Large Issues with Large Objects

Michael Rosenblum
Dulcian, Inc.
www.dulcian.com

NYOUG
June 6, 2007

DULCIAN INC.™

Large Objects Needed

- Most modern systems need to store and access large objects:
  - Pictures
  - Movies
  - Documents
  - Sounds

- Standard datatypes are limited (VARCHAR2 - 4,000 characters)
- Need to use Oracle large object datatype (LOB)

Types of Large Objects

- Internal large objects
  - Stored in the database
  - Can be quickly retrieved
  - 3 types
    - BLOBs – used for binary information (multimedia)
    - CLOBs – used for textual information
    - NCLOB – used for info in National Character Set

- External large objects
  - Stored in the file system
  - Only file names are stored in the database
  - Only one type - BFILE
  - Risks and limitations:
    - Performance issues
    - Read-only access

DULCIAN INC.™

Architecture

DULCIAN INC.™

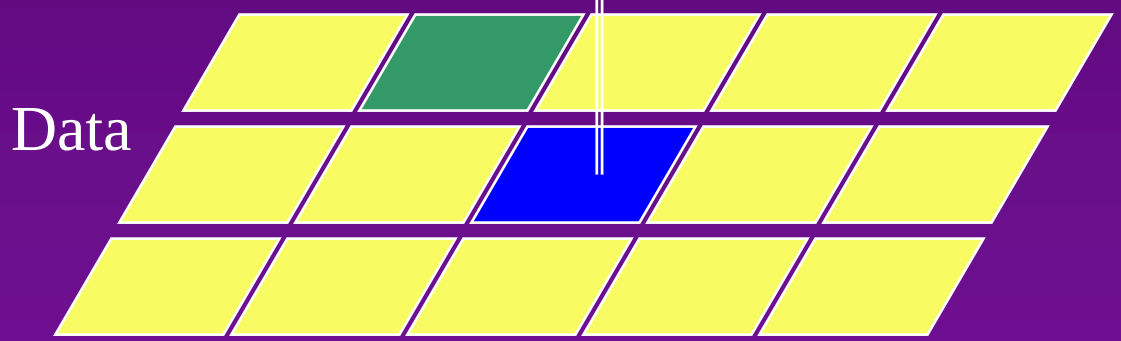
Data Access

- Problem
  - How can you access gigabytes of data?
- Solution
  - Separate LOB data from LOB locator

LOB locator – special logical entity that points to LOB data and allows communication with it.

- Types of LOB operations:
  - ***Copy semantics*** - Data alone is copied from source to destination and a new locator is created for the new LOB.
  - ***Reference semantics*** - Only the locator is copied (without changing the underlying data).

Data Access




```
Declare
     v_cl CLOB;
Begin
     select a_cl
     into   v_cl
     from t_table;

End;
```

LOB Data States

- Variable/column in the row can be in a number of different states:
  - ***Null*** – exists but is not initialized
  - ***Empty*** – exists and has locator that doesn't point to any data
    - Since you can only access LOBs via locators, you must first create them.
    - In some cases, an initial NULL value is needed.
  - ***Populated*** – exists, has locator, and contains real data

DULCIAN INC.

Internal LOB Data Storage

- ***Persistent*** LOBs
  - Represented as a value in the table column
  - Participate in transactions (Commit, Rollback, generate logs)
  - Each LOB has its own storage structure separate form the table in which it is located.
- ***Temporary*** LOBs
  - Created in temporary tablespace and released when no longer needed
  - Created when LOB variable is instantiated
  - When inserted into table, become permanent

Navigation Using Indexes

- Data is stored in ***chunks***:
  - Consist of one or more data blocks up to 32K
  - Each I/O operation works with one chunk.
- Chunks are navigated using ***indexes***
  - Each LOB column is represented by 2 segments
    - One for storing data; one to store the index
    - Each segment has the same storage properties as regular tables.
  - Some restrictions
    - Cannot drop or rebuild LOB indexes
    - Cannot specify different properties for index and data segments

LOB Operations

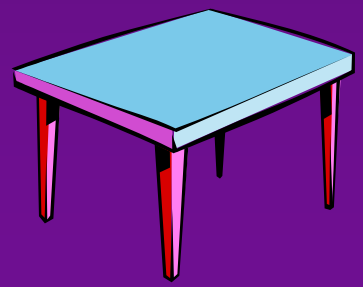
- Require physical I/O
- May have a high number of wait events
- ***Caching options***:
  - NOCACHE (default) – OK for very large LOBs or those with infrequent access
  - CACHE – best option but requires a lot of read/write activity
  - CACHE READS – useful when creating LOB once and reading data from it often.
- Storage "***in row***"
  - Enabled – data less than 4000 characters will look like VARCHAR2(4000)
  - Disabled – everything goes directly to LOB segment

Basic Example

Sample LOB Table

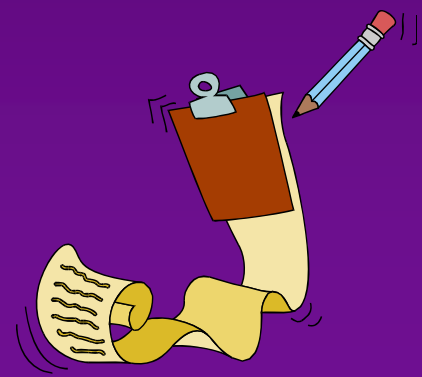
- Using an online shopping catalog as an example, this code uses LOBs to create a table:

```
create table goods_tab
   (item_id      number primary key,
    name_tx      varchar2(256),
    remarks_cl   CLOB DEFAULT empty_clob(),
    manual_cl    CLOB DEFAULT empty_clob(),
    firstpage_bl BLOB DEFAULT empty_blob(),
    mastertxt_bf BFILE)
LOB(remarks_cl) store as remarks_seg(
    tablespace USERS
    enable storage in row
    chunk 8192
    cache)
LOB(manual_cl) store as manual_seg(
    tablespace LOBS_BIG
    disable storage in row
    chunk 32768
    nocache)
LOB(firstpage_bl) store as firstpage_seg(
    tablespace LOBS_BIG
    disable storage in row
    chunk 32768
    cache reads)
```



LOB Code Details (1)

- Each internal LOB:
  - Has its own storage block at the end of the table definition.
  - Has explicit segment names (remarks_seq, manual_seq, firstpage_seq)
- All internal LOBs are initialized to empty values:
  - EMPTY_BLOB( )
  - EMPTY_CLOB( )

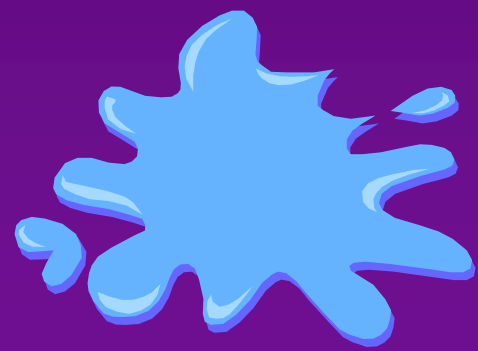


LOB Code Details (2)

- Column ***remarks_cl*** is accessed and modified very often, but the amount of data is not very large:
  - It resides in the same tablespace as the main data.
  - Storage is enabled within the row.
  - The cache option enabled.
- Column ***manual_cl*** is rarely accessed:
  - It resides in an independent tablespace.
  - Large chunk size
  - There is no storage in the row.
  - There are no caching options.
- Difference between ***firstpage_bl*** and ***manual_cl***
  - firstpage_bl is never updated, but often queried.
    - Enable caching on reads
  - Set everything else the same way for both.

Loading BFILE

- Create a directory to access files from the operating system.
- Check the existence of required files (Oracle cannot do it for you.)
- Create a variable of type BLOB using special built-in function BFILENAME that takes the directory and file name and returns a temporary locator.
- Insert a newly created locator to the table and make it permanent.



<<Example 01>>

Loading CLOB and BLOB from files

- Easy way of loading contents into CLOB using the special PL/SQL APIs provided by built-in package DBMS_LOB
- There is no UPDATE in the block, but the value in the table will be changed.
  - Using SELECT…INTO…FOR UPDATE locks the record and returns the locators back to the LOBs.
  - Write data directly to storage using a locator.

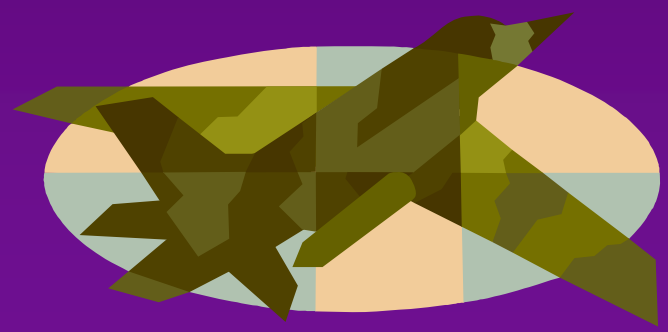
<<Example 02, 03>>

Using BLOBs & CLOBs

- BLOBs
  - Use for any binary information required for the application
  - Unstructured data
  - Mostly read/write with further processing outside of Oracle
- CLOBs –
  - Use for any textual data
  - Semi-structured by definition (character string is already a structure)
  - A lot of extra activity when accessing CLOBs.
  - Oracle modified standard string built-in functions (search for the patterns, get length, get part of the code, etc.) to support CLOBs.
  - Additional functions in DBMS_LOB package

<<Example 04>>

Additional Info



LOB Restrictions – Generic (1)

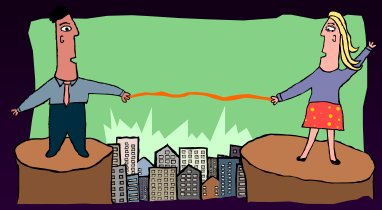
- SQL activity restrictions:
  - Cannot have LOB columns in ORDER BY, GROUP BY clauses or aggregate functions.
  - Cannot have an LOB column in a SELECT DISTINCT statement.
  - Cannot join two tables using LOB columns.
  - Direct binding of string variables is limited to 4000 characters if you are passing a string into the CLOB column.
- DDL restrictions:
  - LOB columns cannot be part of a primary key
  - LOB columns cannot be part of an index (unless it is a domain index or Oracle Text)
  - Cannot specify a LOB column in the trigger clause FOR UPDATE OF.
  - Changing LOBs using locator/DBML_LOB package means no UPDATE trigger fired on the table

<<Example 05>>

LOB Restrictions – Generic (2)

- DBLink restrictions:
  - You can only use CREATE TABLE AS SELECT and INSERT AS SELECT if the remote table contains LOBs.
  - No other activity is permitted.
- Administration restrictions:
  - Only a limited number of BFILEs can be opened at the same time.
    - The maximum number is set up by the initialization parameter SESSION_MAX_OPEN_FILES.
    - The default value is 10, but it can be modified by the DBA.
  - Once a table with an internal LOB is created, only some LOB parameters can be modified.
    - You can change the tablespace, storage properties, caching options, but you cannot modify the chunk size, or storage-in-the-row option.

DULCIAN INC.



String Issues

- Advantages:
  - Overloads of standard built-in functions simplify string activities
  - Explicit conversions of datatypes.
    - Can assign a CLOB column to a VARCHAR2 PL/SQL variable as long as it can hold all of the data from the CLOB.
    - Can initialize a CLOB variable with a VARCHAR2 value.
- Problems:
  - When not to use SQL semantics:
    - More than 100K of data for each CLOB → APIs handle caching much better.
    - Random access to CLOB → APIs utilize LOB indexes much more efficiently.
  - Compare LOBs (>,!=, between) only as part of a PL/SQL routine.
  - Using SQL semantics can cause problems with some built-in functions.
    - INITCAP, SOUNDEX, TRANSLATE, DECODE and some other functions will only process the first 4K (for SQL statements) and 32K (for PL/SQL code) of your data.
    - There will be no notifications, and no errors, but part of the data will be removed.

<<Example 06, 07>>

Transaction Restrictions

- Each locator may or may not contain a transaction ID.
  - If you already started a new transaction (SELECT FOR UPDATE, INSERT/UPDATE/DELETE, PRAGMA autonomous transaction), your locator will contain the transaction ID.
  - If you use SELECT FOR UPDATE of an LOB column, the transaction is started implicitly and your locator will contain the transaction ID.
- You cannot read using the locator when it contains an old transaction ID if the transaction level is set to SERIALIZABLE.
- First write using a locator
  - You need to have a lock on the record that you are trying to update.
  - If the locator did not have transaction ID before update, now it has one.
- Consecutive write using a locator
  - The locator must have the same transaction ID as in current transaction

<<Example 08, 09>>

Transaction Rules

- 1. You can perform read operations using locators as much as you want.
- 2. If you want to write using a locator, you need to have a lock on the record.
- 3. If you want to write using the same locator multiple times, you must do it in the same transaction.

Issues with Advanced Features

- Autonomous transactions:
  - If you locked the record in the parent transaction, you cannot lock it in the child one (otherwise deadlock will occur).
  - If you locked the record in the parent transaction and already did the first update (so the locator contains transaction ID), the update in the subroutine will fail because its transaction ID is different from the one passed with the locator
  - If you locked the record in the child transaction, you have to either commit or rollback the changes to complete it. As a result, the locator will contain the wrong transaction ID.
- Dynamic SQL:
  - All DDL commands fire implicit COMMITs.

<<Example 10, 11, 12, 13>>

Real World Examples

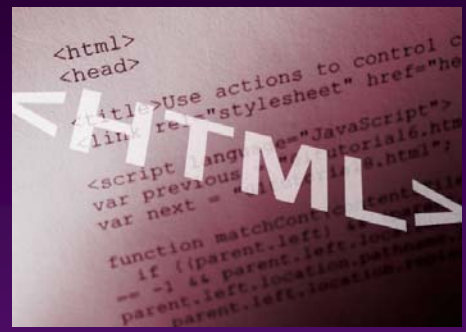
DULCIAN INC.™

HTML on the Fly

- Task:
  - There is a large organizational structure that could be versioned.
  - New structural model should be validated before rolling it over the old one.
- Possible approaches:
  - Populate a temporary table where one row represents one error.
    - **Advantages :** No clean up is needed.
    - **Disadvantages :** Two-step process (populate and query) requiring the same session, which may not be possible in the web environment. Formatting the result must be hard-coded on the client side, so there is no way to change it.
  - Populate (and commit) a permanent table and clean up after the user confirms that he/she saw the report.
    - **Advantages:** Resolves the problem of session-dependency.
    - **Disadvantages:** What if the user's connection was broken? There may be a lot of records that will not be cleaned. Formatting of the result has to be hard-coded on the client side, so there is no way to change it.
  - Create a function that returns an object collection.
    - **Advantages:** One-step process; no clean up is required.
    - **Disadvantages:** Formatting the result has to be hard-coded on the client side, so there is no way to change it.

CLOBs and HTML

- Issues to be resolved:
  - Session-dependency
  - Clean up
  - Formatting



- Solution:
  - Special function that returns CLOB and uses HTML-tags to format the output.
- Major points:
  - A function takes a parameter and returns a CLOB in one round-trip.
  - Temporary CLOBs are released automatically.
  - Full formatting can be done in the CLOB itself.

<<Example 14>>

XML-Based Forms

- Task:
  - Data stored in the relational database needs to be passed to an environment similar to XML (not exactly XML so you cannot use standard Oracle features).
- Solution:
  - Special mapping routine is built to have a CLOB with XML as an output
- Advantages:
  - Free formatting
  - Easy clean up
  - One roundtrip

- Major points:
  - Whenever you need to pass structured data, an XML-based format can be very useful.
  - Oracle also uses CLOB as the storage mechanism for its XMLType datatype.

<<Example 15 (Demo)>>

Emailing from the Database

- Task:
  - Send attachments directly from database.
- Solution
  - Load the file you are planning to attach to the temporary LOB (for performance reasons).
    - You can read data directly using BFILE, but in that case, each operation will cause a direct read (and a large number of wait events with any significant number of users).
  - Attach the data to the email
    - By SMTP protocol standards, you cannot just pour binary data into the body of the email.
    - Data should be encoded into the special BASE64 format (textual representation of binary data).
    - Only send a limited number of bytes at a time.

<<Example 16>>

DULCIAN INC.™

Conclusions

- Large objects can be very useful because most information can now be stored in the database.
- You need a thorough understanding of the core mechanisms, ideas and principles of LOBs or you may do more harm than good.
- Don’t try to use new features in production systems before doing thorough testing.
- Don’t believe everything you read without testing it for yourself.

Contact Information

- Michael Rosenblum – mrosenblum@dulcian.com
- Dulcian website - www.dulcian.com

Available now!
Oracle PL/SQL for Dummies
By Michael Rosenblum &
Dr. Paul Dorsey