

Tester med DB2 Universal JDBC Driver V8 på z/OS med fokus på strömning av LOBar

Mats Börjvall / Rolf Gustafsson

Handelsbanken

Bakgrund

- Att med Java och DB2 testa hur LOBar strömmas genom applikationen vid DB2 åtkomst. Full materialisering av LOB i minnet är ej önskvärt.
- Testmiljön:
 - Websphere Application Server 6.01 z/OS (USS)
 - Connection Pooling mha Websphere J2EE Datasource
 - DB2 v8 z/OS full function mode. Körs på samma LPAR
 - DB2 Universal Driver type 2 (RRSAF), property `fullyMaterializeLobData=false` (LOB locator support)
- Databasåtkomst via dynamisk SQL med position markers (Prepared Statement) direkt i Javakoden.
FullyMaterializeLobData har ingen effekt för Stored Procedures. LOB Parametrar hanteras alltid som true, medan resultatset hanteras alltid som false (min egen tolkning av manualen).

Strömning?

- I Websphere fungerar detta bra idag för inkommande filströmmar som sparas ned på fil. Dock inte ännu inte bra för utgående filströmmar.

Man kan själv styra storleken på buffrarna. Normalt 16 – 32 K.

SQL DDL

```
CREATE TABLE GBIA01FT.BLOBTEST (  
    IDNAMN CHAR(10) PRIMARY KEY NOT NULL,  
    IDLOB ROWID GENERATED ALWAYS NOT NULL,  
    FILE BLOB(10000000) NOT NULL  
    ) IN GBIA01FT.GBIA01FA;
```

```
CREATE UNIQUE INDEX GBIA01FT.BLOBTESTI1  
    ON GBIA01FT.BLOBTEST  
    (IDNAMN ASC)  
    USING STOGROUP DBFTDATA;
```

```
CREATE AUX TABLE GBIA01FT.FILE01 IN GBIA01FT.GBIA01FB  
    STORES GBIA01FT.BLOBTEST  
    COLUMN FILE;
```

```
CREATE INDEX GBIA01FT.FILE01I1 ON GBIA01FT.FILE01  
    USING STOGROUP DBFTDATA;
```

INSERT BLOB Javakod

```
Connection connection = datasource.getConnection();
PreparedStatement ps = connection.prepareStatement("INSERT
INTO GBIA01FT.BLOBTEST (IDNAMN, FILE)
VALUES (?,?);");
ps.setString(1, name);
ps.setBinaryStream(2, bais, baisSize);
ps.execute();
```

SELECT BLOB Javakod

```
Connection connection = datasource.getConnection();
PreparedStatement ps = connection.prepareStatement("SELECT
FILE FROM
GBIA01FT.BLOBTEST WHERE IDNAMN=?");
ps.setString(1, name);
ResultSet rs = ps.executeQuery();
InputStream is = rs.getBinaryStream("FILE");
while ( is.read() != -1);
```

Testresultat: INSERT BLOB

- Buffertarean är alltid större än BLOB storleken
 - En 1 000 000 bytes BLOB ger ca. 1MB buffer
 - En 1 MB BLOB ger ca. 3 MB buffer
 - En 10 MB BLOB ger ca.17MB buffer
- Bufferten skapas per Prepared Statement och Connection.
- Livslängden för bufferten och dess PS är lika lång som den connection som skapat PS'et.

Testresultat: SELECT BLOB

- Liknar Webspheres beteende av utdataströmmar.
 - En 1 MB BLOB ger upphov till att $2 * 0,5$ MB objekt allokeras
 - En 2 MB BLOB ger upphov till att $4 * 0,5$ MB objekt allokeras
 - En 10 MB BLOB ger upphov till att $20 * 0,5$ MB objekt allokeras
- Storleken på 0,5 MB objekt kan inte påverkas.
- Objekten allokeras för varje gång en läsning av ett BLOB objekt görs. Ingen återanvändning.

Avslutningvis

- Strömmar inte, fullt minne allokeras. Vad skiljer egentligen mot `fullyMaterializeLobData=true`?
- DB2 z/OS V9:
 - "Progressive locator" eller "fetch continue" för LOB data, som ska förbättra bufferhanteringen och prestanda.
 - File reference variable (Finns redan i DB2 LUW). Verkar vara det vi behöver. Att mellanlagra LOB som fil. Hur kan DB2 nå filen som ligger i USS-filsystemet?
- Kontakt mailledes med Chunyang Xia, IBM Santa Teresa.