3
Temporary LOBs

Temporary LOBs are transient, just like other local variables in an application. This chapter discusses operations that are specific to temporary LOBs.

- Before You Begin
  Ensure that you go through the topics in this section before you start working with temporary LOBs.
- Temporary LOB APIs in Different Programmatic Interfaces
  This section lists the temporary LOB specific APIs in different Programmatic Interfaces.
- Transforming LOBs
  Value LOBs are a subset of read-only temporary LOBs, which are a subset of temporary LOBs.

3.3 Transforming LOBs

Value LOBs are a subset of read-only temporary LOBs, which are a subset of temporary LOBs.

The following table summarizes how you can transform different kinds of LOBs. The column on the left is the source LOB which you want to transform. The column headings are the target LOBs, the final state of the LOB after the transformation. For example, to transform read-only temporary LOBs to value LOBs, you can use `LOB_VALUE(lob_producing_plsql_function(...))`.

Source LOB	Value LOBs	Read-only Temporary LOBs	Temporary LOBs	Persistent LOBs
Value LOBs	Not applicable	• Pass from SQL to PLSQL. If passed from PLSQL out to JDBC, OCI etc, it stays a read-only temporary LOB. • Send to older JDBC, OCI etc clients	Not directly possible	Not directly possible
Read-only Temporary LOBs	`LOB_VALUE(lob_producing_plsql_function(...))`	Not applicable	Not directly possible	Not directly possible
Temporary LOBs	`LOB_VALUE(`*`temporary_lob`*`)`	Open in READ mode	Not applicable	Not directly possible
Persistent LOBs	• SELECT from column declared as QUERY AS VALUE • LOB_VALUE(persistent_lob)	Not directly possible	Use SQL operators, such as `to_clob()` or `substr()`	Not applicable

The following example shows how you can transform temporary LOBs and read-only temporary LOBs to value LOBs. Also, how you can transform a value LOB to a read-only temporary LOB.

```
DROP TABLE t;

CREATE TABLE t (c clob) lob(c) query as value;
INSERT INTO t VALUES ('I am a CLOB');

CREATE OR REPLACE FUNCTION Vbl2rdo (c clob) RETURN clob IS
BEGIN
    RETURN c;
END;

/
-- Transform value LOB to read-only temporary LOB
var tc clob;
BEGIN
    SELECT c INTO :tc FROM t;
END;
/

print :tc
SELECT Vbl2rdo(c) FROM t;

-- Transform read-only temporary LOB to value LOB
SELECT lob_value(:tc) FROM dual;

-- Transform temporary LOB to value LOB
SELECT lob_value(to_clob('I am a temporary LOB')) FROM dual;
```

The following example shows how you can transform a persistent LOB to a temporary LOB and a value LOB.

```
DROP TABLE t2;

CREATE TABLE t2 (c CLOB);

INSERT INTO t2 VALUES ('I am a CLOB');

-- Transform persistent LOB to value LOB
SELECT Lob_value(c) FROM t2;

-- Transform persistent LOB to temporary LOB
SELECT To_clob(c) FROM t2;
```

3.1 Before You Begin

Ensure that you go through the topics in this section before you start working with temporary LOBs.

- Creating Temporary LOBs
  This section describes how a temporary LOB gets created or generated in a client program.

- Handling Temporary LOBs on the Client Side
  You must consider the aspects discussed in this section while handling the temporary LOBs that are generated by the client programs.

3.1.1 Creating Temporary LOBs

This section describes how a temporary LOB gets created or generated in a client program.

You can create temporary LOB instances in one of the following ways:

- Declare a variable of the given LOB data type and pass it to the temporary LOB creation API. For example, in PL/SQL it is `DBMS_LOB.CREATETEMPORARY`, and in OCI it is `OCILobCreateTemporary()`.
- Invoke a SQL or PL/SQL built-in function that produces a temporary LOB, for example, the `SUBSTR` function.
- Invoke a PL/SQL stored procedure or function that returns a temporary LOB as an `OUT` bind variable or a return value.

The temporary LOB instance exists in your application until it goes out of scope, your session terminates, or you explicitly free the instance.

Temporary LOBs reside in either the PGA memory or the temporary tablespace, depending on their size. Ensure that the PGA memory and the temporary tablespace have space that is large enough for the temporary LOBs used by your application.

> **Note:**
>
> - Oracle highly recommends that you release the temporary LOB instances to free the system resources. Failure to do so may cause accumulation of temporary LOBs and can considerably slow down your system.
> - Starting with Oracle Database Release 21c, you do not need to check whether a LOB is temporary or persistent before releasing the temporary LOB. If you call the `DBMS_LOB.FREETEMPORARY` procedure or the `OCILobFreeTemporary()` function on a LOB, it will perform either of the following operations:
>   - For a temporary LOB, it will release the LOB.
>   - For a persistent LOB, it will do nothing (no-op).

> **See Also:**
>
> Performance Guidelines

3.1.2 Handling Temporary LOBs on the Client Side

You must consider the aspects discussed in this section while handling the temporary LOBs that are generated by the client programs.

Preventing Temporary LOB Accumulation

Every time a client program such as JDBC or OCI obtains a LOB locator from SQL or PL/SQL, and you suspect that it is producing a temporary LOB, then free the LOB as soon as your application has consumed the LOB. If you do not free the temporary LOB, then it will lead to accumulation of temporary LOBs, which can considerably slow down your system.

> **Note:**
>
> A temporary LOB duration is always upgraded to `SESSION`, when it is shipped to the client side.

For example, to prevent temporary LOB accumulation, an OCI application must call the `OCILobFreeTemporary()` function in the following scenarios:

- After getting a locator from a define during a `SELECT` statement or an `OUT` bind variable from a PL/SQL procedure or function. It is desirable that you free the temporary LOB as soon as you finish performing the required operations on it. If not, then you must free it before reusing the variable for fetching the next row or for another purpose.
- Before performing a pointer assignment, like `<var1 = var2>`, free the old temporary LOB in the variable `<var1>`.

LOB Assignment
You must take special care when assigning the `OCILobLocator` pointers in an OCI program while using the assignment (=) operator. Pointer assignments create a shallow copy of the LOB. After the pointer assignment, the source and the target LOBs point to the same copy of data. This means that if you call the `OCILobFreeTemporary()` function on either one of them, then both variables will point to non-existent LOBs.

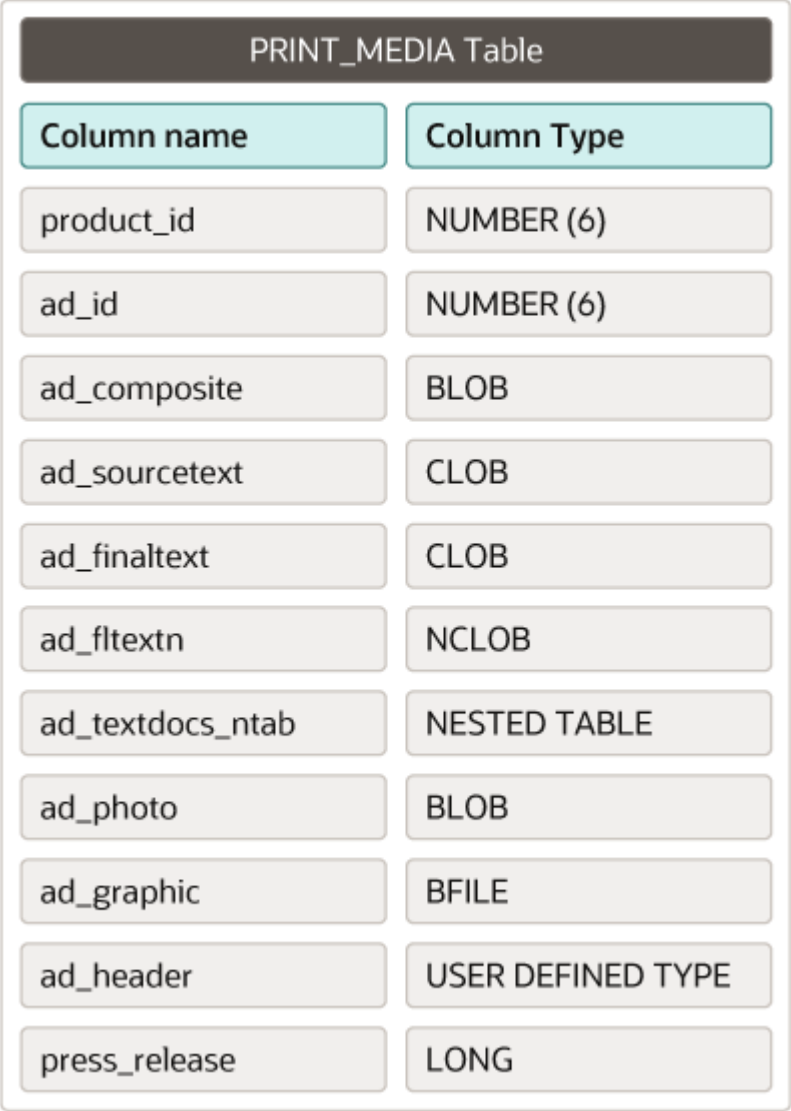
These semantics are different from using the LOB APIs, such as the `OCILobLocatorAssign()` function to perform assignments. When you use these APIs, the locators logically point to independent copies of data after assignment. This means that eventually the `OCILobFreeTemporary()` function must be called on each LOB descriptor separately, so that it frees all LOBs involved in the operation.

For temporary LOBs, before performing pointer assignments, you must ensure that you free any temporary LOB in the target LOB locator by calling the `OCIFreeTemporary()` function. In contrast, when the `OCILobLocatorAssign()` function is used, the original temporary LOB in the target LOB locator variable, if any, is freed automatically before the assignment happens.

3.2 Temporary LOB APIs in Different Programmatic Interfaces

This section lists the temporary LOB specific APIs in different Programmatic Interfaces.

Most of the examples in the following sections use the `print_media` table. Following is the structure of the `print_media` table.



PRINT_MEDIA Table

Column name	Column Type
product_id	NUMBER (6)
ad_id	NUMBER (6)
ad_composite	BLOB
ad_sourcetext	CLOB
ad_finaltext	CLOB
ad_fltextn	NCLOB
ad_textdocs_ntab	NESTED TABLE
ad_photo	BLOB
ad_graphic	BFILE
ad_header	USER DEFINED TYPE
press_release	LONG

- PL/SQL APIs for Temporary LOBs
  This section describes the PL/SQL APIs used with temporary LOBs.
- JDBC API for Temporary LOBs
  This section describes the PL/SQL APIs used with temporary LOBs.
- OCI APIs for Temporary LOBs
  This section describes the OCI APIs used with temporary LOBs.
- ODP.NET API for Temporary LOBs
  This section describes the ODP.NET APIs used with temporary LOBs.
- Pro*C/C++ and Pro*COBOL APIs for Temporary LOBs
  This section describes the Pro*C/C++ and Pro*COBOL APIs for Temporary LOBs.

See Also:

Comparing the LOB Interfaces

3.2.1 PL/SQL APIs for Temporary LOBs

This section describes the PL/SQL APIs used with temporary LOBs.

> **See Also:**
>
> DBMS_LOB

Table 3-1 DBMS_LOB Functions and Procedures for Temporary LOBs

Function / Procedure	Description
CREATETEMPORARY	Creates a Temporary LOB
ISTEMPORARY	Checks if a LOB locator refers to a temporary LOB
FREETEMPORARY	Frees a temporary LOB

Example 3-1 PL/SQL API for Temporary LOBs

```
DECLARE
  blob1 BLOB;
  clob1 CLOB;
  clob2 CLOB;
  nclob1 NCLOB;
BEGIN
  -- create a temp LOB using CREATETEMPORARY and fill it with data
  DBMS_LOB.CREATETEMPORARY(blob1,TRUE, DBMS_LOB.SESSION);
  writeDataToLOB_proc(blob1);

  -- create a temp LOB using SQL built-in function
  SELECT substr(ad_sourcetext, 5) INTO clob1 FROM print_media WHERE
product_id=1 AND ad_id=1;

  -- create a temp LOB using a PLSQL built-in function
  nclob1 := TO_NCLOB(clob1);

  -- create a temp LOB using a PLSQL procedure. Assume foo creates a temp lob
and it's parameter is IN/OUT
  foo(clob2);

  -- Other APIs
  CALL_LOB_APIS(blob1, clob1, clob2, nclob1);

  -- free temp LOBs
  DBMS_LOB.FREETEMPORARY(blob1);
  DBMS_LOB.FREETEMPORARY(clob1);
  DBMS_LOB.FREETEMPORARY(clob2);
  DBMS_LOB.FREETEMPORARY(nclob1);

END;
/
show errors;
```

3.2.2 JDBC API for Temporary LOBs

This section describes the PL/SQL APIs used with temporary LOBs.

> **See Also:**
>
> Working with LOBs and BFILEs

Table 3-2 jdbc.sql.Clob and java.sql.Blob APIs for Temporary LOBs

Methods	Description
`createTemporary`	Creates a temporary LOB
`isTemporary`	Checks if a LOB locator refers to a temporary LOB
`freeTemporary`	Frees a temporary LOB

Example 3-2 JDBC API for Temporary LOBs

```
public class listempc
{
  public static void main (String args [])
    throws Exception
  {
    Connection conn = LobDemoConnectionFactory.getConnection();

    // SELECT TEMPORARY LOB USING SQL
    Statement stmt = conn.createStatement ();
    ResultSet rset = stmt.executeQuery
          ("SELECT SUBSTR(ad_sourcetext, 5) FROM Print_media WHERE product_id
= 3106 AND ad_id = 1");
    if (rset.next())
    {
      Clob clob = rset.getClob (1);
      System.out.println("Is lob temporary: " + ((CLOB)clob).isTemporary());

      call_other_apis_to_read_write_from_lob(clob);
      clob.free();
    }
    stmt.close();

    // CREATE TEMPORARY LOB VIA API
    Clob clob = conn.createClob();

    System.out.println( "Is clob temporary: " +
((oracle.jdbc.OracleClob)clob).isTemporary());

    call_other_apis_to_read_write_from_lob(clob);

     // ALWAYS FREE THE TEMPORARY LOB WHEN DONE WITH IT
    clob.free();

    conn.close();
```

```
  }
}
```

3.2.3 OCI APIs for Temporary LOBs

This section describes the OCI APIs used with temporary LOBs.

> **See Also:**
>
> LOB and BFILE Operations

Table 3-3 OCI APIs for Temporary LOBs

Function / Procedure	Description
`OCILobCreateTemporary()`	Creates a Temporary LOB
`OCILobIsTemporary()`	Checks if a LOB locator refers to a temporary LOB
`OCILobFreeTemporary()`	Frees a temporary LOB

Example 3-3 OCI APIs for Temporary LOBs

```
void temp_lob_operations()
{
  OCILobLocator *temp_clob1;
  OCILobLocator *temp_clob2;
  OCIStmt       *stmhp = (OCIStmt *) 0;
  OCIDefine     *dfnhp1;
  ub1            bufp[BUFLEN];
  ub4            amtp = 0;
  ub8            bamtp = 0;
  ub8            camtp = 0;
  ub2            retl1, rcode1;
  sb4            ind_ptr1 = 0;
  boolean        istemp = FALSE;
  char          *sel_stmt = "SELECT SUBSTR(ad_sourcetext, 5) FROM Print_media
WHERE product_id = 3106 AND ad_id = 1";

  /* allocate lob descriptors */
  checkerr(errhp, OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &temp_clob1,
                                     (ub4) OCI_DTYPE_LOB, (size_t) 0,
                                     (dvoid **) 0));
  checkerr(errhp, OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &temp_clob2,
                                     (ub4) OCI_DTYPE_LOB, (size_t) 0,
                                     (dvoid **) 0));

  /* statement handle */
  checkerr(errhp, OCIHandleAlloc( (dvoid *)envhp, (dvoid **) &stmhp,
                   (ub4) OCI_HTYPE_STMT, (size_t) 0, (dvoid **) 0));
  checkerr(errhp, OCIHandleAlloc( (dvoid *)stmhp, (dvoid **) &dfnhp1,
                   (ub4) OCI_HTYPE_DEFINE, (size_t) 0, (dvoid **) 0));

  /*-------------------- SELECT TEMPORARY LOB USING SQL
------------------------*/
```

```
  checkerr(errhp, OCIStmtPrepare(stmhp, errhp, (text *) sel_stmt,
          (ub4) strlen(sel_stmt), OCI_NTV_SYNTAX, OCI_DEFAULT));

  checkerr(errhp, OCIDefineByPos(stmhp, &dfnhp1, errhp, (ub4) 1, &temp_clob1,
                  (sb4) -1, SQLT_CLOB, &ind_ptr1, &retl1, &rcode1,
                  (ub4) OCI_DEFAULT));

  checkerr(errhp, OCIStmtExecute(svchp, stmhp, errhp, (ub4) 0, (ub4) 0,
                     (OCISnapshot *) NULL, (OCISnapshot *) NULL,
OCI_DEFAULT));
  checkerr(errhp, OCIStmtFetch(stmhp, errhp, 1, OCI_FETCH_NEXT, OCI_DEFAULT));

  checkerr(errhp, OCILobWriteAppend2(svchp, errhp, temp_clob1,
           (oraub8 *)&bamtp, (oraub8 *) &camtp, bufp, (oraub8)BUFLEN,
            OCI_ONE_PIECE, (dvoid*)0, (OCICallbackLobWrite2)0, (ub2)0,
            (ub1)SQLCS_IMPLICIT));

  /*--------------------- CREATE TEMPORARY LOB USING API
-------------------------*/
  checkerr(errhp, OCILobCreateTemporary(svchp, errhp, temp_clob2,
                  (ub2) 0, OCI_DEFAULT, OCI_TEMP_CLOB,
                  FALSE, OCI_DURATION_SESSION));

  /* write into bufp */
  strcpy((char *)bufp, (const char *)"Demo program for testing temp lobs");
  bamtp = amtp = (ub4) strlen((char *)bufp);

  /* write bufp contents to temp lob */
  checkerr(errhp, OCILobWrite2(svchp, errhp, temp_clob2, &amtp, 1,
                  (dvoid *)bufp, (ub4)bamtp , OCI_ONE_PIECE, (dvoid *)0,
                  (OCICallbackLobWrite) 0, (ub2) 0, (ub1) SQLCS_IMPLICIT));

  /*--------------------- ALWAYS FREE TEMPORARY LOBS -------------------------
*/
  checkerr(errhp, OCILobIsTemporary(envhp, errhp, temp_clob1, &istemp));
  if (istemp)
    checkerr(errhp, OCILobFreeTemporary(svchp, errhp, temp_clob1));

  checkerr(errhp, OCILobIsTemporary(envhp, errhp, temp_clob2, &istemp));
  if (istemp)
    checkerr(errhp, OCILobFreeTemporary(svchp, errhp, temp_clob2));

 /* Free lob descriptors */
  checkerr(errhp, OCIDescriptorFree ((dvoid *)temp_clob1, (ub4)
OCI_DTYPE_LOB));
  checkerr(errhp, OCIDescriptorFree ((dvoid *)temp_clob2, (ub4)
OCI_DTYPE_LOB));
}
```

3.2.4 ODP.NET API for Temporary LOBs

This section describes the ODP.NET APIs used with temporary LOBs.

> **See Also:**
>
> Temporary LOBs

Table 3-4 ODP.NET methods for Temporary LOBs in the OracleClob and OracleBlob Classes

Methods	Description
`Add()`	Creates a temporary LOB
`IsTemporary()`	Checks if a LOB locator refers to a temporary LOB
`Dispose()` or `Close()`	Frees a temporary LOB

3.2.5 Pro*C/C++ and Pro*COBOL APIs for Temporary LOBs

This section describes the Pro*C/C++ and Pro*COBOL APIs for Temporary LOBs.

> **See Also:**
>
> - Pro*C/C++ Programmer's Guide
> - Pro*COBOL Programmer's Guide

Table 3-5 Pro*C/C++ and Pro*COBOL APIs for Temporary LOBs

Statement	Description
`CREATE TEMPORARY`	Creates a Temporary LOB
`DESCRIBE [ISTEMPORARY]`	Checks if a LOB locator refers to a temporary LOB
`FREE TEMPORARY`	Frees a temporary LOB