[**Oracle 대형 데이터형(BLOB, CLOB, LOB, NCLOB, BFILE)과 Java 예제**](http://orderby.tistory.com/entry/Oracle-%EB%8C%80%ED%98%95-%EB%8D%B0%EC%9D%B4%ED%84%B0%ED%98%95BLOB-CLOB-LOB-NCLOB-BFILE%EA%B3%BC-Java-%EC%98%88%EC%A0%9C)

랭킹메이커 2015.10.13 23:17

**1. LOB - Long Object**

- LOB은 TEXT, 그래픽, 이미지, 비디오, 사운드 등 구조화되지 않은 대형 데이터를 저장시 사용한다.

- 일반적으로 테이블에 저장되는 구조화된 데이터들은 크기가 작지만, 멀티미디어 데이터의 크기가 크다.

- 크기가 큰 데이터는 DB에 저장하기 힘들기 때문에 OS상 존재하는 파일을 데이터베이스가 접근하게 된다.

- LONG, LONG RAW 데이터 유형은 예전에 사용던 것이고, 현재는 대부분 LOB 데이터 유형을 사용한다.

- TO\_LOB 함수를 이용하여 LONG 및 LONG RAW 를 LOB 으로 변경할 수 있다.

**(종류)**

- **CLOB**: 문자 대형 객체 (**Character**). Oracle Server는 CLOB과 VARCHAR2 사이에 암시적 변환을 수행한다.

- **BLOB**: 이진 대형 객체 (**Binary**). 이미지, 동영상, MP3 등

- **NCLOB**: 내셔널 문자 대형 객체 (**National**). 오라클에서 정의되는 National Character Set을 따르는 문자.

- **BFILE**: OS에 저장되는 이진 파일의 이름과 위치를 저장. 읽기 전용 모드로만 액세스 가능.

**(데이터베이스 내부, 외부에 따라)**

- 내부 : BLOB, CLOB, NCLOB

         -> Table에 LOB 형식의 컬럼을 생성하고 이곳에 데이터의 실제위치를 가리키는 Locator(위치자) 저장.

- 외부 : BFILE

**특징**

- 하나의 테이블에 여러 개의 LOB 열(column) 가능

- 최고 4GB까지 저장

- SELECT로 위치자 반환

- 순서대로 또는 순서없이 데이터 저장

- 임의적 데이터 액세스

**(LOB 구성)**

- LOB 값 : 저장될 실제 객체를 구성하는 데이터

- LOB 위치자 : 데이터베이스에 저장된 LOB값의 위치에 대한 포인터

- LOB열에는 데이터가 없고 LOB 위치자만 들어있다.

**2. 내부 LOB**

- LOB값은 데이터베이스에 저장된다.

- 사용자 정의 유형 속성, 테이블의 열, 바인드 변수, 호스트 변수, PL/SQL변수, 파라미터 또는 결과

- 동시성 방식, 리두 로그 및 복구 방식, 커밋또는 롤백 트랜잭션 사용 가능

- BLOB 데이터 유형은 Oracle Server 내에서 비트 스트림으로 해석된다.

- CLOB 데이터 유형은 단일 바이트 문자 스트림으로 해석된다.

- NCLOB 데이터 유형은 데이터베이스 national character set 의 바이트 길이에 따라 멀티바이트 문자 스트림으로 해석된다.

**(내부 LOB 관리)**

- LOB 데이터 유형을 포함하는 테이블을 생성한 후 이를 채운다.

- 프로그램에서 LOB 위치자를 선언하고 초기화한다.

- SELECT FOR UPDATE를 사용하여 LOB이 포함된 행을 잠그고 해당 행 값을 LOB 위치자에 넣는다.

- LOB 위치자를 LOB값에 대한 참조로 사용하여 DBMS\_LOB 패키지 프로시저, OCI호출, OLE용 오라클 객체, 오라클 선행 컴파일러 또는 JDBC를 통해 LOB을 조작한다.

- SQL을 통해서도 LOB 관리 가능(일부)

- COMMIT을 통해 변경사항 적용.

**3. CLOB**

---------------------------------------------------------------------------------------

clob\_test.sql

---------------------------------------------------------------------------------------

CREATE TABLE CAR (

        id NUMBER(6) PRIMARY KEY NOT NULL,

        name VARCHAR2(20),

        price NUMBER(10),

        review  **CLOB**);

INSERT INTO CAR VALUES (100, 'SM7', 25000000, '삼성에서 나온 SM7을 타보니 정말 편안했다.');

INSERT INTO CAR VALUES (200, 'PORTER', 13000000, '역시 1톤트럭의 대표자!!');

INSERT INTO CAR VALUES (300, 'VERNA', 9000000, '싸고 좋은 거 같아용~^^\*');

INSERT INTO CAR VALUES (400, 'SANTAFE', 33000000, EMPTY\_CLOB());

- 테이블 생성시 CLOB 타입 컬럼을 선언하고, 데이터 입력.

- 따로 SIZE는 지정하지 않는다.

- EMPTY\_CLOB() 함수는 위치자(Locator)를 아무 것도 없는 데이터로 초기화시킨다.

**1) import 시켜야할 API**

① oracle.sql.BLOB

② oracle.sql.CLOB

③ oracle.jdbc.driver.OracleResultSet

**2) CLOB**

**\*\*\*\* CLOB 쓰기(WRITE/UPDATE) 예제\*\*\*\***

       // UPDATE 또는 INSERT 명령으로 DB 에 공간 확보

     String query = "UPDATE TABLE SET CLOB\_DATA = EMPTY\_CLOB() " ;

     stmt.executeUpdate(query);

     // 그런 다음 다시 요놈을 다시 SELECT

     query = "SELECT CLOB\_DATA  FROM TABLE WHERE ~ " ;

     stmt = dbConn.createStatement();

     rs = stmt.executeQuery(query);

     if(rs.next()) {

          CLOB clob = null;

          Writer writer = null;

          Reader src = null;

          char[] buffer = null;

          int read = 0;

          clob = ((OracleResultSet)rs).getCLOB(1);

          writer = clob.getCharacterOutputStream();

          // str -> DB에 넣을 내용

          src = new CharArrayReader(str.toCharArray());

          buffer = new char[1024];

          read = 0;

          while ( (read = src.read(buffer,0,1024)) != -1) {

               writer.write(buffer, 0, read); // write clob.

          }

          src.close();

          writer.close();

     }

     dbConn.commit();

     dbConn.setAutoCommit(true);

**\*\*\*\* BLOB 읽기(SELECT) 예제\*\*\*\***

      // SELECT

     String query = "SELECT CLOB\_DATA  FROM TABLE WHERE ~ " ;

     stmt = dbConn.createStatement();

     rs = stmt.executeQuery(query);

     if(rs.next()) {

          StringBuffer output = new StringBuffer();

          Reader input = rs.getCharacterStream("CLOB\_DATA");

          char[] buffer = new char[1024];

          int byteRead = 0;

          while((byteRead=input.read(buffer,0,1024))!=-1){

               output.append(buffer,0,byteRead);

          }

          // contents -> CLOB 데이터가 저장될 String

          String contents = output.toString();

     }

     dbConn.commit();

     dbConn.setAutoCommit(true);

**3) BLOB**

**\*\*\*\* BLOB 쓰기(WRITE/UPDATE) 예제\*\*\*\***

      // UPDATE 또는 INSERT 명령으로 DB 에 공간 확보

     String query = "UPDATE TABLE SET BLOB\_DATA = EMPTY\_BLOB() " ;

     stmt.executeUpdate(query);

     // 그런 다음 다시 요놈을 다시 SELECT

     query = "SELECT BLOB\_DATA  FROM TABLE WHERE ~ " ;

     stmt = dbConn.createStatement();

     rs = stmt.executeQuery(query);

     if(rs.next()) {

          BLOB blob = null;

          BufferedOutputStream out = null;

          BufferedInputStream in = null;

          byte[] buf = null;

          int bytesRead= 0;

          blob = ((OracleResultSet)rs).getBLOB(1);

          out = new BufferedOutputStream(blob.getBinaryOutputStream());

          // str -> DB에 넣을 내용

          in = new BufferedInputStream(new StringBufferInputStream(str));

          int nFileSize = (int)str.length();

          buf = new byte[nFileSize];

          while ((bytesRead = in.read(buf)) != -1){

               out.write(buf, 0, bytesRead);

          }

          in.close();

          out.close();

     }

     dbConn.commit();

     dbConn.setAutoCommit(true);

**\*\*\*\* CLOB SELECT 예제\*\*\*\***

      // SELECT

     String query = "SELECT CLOB\_DATA  FROM TABLE WHERE ~ " ;

     stmt = dbConn.createStatement();

     rs = stmt.executeQuery(query);

     if(rs.next()) {

          BLOB blob = ((OracleResultSet)rs).getBLOB(1);

          BufferedInputStream in = new BufferedInputStream(blob.getBinaryStream());

          int nFileSize = (int)blob.length();

          byte[] buf = new byte [nFileSize];

          int nReadSize = in.read(buf, 0, nFileSize);

          in.close();

           // contents -> BLOB 데이터가 저장될 String

          String contents = new String(buf);

     }

     dbConn.commit();

     dbConn.setAutoCommit(true);

----------------------------------------------------------------------------

**(CLOB 사용법)**

**\*\*\*\* CLOB INSERT 예제\*\*\*\***

public void insertData(String key, String data){

    Connection con = null;

    PreparedStatement pstmt = null;

    try{

        con = ds.getConnection();

        pstmt = con.prepareStatement("insert into lobTbl values(?,?)");

        pstmt.setString(1, key);

        // 요기가 CLOB insert 하는 부분입니다. LONGVARCHAR 와 동일

        StringReader sr = new StringReader(data);

        pstmt.setCharacterStream(2, sr, data.length());

        pstmt.executeUpdate();

    }catch(SQLException se){

        se.printStackTrace();

    }finally{

        if(pstmt != null)

            try{pstmt.close();}catch(SQLException cs){cs.printStackTrace();}

        if(con != null)

            try{con.close();}catch(SQLException cs){cs.printStackTrace();}

    }

}

**\*\*\*\* CLOB SELECT 예제\*\*\*\***

while(rs.next()){

    Reader instream = rs.getCharacterStream("data");

    char[] buffer = new char[1024];  // create temporary buffer for read

    int length = 0;   // length of characters read

    // fetch data

    while ((length = instream.read(buffer)) != -1)  {

        for (int i=0; i<length; i++){

            sb.append(buffer[i]);

        }

    }

    instream.close();// Close input stream

}

일반적인 Oracle 에서 CLOB 을 가져올 때는,

  CLOB clob = ((OracleResultSet)rs).getCLOB("data");

라는 형식으로 가져올 수 있다. 그러나 Weblogic6.1 의 테스트 결과 rs 가

weblogic.jdbc.rmi.SerialResultSet 이라는 객체형식으로 반환을 합니다. 그래서 위와같이

casting 하면 ClassCastException 이 발생합니다.

  CLOB clob = ((SerialResultSet)rs).getCLOB("data");

하면 casting 은 문제가 없는데 CLOB 을 지원하지 않는다고 나옵니다. 그래서 위와같이

그냥 getCharacterStream 을 이용하면 됩니다.

조금이라도 도움이 되면 좋겠군요. 참고로 WebLogic 이 아닌 일반적인 Application 에서

Oracle 연결방식입니다.

while(rs.next()){

    System.out.println("key : " + rs.getString(1));

    CLOB clob = ((OracleResultSet)rs).getCLOB(2);

    Reader instream = clob.getCharacterStream();  // get character stream to retrieve clob data

    char[] buffer = new char[1024];  // create temporary buffer for read

    int length = 0;  // length of characters read

    // fetch data

    while ((length = instream.read(buffer)) != -1)  {

        for (int i=0; i<length; i++)

            System.out.print(buffer[i]);

    }

    instream.close();  // Close input stream

}