.
- The PL/SQL column identifies the PL/SQL built-in functions and operators that are supported on LOBs.
- Functions designated as CNV in the SQL or PL/SQL column in the table are performed by converting the CLOB to a character data type, such as VARCHAR2. In the SQL environment, only the first 4K bytes of the CLOB are converted and used in the operation. In the PL/SQL environment, only the first 32K bytes of the CLOB are converted and used in the operation.

Table 6-1 SQL VARCHAR2 Functions and Operators on LOBs

Category	Operator / Function	SQL Example / Comments	SQL	PL/SQL
Concatenation	`||`, `CONCAT()`	`Select clobCol || clobCol2 from tab;`	Yes	Yes
Comparison	`=` , `!=`, `>`, `>=`, `<`, `<=`, `<>`, `^=`	`if clobCol=clobCol2 then...`	No	Yes
Comparison	`IN`, `NOT IN`	`if clobCol NOT IN (clob1, clob2, clob3) then...`	No	Yes
Comparison	`SOME`, `ANY`, `ALL`	`if clobCol < SOME (select clobCol2 from...) then...`	No	N/A
Comparison	`BETWEEN`	`if clobCol BETWEEN clobCol2 and clobCol3 then...`	No	Yes
Comparison	`LIKE [ESCAPE]`	`if clobCol LIKE '%pattern%' then...`	Yes	Yes
Comparison	`IS [NOT] NULL`	`where clobCol IS NOT NULL`	Yes	Yes
Character Functions	`INITCAP`, `NLS_INITCAP`	`select INITCAP(clobCol) from...`	CNV	CNV
Character Functions	`LOWER`, `NLS_LOWER`, `UPPER`, `NLS_UPPER`	`...where LOWER(clobCol1) = LOWER(clobCol2)`	Yes	Yes
Character Functions	`LPAD`, `RPAD`	`select RPAD(clobCol, 20, ' La') from...`	Yes	Yes
Character Functions	`TRIM`, `LTRIM`, `RTRIM`	`...where RTRIM(LTRIM(clobCol,'ab'), 'xy') = 'cd'`	Yes	Yes

Table 6-1 (Cont.) SQL VARCHAR2 Functions and Operators on LOBs

Category	Operator / Function	SQL Example / Comments	SQL	PL/SQL
Character Functions	`REPLACE`	`select REPLACE(clobCol, 'orig','new') from...`	Yes	Yes
Character Functions	`SOUNDEX`	`...where SOUNDEX(clobCOl) = SOUNDEX('SMYTHE')`	CNV	CNV
Character Functions	`SUBSTR`	`...where substr(clobCol, 1,4) = like 'THIS'`	Yes	Yes
Character Functions	`TRANSLATE`	`select TRANSLATE(clobCol, '123abc','NC') from...`	CNV	CNV
Character Functions	`ASCII`	`select ASCII(clobCol) from...`	CNV	CNV
Character Functions	`INSTR`	`...where instr(clobCol, 'book') = 11`	Yes	Yes
Character Functions	`LENGTH`	`...where length(clobCol) != 7;`	Yes	Yes
Character Functions	`NLSSORT`	`...where NLSSORT (clobCol,'NLS_SORT = German') > NLSSORT ('S','NLS_SORT = German')`	CNV	CNV
Character Functions	`INSTRB`, `SUBSTRB`, `LENGTHB`	These functions are supported only for `CLOB`s that use single-byte character sets. (`LENGTHB` is supported for `BLOB`s and `CLOB`s.)	Yes	Yes
Character Functions - Regular Expressions	`REGEXP_LIKE`	This function searches a character column for a pattern. Use this function in the `WHERE` clause of a query to return rows matching the regular expression you specify.	Yes	Yes
Character Functions - Regular Expressions	`REGEXP_REPLACE`	This function searches for a pattern in a character column and replaces each occurrence of that pattern with the pattern you specify.	Yes	Yes
Character Functions - Regular Expressions	`REGEXP_INSTR`	This function searches a string for a given occurrence of a regular expression pattern. You specify which occurrence you want to find and the start position to search from. This function returns an integer indicating the position in the string where the match is found.	Yes	Yes
Character Functions - Regular Expressions	`REGEXP_SUBSTR`	This function returns the actual substring matching the regular expression pattern you specify.	Yes	Yes
Conversion	`CHARTOROWID`	`CHARTOROWID(clobCol)`	CNV	CNV
Conversion	`COMPOSE`	`COMPOSE('string')` Returns a Unicode string given a string in the data type `CHAR`, `VARCHAR2`, `CLOB`, `NCHAR`, `NVARCHAR2`, `NCLOB`.	CNV	CNV
Conversion	`DECOMPOSE`	`DECOMPOSE('str' [CANONICAL | COMPATIBILITY] )` Valid for Unicode character arguments.	CNV	CNV
Conversion	`HEXTORAW`	`HEXTORAW(CLOB)`	No	CNV

Table 6-1 (Cont.) SQL VARCHAR2 Functions and Operators on LOBs

Category	Operator / Function	SQL Example / Comments	SQL	PL/SQL
Conversion	`CONVERT`	`select CONVERT(clobCol,'WE8DEC','WE8HP') from...`	Yes	CNV
Conversion	`TO_DATE`	`TO_DATE(clobCol)`	CNV	CNV
Conversion	`TO_NUMBER`	`TO_NUMBER(clobCol)`	CNV	CNV
Conversion	`TO_TIMESTAMP`	`TO_TIMESTAMP(clobCol)`	No	CNV
Conversion	`TO_MULTI_BYTE` `TO_SINGLE_BYTE`	`TO_MULTI_BYTE(clobCol)` `TO_SINGLE_BYTE(clobCol)`	CNV	CNV
Conversion	`TO_CHAR`	`TO_CHAR(clobCol)`	Yes	Yes
Conversion	`TO_NCHAR`	`TO_NCHAR(clobCol)`	Yes	Yes
Conversion	`TO_LOB`	`INSERT INTO... SELECT TO_LOB(longCol)...` Note that `TO_LOB` can only be used to create or insert into a table with LOB columns as `SELECT FROM` a table with a `LONG` column.	N/A	N/A
Conversion	`TO_CLOB`	`TO_CLOB(varchar2Col)`	Yes	Yes
Conversion	`TO_NCLOB`	`TO_NCLOB(varchar2Clob)`	Yes	Yes
Aggregate Functions	`COUNT`	`select count(clobCol) from...`	No	N/A
Aggregate Functions	`MAX`, `MIN`	`select MAX(clobCol) from...`	No	N/A
Aggregate Functions	`GROUPING`	`select grouping(clobCol) from... group by cube (clobCol);`	No	N/A
Other Functions	`GREATEST`, `LEAST`	`select GREATEST (clobCol1, clobCol2) from...`	No	CNV
Other Functions	`DECODE`	`select DECODE(clobCol, condition1, value1, defaultValue) from...`	CNV	CNV
Other Functions	`NVL`	`select NVL(clobCol,'NULL') from...`	Yes	Yes
Other Functions	`DUMP`	`select DUMP(clobCol) from...`	No	N/A
Other Functions	`VSIZE`	`select VSIZE(clobCol) from...`	No	N/A
Unicode	`INSTR2`, `SUBSTR2`, `LENGTH2`, `LIKE2`	These functions use UCS2 code point semantics.	No	CNV
Unicode	`INSTR4`, `SUBSTR4`, `LENGTH4`, `LIKE4`	These functions use UCS4 code point semantics.	No	CNV
Unicode	`INSTRC`, `SUBSTRC`, `LENGTHC`, `LIKEC`	These functions use complete character semantics.	No	CNV

See Also:

- *Oracle Database SQL Language Reference* for syntax details on SQL functions for regular expressions.
- *Oracle Database Development Guide* for information on using regular expressions with the database.

6.2 Detailed Semantics of SQL Operations on LOBs

This section explains semantics of SQL operations on LOBs in details.

- Return Datatype for SQL Operations on LOBs
  The return data type of SQL functions on LOBs is dependent on the input parameters.
- NULL vs EMPTY LOB: Semantic Difference between LOBs and VARCHAR2
  For the `VARCHAR2` data type, a string of length zero is indistinguishable from a `NULL` value for the column.
- WHERE Clause Usage with LOBs
  SQL functions with LOBs as arguments, except functions that compare LOB values, are allowed in predicates of the `WHERE` clause.
- CLOBs and NCLOBs Do Not Follow Session Collation Settings
  Learn about various operators on `CLOB`s and `NCLOB`s and compare the operations on `VARCHAR2` and `NVARCHAR2` variables with respect to LOBs in this section.
- Codepoint Semantics
  Codepoint semantics of the `INSTR`, `SUBSTR`, `LENGTH`, and `LIKE` functions differ depending on the data type of the argument passed to the function.

6.2.1 Return Datatype for SQL Operations on LOBs

The return data type of SQL functions on LOBs is dependent on the input parameters.

The return type of a function or operator that takes a LOB or `VARCHAR2` is the same as the data type of the argument passed to the function or operator. Functions that take more than one argument, such as `CONCAT`, return a LOB data type if one or more arguments is a LOB.

Example 6-1 `CONCAT` function returning `CLOB`

```
CONCAT(CLOB, VARCHAR2)CLOB
```

Any LOB instance returned by a SQL function is a temporary LOB instance. LOB instances in tables (persistent LOBs) are not modified by SQL functions, even when the function is used in the `SELECT` list of a query.

6.2.2 NULL vs EMPTY LOB: Semantic Difference between LOBs and VARCHAR2

For the `VARCHAR2` data type, a string of length zero is indistinguishable from a `NULL` value for the column.

For the column of a `LOB` data type, there are three possible states:

1. `NULL`: This means the column has no LOB locator.

2. Zero-length value: This can be achieved by inserting an `EMPTY LOB` into the column, or by using an API such as `DBMS_LOB.TRIM()` to trim the length to zero. In either case, there is a valid LOB locator in the column, but the LOB value length is zero.
3. Non-zero length value.

Due to this difference, the `LENGTH` function differs depending on whether the argument passed is a `LOB` or a character string:

- For a character string of length zero, the `LENGTH` function returns `NULL`.
- For a `CLOB` of length zero, or an empty locator such as that returned by `EMPTY_CLOB()`, the `LENGTH` and `DBMS_LOB.GETLENGTH` functions return 0.

Similarly, when used with LOBs, the `IS NULL` and `IS NOT NULL` operators determine whether a LOB locator is stored in the row:

- When you pass an initialized LOB of length zero to the `IS NULL` function, `FALSE` is returned. These semantics are compliant with the SQL 92 standard.
- When you pass a `VARCHAR2` of length zero to the `IS NULL` function, `TRUE` is returned.

6.2.3 WHERE Clause Usage with LOBs

SQL functions with LOBs as arguments, except functions that compare LOB values, are allowed in predicates of the `WHERE` clause.

The `LENGTH` function, for example, can be included in the predicate of the `WHERE` clause:

```
CREATE TABLE t (n NUMBER, c CLOB);
INSERT INTO t VALUES (1, 'abc');

SELECT * FROM t WHERE c IS NOT NULL;
SELECT * FROM t WHERE LENGTH(c) > 0;
SELECT * FROM t WHERE c LIKE '%a%';
SELECT * FROM t WHERE SUBSTR(c, 1, 2) LIKE '%b%';
SELECT * FROM t WHERE INSTR(c, 'b') = 2;
```

6.2.4 CLOBs and NCLOBs Do Not Follow Session Collation Settings

Learn about various operators on `CLOB`s and `NCLOB`s and compare the operations on `VARCHAR2` and `NVARCHAR2` variables with respect to LOBs in this section.

Standard operators that operate on `CLOB`s and `NCLOB`s without first converting them to `VARCHAR2` or `NVARCHAR2`, are marked as 'Yes' in the SQL or PL/SQL columns of Table 7-1. These operators do not behave linguistically, except for `REGEXP` functions. Binary comparison of the character data is performed irrespective of the `NLS_COMP` and `NLS_SORT` parameter settings.

These `REGEXP` functions are the exceptions, where, if `CLOB` or `NCLOB` data is passed in, the linguistic comparison is similar to the comparison of `VARCHAR2` and `NVARCHAR2` values.

- `REGEXP_LIKE`
- `REGEXP_REPLACE`
- `REGEXP_INSTR`
- `REGEXP_SUBSTR`
- `REGEXP_COUNT`

Note:

CLOBs and NCLOBs support the default USING NLS_COMP option.

See Also:

Oracle Database Reference for more information about `NLS_COMP`

6.2.5 Codepoint Semantics

Codepoint semantics of the `INSTR`, `SUBSTR`, `LENGTH`, and `LIKE` functions differ depending on the data type of the argument passed to the function.

These functions use different codepoint semantics depending on whether the argument is a `VARCHAR2` or a `CLOB` type as follows:

- When the argument is a `CLOB`, UCS2 codepoint semantics are used for all character sets.
- When the argument is a character type, such as `VARCHAR2`, the default codepoint semantics are used for the given character set:
  - UCS2 codepoint semantics are used for AL16UTF16 and UTF8 character sets.
  - UCS4 codepoint semantics are used for all other character sets, such as AL32UTF8.
- If you are storing character data in a `CLOB` or `NCLOB`, then note that the amount and offset parameters for any APIs that read or write data to the `CLOB` or `NCLOB` are specified in UCS2 codepoints. In some character sets, a full character consists one or more UCS2 codepoints called a surrogate pair. In this scenario, you must ensure that the amount or offset you specify does not cut into a full character. This avoids reading or writing a partial character.
- Oracle Database helps to detect half surrogate pair on read or write boundaries in case of SQL functions and in case of read/write through LOB APIs. The behavior is as follows:
  - If the starting offset is in the middle of a surrogate pair, an error is raised for both read and write operations.
  - If the read amount reads only a partial character, increment or decrement the amount by 1 to read complete characters.

    Note:

    The output amount may vary from the input amount.

  - If the write amount overwrites a partial character, an error is raised to prevent the corruption of existing data caused by overwriting of a partial character in the destination `CLOB` or `NCLOB`.

Note:

This check only applies to the existing data in the `CLOB` or `NCLOB`. You must make sure that the incoming buffer for the write operation starts and ends in complete characters.

6.3 Restrictions on SQL Operations on LOBs

There are many SQL operations that are not supported on LOB columns. This section lists those operations.

Table 6-2 Unsupported Usage of LOBs in SQL

SQL Operations Not Supported	Example of unsupported usage
`SELECT DISTINCT`	`SELECT DISTINCT clobCol from...`
`SELECT` clause `ORDER BY`	`SELECT... ORDER BY clobCol`

`SELECT` clause
`GROUP BY`
	`SELECT avg(num) FROM...`
	`GROUP BY clobCol`

`UNION`, `INTERSECT`, `MINUS` (Note that `UNION ALL` works for LOBs.)	`SELECT clobCol1 from tab1 UNION SELECT clobCol2 from tab2;`
Join queries	`SELECT... FROM... WHERE tab1.clobCol = tab2.clobCol`
Index columns	`CREATE INDEX clobIndx ON tab(clobCol)...`

Related Topics

- BFILE APIs
  This section discusses the different operations supported through `BFILE`s.