IBM Information On Demand 2006

IBM

Exciting News about LOBs in DB2 9 for z/OS

Maryela Weihrauch, IBM Silicon Valley Lab.
1438A, Data Servers - DB2 for z/OS
Thu, Oct 19th, 03:15 PM - 04:30 PM

TAKE BACK CONTROL

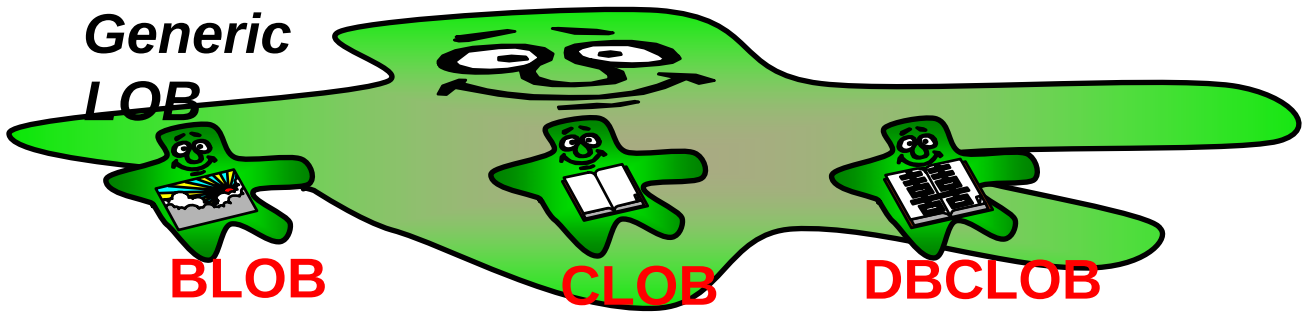
IBM INFORMATION ON DEMAND 2006
October 15 – 20, 2006
Anaheim Convention Center
Anaheim, California

Objectives

- Provides an overview on large objects (LOBs) and how to use them.
- Discusses what is new around LOBs until V8
- Discuss what are the new LOB features in V9 and shares some preliminary performance results.

What are Large Objects (LOBs)

- DB2 z/OS V6 introduced object-relational features
  - User-Defined Functions (UDFs)
  - Triggers
  - Large Objects
    - To store large bit and byte string data with a limit of 2 G
    - There are 3 data types
      - BLOB - Binary Large Object - Useful for Audio, Image data
      - CLOB - Character Large Object (SBCS or Mixed character data)
      - DBCLOB - Double Byte Character Large Object



LOB Overview

Base table space

Base table

Key	ROWID	Column_2	LOB indicator
Key A	prt to LOB 1	user data A	LOB indicator 1
Key B	ptr to LOB 2	user data B	LOB indicator 2

Auxiliary index:
based on ROWID
used to navigate to LOB data

- Rows represent LOBs
- LOBs stored outside base table in auxiliary table
- Base table space may be partitioned
  - If so separate LOB table space for each part

LOB table space

Auxiliary table

ROWID	LOB data
LOB 1 ROWID	LOB data for row user data A
LOB 2 ROWID	LOB data for row user data B

LOB Overview …

```
-- create base table
CREATE TABLE  TB01
        ( FKEY    INTEGER, …
            FROWID  ROWID,
            FCLOB   CLOB(10M),  ….) …

-- create LOB table space
CREATE LOB TABLESPACE   LTS01 … LOG NO;

-- create auxiliary table for LOB column FCLOB
CREATE AUX TABLE  AUXTB01 IN DB
  STORES  TB01
  COLUMN  FCLOB;

-- create index for auxiliary table
CREATE UNIQUE INDEX  AUXIX01
  ON  AUXTB01 …
```

LOB Usage in Applications until V8

- Host Variables
  - May be unmanageable for large LOB values because of storage needs
  - Example
    SELECT FCLOB INTO :buf FROM TB01
    WHERE FKEY= key;
- LOB Locators
  - Refers to a LOB value without accessing the LOB itself
  - Example
    loc SQL TYPE IS CLOB AS LOCATOR;
    SELECT FCLOB INTO :loc FROM TB01
    WHERE FKEY= key;

Widespread LOB usage in DB2 z/OS

- LOB data types have become frequently used data types in business application (e.g. SAP)
  -> performance requirements

- Size of databases is continually growing
  -> utility and management requirements

- Timely and frequent access to LOBs is now an integral part of complex business applications

LOB LOCKS in V8

- DB2 acquires a lock on the LOB while performing:
  - INSERT, UPDATE, DELETE and SELECT operations.
- In V8, a lock acquired on a LOB column is known as a **"LOB Lock".**
- LOB Locks control serialisation of readers and updaters of LOB columns.
- During LOB space allocation, a LOB Lock determines space re-use
  - Determines whether previously de-allocated LOB space can be reallocated

LOB Locks in V9

- Eliminates LOB locks, LRSN and page latching is used instead for consistency checks
  - INSERT, UPDATE, DELETE and SELECT operations continue to hold locks on the base row or page, but no LOB lock is held in either S or X mode
  - Readers with UR attempt now to request a conditional lock on the base row if LOB column(s) are to be selected, in order to serialize with concurrent INSERT or UPDATE
- During LOB space allocation, LOB locks used in the space allocation are also eliminated
  - READ LSN is used to serialize the space reclaim
  - DB2 stores an LSN for each de-allocated page under one lower space map instead of a single LSN in V8

File Reference Variable Support in V9

- A host variable that contains the file name to directly read from or write into the LOB value.
- Advantages:
  - Allow a large LOB or XML value to be inserted from a file or selected into a file rather than a host variable.
  - The application no longer needs to acquire storage to contain the LOB or XML value and avoid related issues.

- facilitate the movement of LOB or XML values from the database server to an application or from an application to a database server without having to go through the working (or dynamic) storage of the application. Applies for local applications.

- DB2 for zOS supports files in the HFS (Hierarchical File System) data set, and also supports BSAM files.

- File Reference Var. are written and read at DASD speed.

Example of File Reference Variable Input Usage

```
EXEC SQL BEGIN DECLARE SECTION
   SQL TYPE IS CLOB_FILE  hv_text_file;
   char hv_patent_title[64];
EXEC SQL END DECLARE SECTION

EXEC SQL BEGIN DECLARE SECTION
struct {
          unsigned long name_length      // file name length
          unsigned long data_length      // data length
          unsigned long file_options     // file options
          char              name[255]    // file name
        } hv_text_file;
   char hv_patent_title[64];
EXEC SQL END DECLARE SECTION

strcpy(hv_text_file.name, "/u/gainer/papers/sigmod.94");
strcpy(hv_patent_title, "Internet-ready Toaster");
hv_text_file.name_length = strlen("/u/gainer/papers/sigmod.94");
hv_text_file.file_options = SQL_FILE_READ;

EXEC SQL INSERT INTO PATENTS(TITLE,TEXT)
VALUES(:hv_patent_title, :hv_text_file);
```

LOB Support in DB2 Utilities

- LOB support introduced in DB2 V6
  - REORG for LOB table spaces
  - CHECK LOB Utility
  - COPY and RECOVER support for LOB table spaces
  - RUNSTATS support for LOB table spaces
  - LOAD support for LOB table spaces
  - etc etc …
- New V7 UNLOAD and COPYTOCOPY Utilities have LOB support
- V7 & V8 maintenance enhancement for:
  - CHECK LOB serviceability improvement including sort enhancement.
  - Cross-loader, LOAD / UNLOAD support for >32K LOBs
    - PTFs available for PK22910 (plus PK24830, PK27566, PK27029, and PK27125)

Improved LOB handling for LOAD & UNLOAD in V9

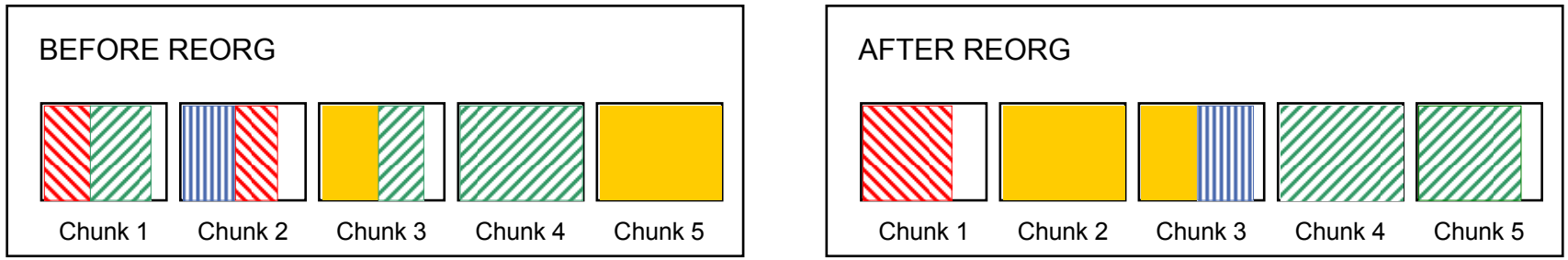
- V9 LOAD and UNLOAD uses File Reference Variables
- LOAD will allow an input field value to contain the name of file containing a LOB column value.
  - The LOB column value is loaded from that file.
- UNLOAD will store the value of a LOB column in a file, and record the name of the file in the unloaded record of the base table.
- UNLOAD Example:|

```
TEMPLATE LOBFRV DSN 'UNLDTEST.&DB..&TS..RESUME'
DSNTYPE(PDS) UNIT(SYSDA)

UNLOAD DATA FROM TABLE DSN8910.EMP_PHOTO_RESUME
(EMPNO CHAR(6),
RESUME VARCHAR(255) CLOBF LOBFRV)
SHRLEVEL CHANGE
```

Dynamically creates a data set UNLDTEST.DB1.TS1.RESUME with a member for each lob value unloaded. And puts that value in the UNLDDN data set.

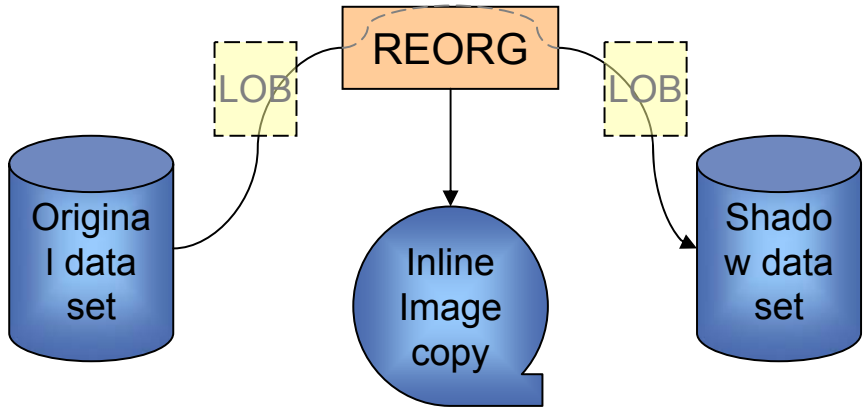
REORG of LOB Table Spaces Until DB2 V8



- Performed in-place by moving individual LOBs within the LOB table space.
  - The aim, to "re-chunk" LOBs to ensure where possible that all pages belonging to an individual LOB are stored in contiguous chunks (sets of 16 contiguous pages) within the table space.
- In the diagram above, this is illustrated without reference to the auxiliary index or to the pointers linking the different parts of a LOB.
  - Each LOB is represented by a different coloured block.
- Free space can not be reclaimed
- No access to LOBs during REORG
- LOG NO is not allowed for REORG of a LOB table space, resulting in additional logging

REORG of LOB table spaces in DB2 V9

- Existing **SHRLEVEL NONE** implementation will continue to work in V9 **(Default).**
- V9 introduces a new **SHRLEVEL REFERENCE** implementation.
  - Works in both CM and NFM.
  - Provides better availability and a more complete reorganisation of LOB data.
  - Allows physical space reclamation from LOB table spaces.

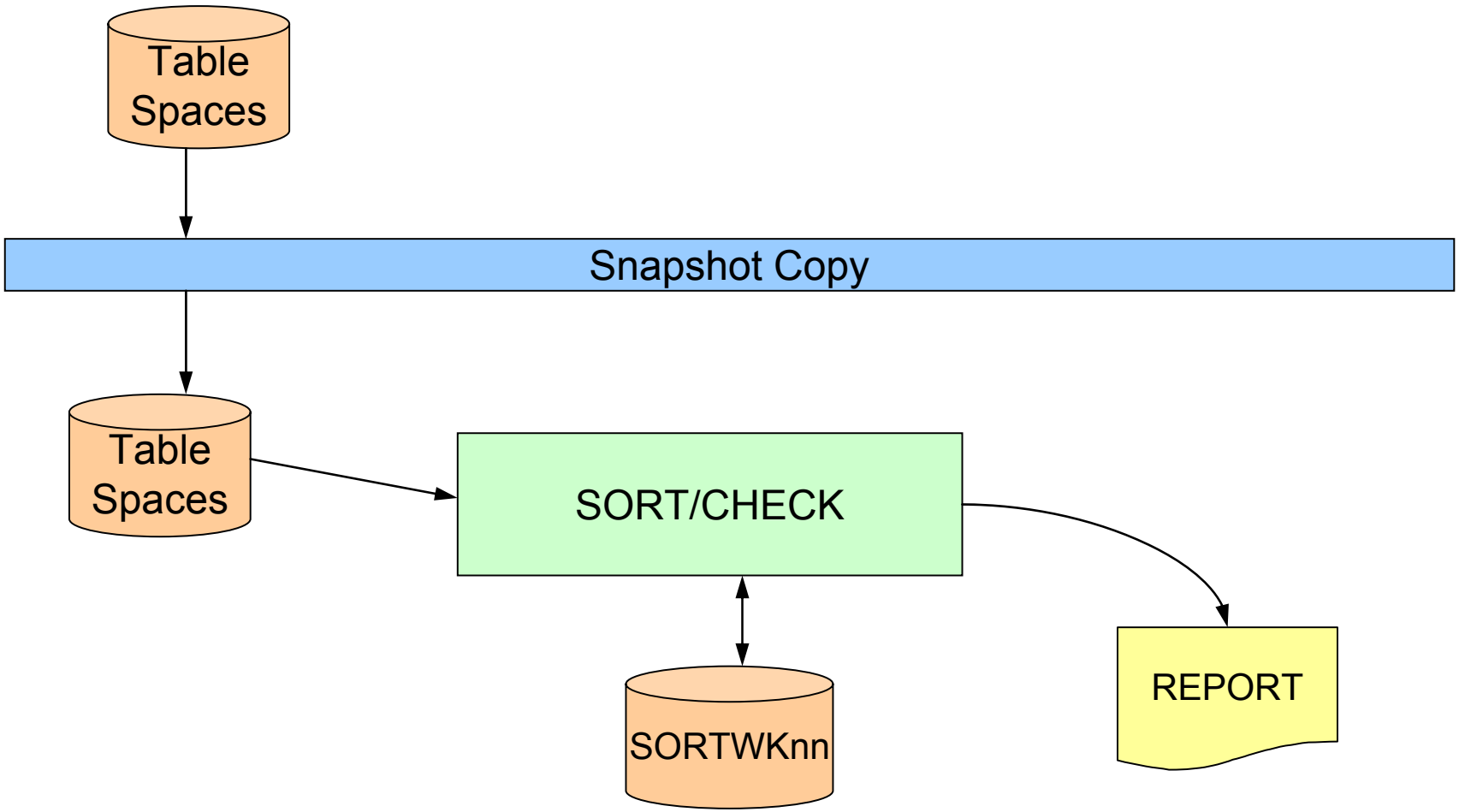


  - Temporarily requires additional DASD space for a shadow data set, but once REORG completes the original data set is deleted.
    - Delete is dependent upon existing rules for DB2-managed vs. user-defined datasets.

Online CHECK LOB in DB2 V9

- Introduces support of SHRLEVEL REFERENCE and SHRLEVEL CHANGE options for CHECK LOB as well as to CHECK DATA
  - SHRLEVEL REFERENCE – allows concurrent read access to the target data but no write access.
  - SHRLEVEL CHANGE – allows concurrent read and write access to the target data.
- SHRLEVEL CHANGE runs against shadow copies of the target objects populated by flash (snapshot) copy.
- At UTILTERM, when the objects are DB2-managed, the shadow data sets are deleted by DB2.
  - For user-managed objects, it is your responsibility to delete the shadow data sets.

Online CHECK LOB (SHRLEVEL CHANGE)



LOB XML Flow Optimization

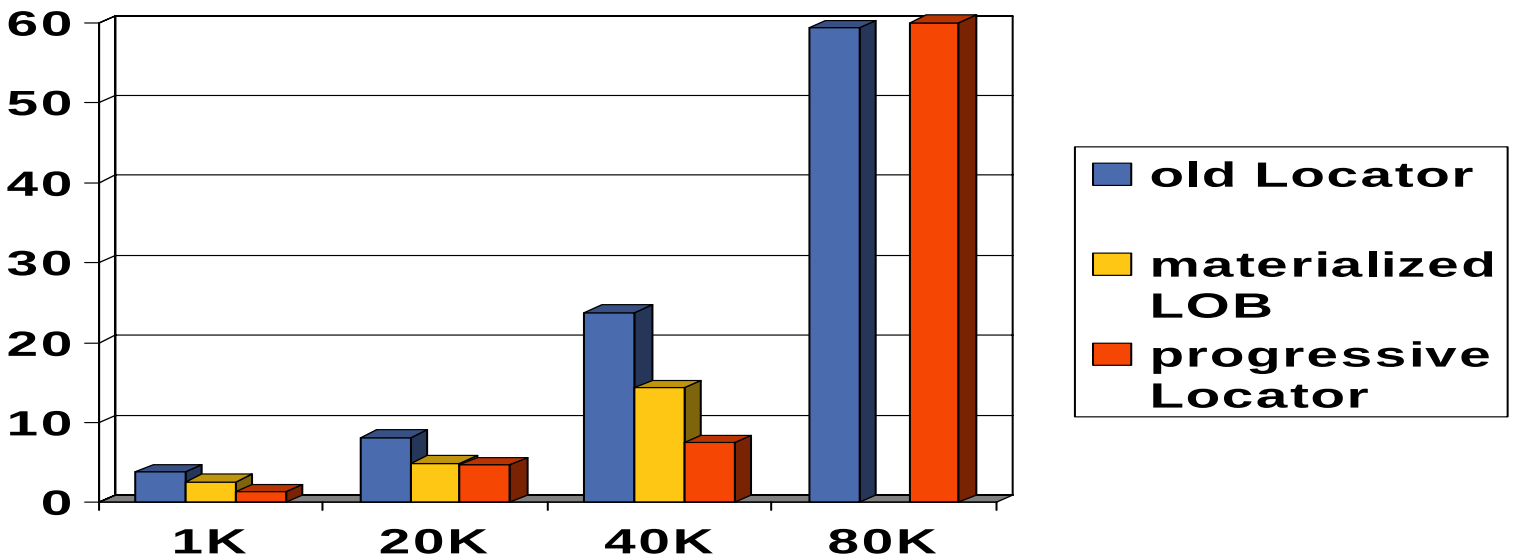
- Need for more efficient retrieval of LOB/XML data where actual data size varies significantly
- Existing support is independent of actual data size via
  - Materialized LOB/XML
    JCC property fullyMaterializeLobData=true (default)
  - Usage of LOB Locator
    JCC property fullyMaterializeLobData=false
- Many applications effectively use locators to retrieve LOB data
  - incurs a separate network flow to get the length of the data to be returned
  - Requester can determine the proper offset and length for SUBSTR operations on the data to avoid any unnecessary blank padding of the value.
- For small LOB data, returning the LOB value directly instead of using a locator would be more efficient.

Progressive Locator in V9

- New support determines the most efficient mode to return LOB/XML based on actual size
  - Usage of progressive Locator
    JCC properties
    progressiveStreaming=ON/OFF and
    streamBufferSize= value (default 1M)
- LOB returned
  - For LOB/XML data size smaller than 12k, the data is inlined similar to varchar data
  - For LOB/XML data size between 12k and streamBufferSize, the data is chained to the query result
  - For LOB/XML data size between streamBufferSize and 2G LOB locator is returned

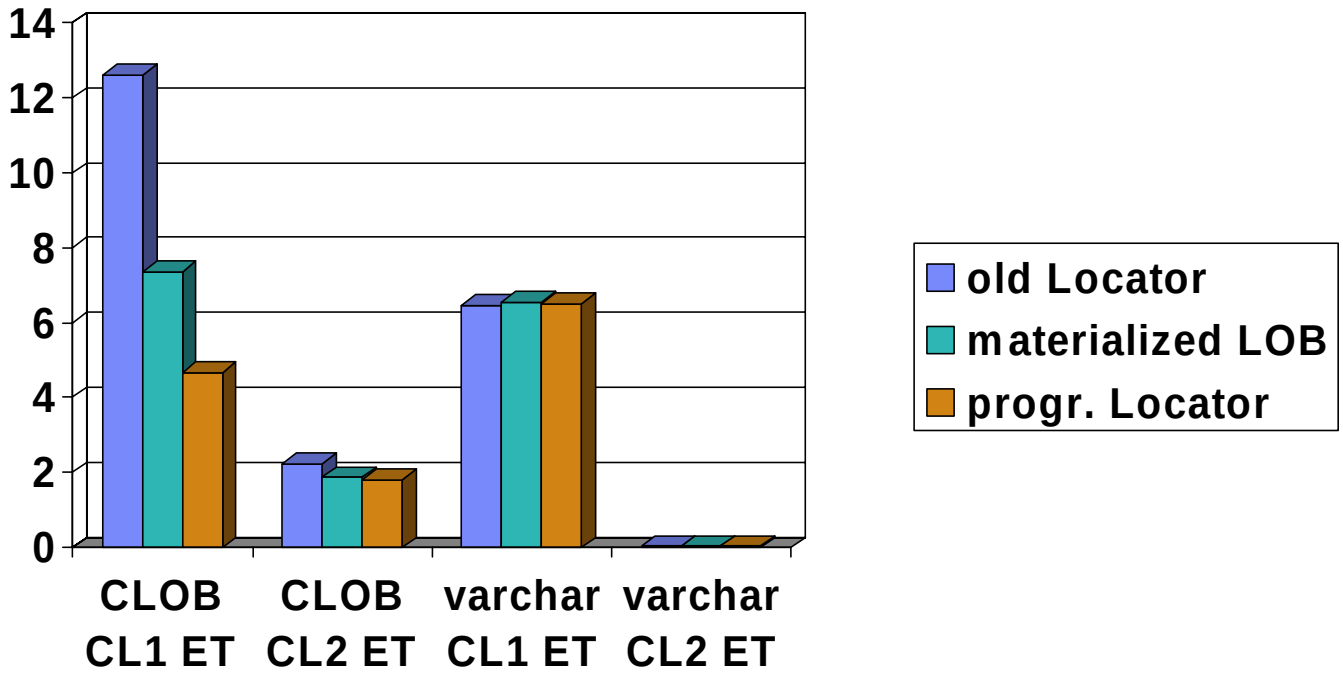
Performance Results

- **Elapsed time to retrieve 1000 CLOB values of varying size streamBufferSize=70000**
- **Progressive Locator processing**
  - 1K and 20K LOB send inlined in query result
  - 40K LOB send chained to query result
  - 80K LOB send via locator



Performance Results …

- Elapsed time to retrieve 1000 20K CLOB versus 1000 20K varchar
- Application response time for large objects (> 12K and <32K) is better than stored in varchar when row buffering is used and multiple rows are retrieved.



FETCH CONTINUE Overview

- Will enable applications to retrieve LOB and XML columns in multiple pieces without using a LOB locator.
- It allows applications to “continue” a FETCH to retrieve the remaining data for LOB and XML columns when truncation occurs.
- The application must ask DB2 to enable this capability on the initial FETCH by using the new **WITH CONTINUE** clause.
- The application can then use a **FETCH CURRENT CONTINUE** operation to retrieve the subsequent pieces of the column.
- The application must manage the buffers and re-assemble the pieces of data.
  - However, the application is *not required* to fetch the entire column before moving to the next row.
- For applications that perform "random access" to parts of a LOB, using functions such as LENGTH, SUBSTR and POSSTR, or when LOB materialization is to be avoided, the use of LOB locators is still recommended.

FETCH CONTINUE Method One

```
EXEC SQL OPEN CURSOR1;
```

Prepare for FETCH:
- Allocate data buffers (32K for each CLOB, XML item)
- Set data pointers and lengths in SQLDA.

```
EXEC SQL FETCH WITH CONTINUE CURSOR1 INTO DESCRIPTOR :SQLDA;
```

if truncation occurred on any LOB or XML column
- loop through each column
  - if column is LOB or XML and was truncated
    - allocate larger buffer area for any truncated columns, move first chunk of data into larger area, reset data pointers, length fields in SQLDA
  - endif
- endloop

```
EXEC SQL FETCH CURRENT CONTINUE CURSOR1 INTO DESCRIPTOR :SQLDA;
```

endif

```
EXEC SQL CLOSE CURSOR1;
```

Reference

- Redbook SG24-6571-00
  "Large Objects with DB2 for z/OS and OS/390"
  http://www.redbooks.ibm.com/abstracts/sg246571.html?Open

- Coming Redbook SG24-7270-00
  "LOBs with DB2 for z/OS: Stronger and Faster"

- General DB2 z/OS resources
  - **DB2 for z/OS and OS/390**
    - **http://www.software.ibm.com/data/db2/zos**
    - **http://www.software.ibm.com/data/net.data**
  - **DB2 Family Performance**
    - **http://www.software.ibm.com/data/db2/performance**
  - **DB2 Solutions Directory Applications + Tool Search**
    - **http://www.software.ibm.com/solutions/isv**
  - **DB2 Magazine**
    - **http://www.db2mag.com**