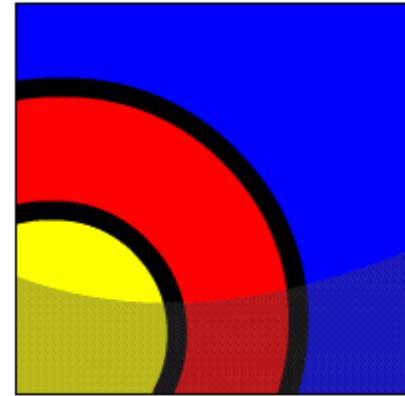


Using Large Object Data Types

What Will I Learn?

In this lesson, you will learn to:

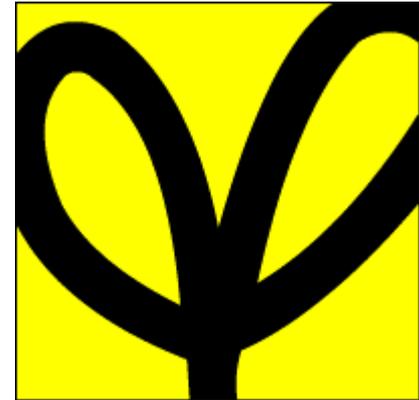
- Describe LOB (Large Object) data types and how they are used
- Differentiate between internal and external LOBs
- Compare and contrast LONG and LOB data types
- Create and maintain LOB datatypes
- Migrate data from LONG to LOB





Why Learn It?

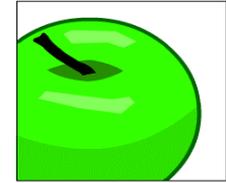
Suppose you want to add new columns to your EMPLOYEES table to store an employee badge photo, an annual performance evaluation, and a video clip of the employee from the last Company picnic.



This new data needs a lot of storage space in the database; far more than you can fit into a VARCHAR2 or RAW column.

You need extra column data types that can store larger data values.

Tell Me/Show Me



New Columns for EMPLOYEES

- The HR department wants to add a column that contains all annual performance evaluations.
 - Stored as free-form text, it should be searchable in a similar fashion to Google. Assume a typewritten page is 100x100 characters (10 KB) and it could be up to 100 pages (1 MB).
- The Security department wants to add a column that contains a photo for their ID badges.
 - Dimensions are small, passport or driver's license size
 - The JPEG photo files are approximately 15 KB each.
- The Advertising department took videos of the last Company awards ceremony, and wants to add short video clips of the employees receiving awards.
 - They are stored on DVDs. Most are under 5 minutes.



Tell Me/Show Me

You Need Large Object (LOB) Column Data Types

- The problem:
 - In SQL, the largest character column is 4096 bytes.
 - There is no specific datatype for MP3, JPEG, EXE, and so on
 - What if there is a need to store an object bigger than 4 KB?
- The solution:
 - Large Objects (LOBs) address all of these problems.
 - They can store *ANYTHING* of any type.
 - A single LOB field in a table can be up to 4 GB in version 9i, and up to 128 TB in version 10g.



Tell Me/Show Me

Two Ways to Store Large Objects: The Old and the New Way

- The old way:
 - There are two deprecated data types: `LONG` and `LONG RAW`.
 - They should not be used any more
- The new way:
 - LOBs come in four flavors: `CLOB`, `BLOB`, `BFILE`, and `NCLOB`.
 - `CLOB`: Character Large Objects, such as resumes, text articles, and source code files.
 - `BLOB`: Binary Large Objects, such as sound (MP3), photos (JPEG, BMP), proprietary formats (PDF, DOC, XLS), and executables (EXE, DDL).
 - `BFILE`: Binary Files, just like `BLOB` but stored outside the database, often on separate media (CD, DVD, HD-DVD).
 - `NCLOB`: National Character Set Large Objects used with multi-byte alphabets.



Tell Me/Show Me

The Old Way

- **LONG**
 - Up to 2 GB
 - Replaced by **CLOB**
- **LONG RAW**
 - Up to 2 GB
 - Replaced by **BLOB** and **BFILE**

The New Way

- **CLOB**
 - Up to 4 GB or 128 TB
 - Replaces **LONG**
- **BLOB** and **BFILE**
 - Up to 4 GB or 128 TB
 - Replace **LONG RAW**
 - Inside or outside the database

A table can contain only one **LONG** or **LONG RAW** column, but as many **LOB** columns as needed.

CLOB and **BLOB** data are stored in the database (“internal **LOBs**”). **BFILES** are stored outside the database (“external **LOBs**”) in separate files.



Tell Me/Show Me

Converting LONG to CLOB

You can convert LONG columns to CLOBs (and LONG RAW columns to BLOBs) using ALTER TABLE:

```
ALTER TABLE table_name
  MODIFY (long_col_name {CLOB | BLOB});
```

For example, to convert the RESUMES column in the EMPLOYEES table from LONG to CLOB, you enter:

```
ALTER TABLE employees
  MODIFY (resumes CLOB);
```

Tell Me/Show Me

CLOB column

- Text only:
 - No fonts, no bold, no italic, no formatting, nothing fancy.
 - Useful for storing XML, HTML, DDL, PL/SQL, scripts and other source code for programs
 - Can use all the built-in SQL character functions, such as SUBSTR, LENGTH.

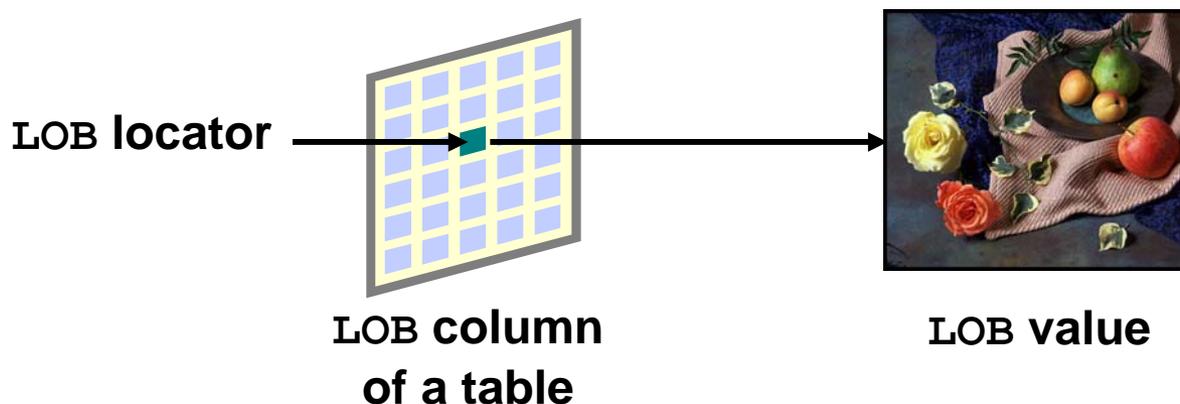
BLOB column

- Can store absolutely anything (for example, ZIP, EXE, and DLL file types).
- A few file types, such as PDF, BMP, GIF, JPEG, MP3, and WAV (usually the ones known by web browsers) can be displayed by Oracle Application Express.

Tell Me/Show Me

How and Where Is LOB Data Stored?

Unlike other data types in which the column value is stored inline as part of the main row data, a LOB column value is stored in a separate area of the database with a pointer to it from the main table row.



You say that the LOB data value is stored out-of-line. The pointer from the main row is called a locator.



Tell Me/Show Me

Adding a LOB Column to a Table

```
ALTER TABLE employees ADD (annual_evals CLOB);  
  
ALTER TABLE employees ADD (badge_photo BLOB);
```

Like any other newly added column, the LOB value is NULL. This means that the column data does not exist, and the locator is set to NULL.



Tell Me/Show Me

Initializing a LOB Column

Because a LOB column is in two parts (the locator that points to the value), you must initialize the locator before you can insert the data value:

```
UPDATE employees
   SET annual_evals = EMPTY_CLOB(),
       badge_photo   = EMPTY_BLOB();
```

EMPTY_CLOB and EMPTY_BLOB are built-in SQL functions, just like UPPER, TO_CHAR, and so on, except that they can only be used in DML statements because they modify the table. The functions allocate initial space elsewhere in the database to hold the data value, and update the locator to point to this space.



Tell Me/Show Me

Populating a CLOB Column with Data

Now you can insert the data values using normal DML statements:

```
UPDATE employees
  SET annual_evals = 'Evaluation Date: 14 September
                    2005. Performance Rating: Good ... '
  WHERE employee_id = 100;
```

This two-step method (initializing, then populating) is necessary because the data is stored out-of-line, and you cannot access it in any way until you have created a pointer to it (that is, initialized the locator).



Tell Me/Show Me

Reading CLOB Data From the Table

You can `SELECT` a CLOB column like any other column:

```
SELECT annual_evals FROM employees
WHERE employee_id = 100 ;
```

But these values can be large. Reading the whole of a 4 GB CLOB value takes a long time and uses a lot of memory. And maybe you only want to see part of the value anyway:

```
SELECT SUBSTR(annual_evals,2000,1000)
FROM employees WHERE employee_id = 100;
```



Tell Me/Show Me

Updating CLOB Data

But you cannot use SQL functions, such as SUBSTR in an UPDATE statement:

```
UPDATE employees
  SET substr(annual_evals,2001,8) = 'NEW TEXT'
  WHERE employee_id = 100;
```

Instead, you must use the DBMS_LOB PL/SQL package. And you cannot do this directly in an SQL DML statement. You can do it only from inside a PL/SQL block.

The next slide shows how.



Tell Me/Show Me

Updating CLOB Data Using DBMS_LOB

```
DECLARE
    v_lobloc    CLOB;    -- this will store the LOB locator
    v_new_text  VARCHAR2(32767) := 'NEW TEXT';
    v_amount    INTEGER;
    v_offset    INTEGER;
BEGIN
    SELECT annual_evals INTO v_lobloc
    FROM employees
        WHERE employee_id = 100
        FOR UPDATE;
    v_offset := DBMS_LOB.GETLENGTH(v_lobloc) + 2;
    v_amount := LENGTH(v_new_text);
    DBMS_LOB.WRITE(v_lobloc,v_amount,v_offset,v_new_text);
END;
```



Tell Me/Show Me

Populating a CLOB Column With a Large Value Using DBMS_LOB

You have already learned that you can populate a CLOB column with a DML UPDATE statement:

```
UPDATE employees
  SET annual_evals = 'Evaluation Date: 14 September
                    2005. Performance Rating: Good ... '
 WHERE employee_id = 100;
```

But what if the value is large? A character literal in an SQL statement cannot be 4 GB in size!

Again, you can use DBMS_LOB to load the value one piece at a time. The next slide shows how.



Tell Me/Show Me

Populating a CLOB Column Using DBMS_LOB

```
DECLARE
  v_lobloc    CLOB;    -- this will store the LOB locator
  v_text      VARCHAR2(32767);
  v_length    INTEGER;
  v_offset    INTEGER;
BEGIN
  SELECT annual_evals INTO v_lobloc FROM employees
 WHERE employee_id = 100 FOR UPDATE;
  FOR i IN 1..3 LOOP
    v_text      := 'The next piece of text number ' || i;
    v_offset    := DBMS_LOB.GETLENGTH(v_lobloc)+ 2;
    v_length    := LENGTH(v_text);
    DBMS_LOB.WRITE(v_lobloc,v_length,v_offset,v_text);
  END LOOP;
END;
```



Tell Me/Show Me

Reading BLOB Column Data Using DBMS_LOB

BLOB data cannot be displayed in Oracle Application Express, but you can see that the data exists by finding and displaying its length:

```
DECLARE
  CURSOR country_curs IS
    SELECT country_id, country_name, flag
      FROM wf_countries WHERE country_name LIKE 'A%';
  v_length      NUMBER;
BEGIN
  FOR country_rec IN country_curs LOOP
    v_length := DBMS_LOB.GETLENGTH(country_rec.flag);
    DBMS_OUTPUT.PUT_LINE(country_rec.country_id || ' '
      || country_rec.country_name || ' ' || v_length);
  END LOOP;
END;
```



Tell Me/Show Me

Reading BLOB Column Data Using DBMS_LOB (continued)

The following is the output from the previous slide. Notice that Antarctica does not have a flag!

Results Explain Describe Saved SQL History

```
672 Antarctica 0
20 Arab Republic of Egypt 489
297 Aruba 604
1268 Antiqua and Barbuda 769
54 Argentine Republic 1270
1264 Anguilla 1431
```

```
Statement processed.
```

Tell Me/Show Me

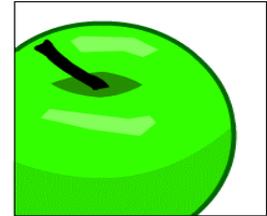
Terminology

Key terms used in this lesson include:

CLOB

BLOB

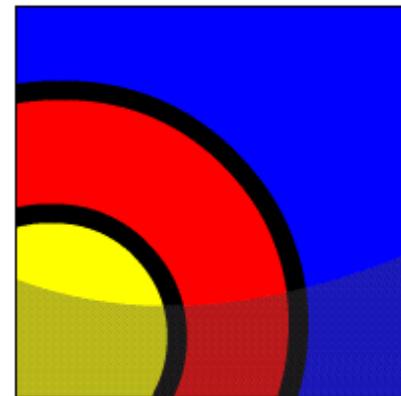
BFILE



Summary

In this lesson, you learned to:

- Describe LOB (Large Object) data types and how they are used
- Differentiate between internal and external LOBs
- Compare and contrast LONG and LOB data types
- Create and maintain LOB datatypes
- Migrate data from LONG to LOB





Try It/Solve It

This practice covers the following topics:

- Creating and maintaining LOB data types
- Describing LOB data types and how they are used
- Comparing and contrasting LONG and LOB (Large Object) data types
- Migrating data from LONG to LOB

