Troubleshooting CallableStatement Not Returning All Rows

When using a CallableStatement to execute a stored procedure and retrieve a large result set (e.g., 95,000 rows), several common issues can prevent all rows from being returned. Below are the most likely causes and solutions based on the provided context.

**1. Retrieving the Result Set Properly**

After executing the stored procedure with CallableStatement.execute(), you must retrieve the result set using getResultSet() and iterate through it with next(). If your procedure returns multiple result sets or update counts (such as from INSERT, UPDATE statements before the SELECT), you may need to loop through results using getMoreResults() until you reach the actual result set[2](https://www.ibm.com/docs/en/db2-for-zos/13?topic=rmrsfspija-retrieving-known-number-result-sets-from-stored-procedure-in-jdbc-application)[5](https://github.com/vert-x3/vertx-jdbc-client/issues/96).

**Sample Approach:**

java

CallableStatement cstmt = connection.prepareCall("{call your\_procedure(?)}");

cstmt.execute();

ResultSet rs = **null**;

*// Loop until a ResultSet is found*

**while** (true) {

rs = cstmt.getResultSet();

**if** (rs != **null**) **break**;

**if** (!cstmt.getMoreResults() && cstmt.getUpdateCount() == -1) **break**;

}

*// Now iterate through all rows*

**while** (rs != **null** && rs.next()) {

*// Process each row*

}

This ensures you are not missing the result set due to prior update counts or intermediate results[5](https://github.com/vert-x3/vertx-jdbc-client/issues/96).

**2. Stored Procedure Output vs. Result Set**

If your procedure uses OUT parameters to return data, those are not intended for large data sets. OUT parameters are for scalar values, not for returning thousands of rows[3](https://docs.oracle.com/javase/1.5.0/docs/guide/jdbc/getstart/callablestatement.html)[4](https://coderanch.com/t/663717/databases/retrieve-data-Stored-Procedure-Callable). To return large data sets, your procedure must use a SELECT statement, which will be returned as a ResultSet.

**3. Driver and Database Settings**

* **JDBC Fetch Size:** For very large result sets, set the fetch size on your Statement or ResultSet to optimize memory usage and performance:

java

rs.setFetchSize(1000); *// Example value*

* **Database/Network Timeouts:** Large result sets may trigger timeouts. Check your database and JDBC driver timeout settings.

**4. SQL Server/SET NOCOUNT ON**

If you are using SQL Server, intermediate update counts can interfere with result set retrieval. Use SET NOCOUNT ON at the start of your stored procedure to suppress these counts, ensuring the result set is returned as expected[5](https://github.com/vert-x3/vertx-jdbc-client/issues/96).

**5. Procedure Returns Too Many Rows for OUT Parameter**

If your stored procedure is designed to return a single value via an OUT parameter, but the underlying query returns multiple rows, this will cause errors or incomplete results[4](https://coderanch.com/t/663717/databases/retrieve-data-Stored-Procedure-Callable). Ensure your procedure is structured to return a result set for large data, not via OUT parameters.

Summary Table

| **Issue** | **Symptom** | **Solution** |
| --- | --- | --- |
| Not using getMoreResults | Only partial or no result set returned | Loop with getMoreResults until ResultSet is found[2](https://www.ibm.com/docs/en/db2-for-zos/13?topic=rmrsfspija-retrieving-known-number-result-sets-from-stored-procedure-in-jdbc-application)[5](https://github.com/vert-x3/vertx-jdbc-client/issues/96) |
| OUT parameter for large data | Only one value or error is returned | Use SELECT to return data as ResultSet[3](https://docs.oracle.com/javase/1.5.0/docs/guide/jdbc/getstart/callablestatement.html)[4](https://coderanch.com/t/663717/databases/retrieve-data-Stored-Procedure-Callable) |
| Update counts block results | ResultSet is null or not found | Use SET NOCOUNT ON in procedure[5](https://github.com/vert-x3/vertx-jdbc-client/issues/96) |
| Fetch size too small | Slow or incomplete retrieval | Set appropriate fetch size |
| Timeouts/network issues | Retrieval hangs or fails | Adjust timeout settings |

Recommendations

* Ensure your stored procedure ends with a SELECT statement to return the rows.
* After executing the callable statement, loop with getMoreResults() until you find the result set.
* For SQL Server, add SET NOCOUNT ON at the start of your procedure.
* Avoid using OUT parameters for large data sets; use them only for single values.
* Set fetch size and check for any timeout or memory issues.

By following these steps, you should be able to retrieve all 95,000 rows from your stored procedure using a CallableStatement[2](https://www.ibm.com/docs/en/db2-for-zos/13?topic=rmrsfspija-retrieving-known-number-result-sets-from-stored-procedure-in-jdbc-application)[3](https://docs.oracle.com/javase/1.5.0/docs/guide/jdbc/getstart/callablestatement.html)[5](https://github.com/vert-x3/vertx-jdbc-client/issues/96).