Altibase 7.1.0.8.1 Patch Notes

Table of Contents

New Features

BUG-49963 Added aku (Altibase Kubernetes Utility).

module

`rp`

Category

`Usability`

Reproducibility

`Always`

Description

The Altibase Kubernetes Utility (aku) is a utility that helps you perform tasks such as synchronizing Altibase's data or initializing synchronization information based on pod creation and shutdown when scaling from Kubernetes' Statefulset.

Fixed Bugs

BUG-49910 Fixes a phenomenon in which records are inserted even if the INSERT statement execution fails when binding the bind parameter of the INSERT statement to a LOB data type.

module

`qp-dml-execute`

Category

`Functional Error`

Reproducibility

`Always`

Description

Fixes a phenomenon where, when binding the bind parameter of an INSERT statement to a LOB data type, an error occurs saying that the INSERT execution failed, but the record is actually inserted.

This bug occurs when you bind to a LOB data type where the SQL data type of the bind parameter is different from the data type of the actual column. In Altibase Server 7.1.0.8.1 or later, where this bug is reflected, SQL Actual Results are different when performing the same action corresponding to the bug condition.

How to reproduce this bug

- **Reproduction conditions**

  ```
  CREATE TABLE TEST (C1 CHAR(10), C2 CHAR(10));
  ```

  ```
  // 예제 코드 demo_ex2.cpp 일부
      /* prepares an SQL string for execution */
      rc = SQLPrepare(stmt, (SQLCHAR *)"INSERT INTO TEST VALUES(?, ?)", SQL_NTS);
      if (!SQL_SUCCEEDED(rc))
      {
          PRINT_DIAGNOSTIC(SQL_HANDLE_STMT, stmt, "SQLPrepare");
          goto EXIT_STMT;
      }
      /* binds a buffer to a parameter marker in an SQL statement */
      rc = SQLBindParameter(stmt,
                            1,                /* Parameter number, starting at 1 */
                            SQL_PARAM_INPUT,  /* in, out, inout */
                            SQL_C_CHAR,       /* C data type of the parameter */
  ```

```
                          SQL_CHAR,         /* SQL data type of the parameter
: char(8)*/
                          10,               /* size of the column or
expression, precision */
                          0,                /* The decimal digits, scale */
                          id,               /* A pointer to a buffer for the
parameter¡?s data */
                          sizeof(id),       /* Length of the
ParameterValuePtr buffer in bytes */
                          &id_ind);         /* indicator */
    if (!SQL_SUCCEEDED(rc))
    {
        PRINT_DIAGNOSTIC(SQL_HANDLE_STMT, stmt, "SQLBindParameter");
        goto EXIT_STMT;
    }
    /* binds a buffer to a parameter marker in an SQL statement */
    rc = SQLBindParameter(stmt,
                          2,                /* Parameter number, starting at
1 */
                          SQL_PARAM_INPUT, /* in, out, inout */
                          SQL_C_CHAR,       /* C data type of the parameter
*/
                          SQL_CLOB,         /* SQL data type of the parameter
: char(8)*/
                          10,               /* size of the column or
expression, precision */
                          0,                /* The decimal digits, scale */
                          name,             /* A pointer to a buffer for the
parameter¡?s data */
                          sizeof(name),     /* Length of the
ParameterValuePtr buffer in bytes */
                          &name_ind);       /* indicator */
    if (!SQL_SUCCEEDED(rc))
    {
        PRINT_DIAGNOSTIC(SQL_HANDLE_STMT, stmt, "SQLBindParameter");
        goto EXIT_STMT;
    }
    /* executes a prepared statement */
    sprintf(id, "10000000");
    sprintf(name, "name1");
    id_ind         = SQL_NTS;        /* id   => null terminated string */
    name_ind       = SQL_NTS;            /* name => length=5 */
    rc = SQLExecute(stmt);
    if (!SQL_SUCCEEDED(rc))
    {
        PRINT_DIAGNOSTIC(SQL_HANDLE_STMT, stmt, "SQLExecute");
    }
    execute_select(dbc);
```

- **Actual Results**

```
$ demo_ex2
Error : 187 : SQLExecute
  Diagnostic Record 1
```

```
    SQLSTATE     : HY000
    Message text : LobLocator cannot span the transaction 0.
    Message len  : 41
    Native error : 0x110C4
  Diagnostic Record 2
    SQLSTATE     : HY000
    Message text : LobLocator cannot span the transaction 0.
    Message len  : 41
    Native error : 0x110C4
  Diagnostic Record 3
    SQLSTATE     : HY000
    Message text : LobLocator cannot span the transaction 0.
    Message len  : 41
    Native error : 0x110C4
I  D: 10000000
NAME: NULL
```

- **Expected Results**

```
$ demo_ex2
Error : 187 : SQLExecute
  Diagnostic Record 1
    SQLSTATE     : HY000
    Message text : LobLocator cannot span the transaction 0.
    Message len  : 41
    Native error : 0x110C4
NO DATA
```

Workaround

`none`

Changes

- Performance view
- Property
- Compile Option
- Error Code

BUG-49911 [SQLException: Invalid column name] error occurs when returning IS_AUTOINCREMENT, IS_GENERATEDCOLUMN column values from the DatabaseMetaData.getColumns method.

module

`mm-jdbc`

Category

`Functionality`

Reproducibility

`Frequence`

Description

Fix a phenomenon in which [SQLException: Invalid column name] error occurs when returning IS_AUTOINCREMENT, IS_GENERATEDCOLUMN column values of the DatabaseMetaData.getColumns method.

This bug occurs when using the JDBC driver that corresponds to the version below..

- Altibase 7.1 JDBC driver with partial support for JDBC 4.2 API(Altibase42.jar)

Additionally, this bug has been modified so that the return order of the SPECIFIC_NAME and PROCEDURE_TYPE columns in the return result of the getProcedures() method is returned in the order defined in the JDBC 4.2 API specification. Actual Results may vary depending on the application code. The return order of the SPECIFIC_NAME and PROCEDURE_TYPE columns before and after applying this bug is as follows.

before bug fix	after bug fix
rs.getString(8) ==> SPECIFIC_NAME rs.getString(9) ==> PROCEDURE_TYPE	rs.getString(8) => PROCEDURE_TYPE rs.getString(9) => SPECIFIC_NAME

Altibase JDBC driver needs to be patched to apply this bug.

How to reproduce this bug

- **Reproduction conditions**
- **Actual Results**
- **Expected Results**

Workaround

`none`

Changes

- Performance view
- Property
- Compile Option
- Error Code

BUG-49926 Change the maximum value of the MEMORY_ALLOCATOR_TYPE property.

module

`id`

Category

`Fatal`

Reproducibility

`Always`

Description

Change the maximum value of the MEMORY_ALLOCATOR_TYPE property from 1 to 0. If MEMORY_ALLOCATOR_TYPE=1 is added to the Altibase server property file (altibase.properties) in Altibase server 7.1.0.8.1 or later to which this bug is applied, [Property [MEMORY_ALLOCATOR_TYPE] 1 Overflowed the Value Range.(0~0)] error occurs.

How to reproduce this bug

- **Reproduction conditions**
- **Actual Results**
- **Expected Results**

Workaround

`none`

Changes

- Performance view
- Property
- Compile Option
- Error Code

BUG-49939 [ERR-31001: SQL syntax error] error occurs when the GROUP BY GROUPING SETS clause and ORDER BY NULLS FIRST clause or ORDER BY NULLS LAST clause are used together.

module

`qp-select`

Category

`Functional Error`

Reproducibility

`Always`

Description

Fixes a phenomenon in which [ERR-31001: SQL syntax error] error occurs when the GROUP BY GROUPING SETS clause and the ORDER BY NULLS FIRST clause or ORDER BY NULLS LAST clause are used together.

How to reproduce this bug

- **Reproduction conditions**

```
DROP TABLE BUG-49939;

CREATE TABLE BUG_49939 ( C1 VARCHAR(10), C2 VARCHAR(10), C3 VARCHAR(10));

INSERT INTO BUG_49939 VALUES(1,1,1);
INSERT INTO BUG_49939 VALUES(1,2,1);
INSERT INTO BUG_49939 VALUES(2,2,2);
INSERT INTO BUG_49939 VALUES(1,3,2);
INSERT INTO BUG_49939 VALUES(2,1,1);
INSERT INTO BUG_49939 VALUES(2,3,2);

SELECT A1.C1, A1.C2, A1.C3
  FROM BUG_49939 A1
 GROUP BY GROUPING SETS ((A1.C1, A1.C2, A1.C3), ())
 ORDER BY A1.C1 NULLS FIRST, A1.C3 NULLS FIRST;
```

- **Actual Results**

```
[ERR-31001 : SQL syntax error

line 4: missing or invalid syntax
 ORDER BY A1.C1 NULLS FIRST, A1.C3 NULLS FIRST
               ^   ^

]
```

- **Expected Results**

```
C1                    C2                    C3
----------------------------------------------------------------------


1                     1                     1
1                     2                     1
1                     3                     2
2                     1                     1
2                     2                     2
2                     3                     2
7 rows selected.
------------------------------------------------------------
```

Workaround

You can avoid the bug by transforming your query as below.

```
SELECT A1.C1, A1.C2, A1.C3
 FROM BUG_49939 A1
GROUP BY A1.C1, A1.C2, A1.C3
UNION ALL
SELECT NULL C1, NULL C2, NULL C3
 FROM BUG_49939 A1
GROUP BY NULL
ORDER BY C1 NULLS FIRST, C3 NULLS FIRST;
```

Changes

- Performance view
- Property
- Compile Option
- Error Code

BUG-49940 Adds a property that determines the FIXED/VARIABLE option of a column when ALTER TABLE ~ ADD COLUMN is executed.

module

`qp-ddl-dcl-execute`

Category

`Functional Error`

Reproducibility

`Always`

Description

Adds a property that determines the FIXED/VARIABLE option of a column when ALTER TABLE ~ ADD COLUMN is executed.

This bug only affects memory tables.

You need to change the private property to apply this bug. If necessary, please contact the Altibase Technical Support Center.

How to reproduce this bug

- **Reproduction conditions**
- **Actual Results**
- **Expected Results**

Workaround

`none`

Changes

- Performance view
- Property
- Compile Option
- Error Code

BUG-49960 If you get the column name in Korean with getColumnName(), the Korean is broken and an SQLException: Invalid column name error occurs.

module

`mm-jdbc`

Category

`Functional Error`

Reproducibility

`Always`

Description

Fixes a problem where Korean characters are broken when a column name in Korean is retrieved from JDBC when the character sets of the Altibase server and client are different. This bug affects the use of the following methods of the ResultSetMetaData interface.

- getColumnLabel()
- getColumnName()
- getSchemaName()
- getTableName()

Altibase JDBC driver needs to be patched to apply this bug.

How to reproduce this bug

- **Reproduction conditions**

  ```
  $ export LANG=ko_KR.EUC-KR
  ```

  ```
  CREATE TABLE T1 ("컬럼1" INT, "컬럼2" VARCHAR(10));
  INSERT INTO T1 VALUES (1, 'AAAAAAA');
  ```

  ```
  ### Sample Code Part of CharacterSetTest.java
  [source encoding = utf8]
  Connection sCon = getAltiConnection();
  Statement sStmt = sCon.createStatement();
  ResultSet sRs = sStmt.executeQuery("SELECT * FROM t1");
  ```

```
ResultSetMetaData sMeta = sRs.getMetaData();
if (sRs.next())
{
    String sCol1Name = sMeta.getColumnName(1);
    System.out.println("Col1 Name===>" + sCol1Name);
    int sCol1 = sRs.getInt("컬럼1");
    System.out.println("sCol1===>" + sCol1);
    String sCol2 = sRs.getString("컬럼2");
    System.out.println("sCol2===>" + sCol2);
}
```

```
$ javac -encoding utf-8 CharacterSetTest.java
$ java CharacterSetTest
```

- **Actual Results**

```
$ java CharacterSetTest
Col1 Name===>占시뤄옙1
Exception in thread "main" java.sql.SQLException: Invalid column name: 而ш??1
        at
Altibase.jdbc.driver.ex.Error.createSQLExceptionInternal(Error.java:197)
        at
Altibase.jdbc.driver.ex.Error.throwSQLExceptionInternal(Error.java:190)
        at Altibase.jdbc.driver.ex.Error.throwSQLException(Error.java:130)
        at
Altibase.jdbc.driver.AltibaseResultSet.findColumn(AltibaseResultSet.java:398
)
        at
Altibase.jdbc.driver.AltibaseResultSet.getInt(AltibaseResultSet.java:770)
        at CharacterSetTest.doTest(CharacterSetTest.java:23)
        at CharacterSetTest.main(CharacterSetTest.java:10)
```

- **Expected Results**

```
$ java CharacterSetTest
Col1 Name===>컬럼1
sCol1===>1
sCol2===>aaaaaaa
```

Workaround

You can work around the bug by using the -Dfile.encoding=euc-kr option.

Changes

- Performance view
- Property
- Compile Option
- Error Code

Changes

Version Info

altibase version	database binary version	meta version	cm protocol version	replication protocol version
7.1.0.8.1	6.5.1	8.10.1	7.1.7	7.4.7

> You can check the module version change history in [Version_Histories]().

Compatibility

Database binary version

The database binary version has not changed.

> The database binary version indicates the compatibility of database image files and log files. If this version needs to be patched to a different version, the database must be reorganized.

Meta Version

The meta version has not changed.

> If you want to roll back the patch after patching to a version with a changed meta version, see [Meta Downgrade]().

CM protocol Version

The cm protocol version has not changed.

Replication protocol Version

The replication protocol version has not changed.

Altibase Server Properties

Added Properties

Changed Properties

Deleted Properties

Performance Views

Added Performance Views

Changed Performance Views

Deleted Performance Views