

# Altibase 7.1.0.8.1 Patch Notes

---

# Table of Contents

---

- [New Features](#)
  - [BUG-49963 aku\(Altibase Kubernetes Utility\)가 추가되었습니다.](#)
- [Fixed Bugs](#)
  - [BUG-49910 INSERT문의 바인드 파라미터를 LOB 데이터 타입으로 바인드할 때 INSERT문 실행이 실패했음에도 레코드가 삽입되는 현상을 수정합니다.](#)
  - [BUG-49911 DatabaseMetaData.getColumns 메소드의 IS AUTOINCREMENT, IS GENERATEDCOLUMN 컬럼값 반환 시 SQLException: Invalid column name 에러가 발생합니다.](#)
  - [BUG-49926 MEMORY ALLOCATOR TYPE 프로퍼티의 최대값을 변경합니다.](#)
  - [BUG-49939 GROUP BY GROUPING SETS 절과 ORDER BY NULLS FIRST 절 또는 ORDER BY NULLS LAST 절을 같이 사용할 때 ERR-31001 : SQL syntax error 에러가 발생합니다.](#)
  - [BUG-49940 ALTER TABLE ~ ADD COLUMN 수행 시 컬럼의 FIXED/VARIABLE 옵션을 결정하는 프로퍼티를 추가합니다.](#)
  - [BUG-49946 ado.net에서 PSM 수행 시 DbDataReader.NextResult\(\)에서 잘못된 결과를 반환합니다.](#)
  - [BUG-49960 getColumnName\(\)으로 한글로 된 컬럼의 이름을 가져오면 한글이 깨지고 SQLException: Invalid column name 에러가 발생합니다.](#)
- [Changes](#)
  - [Version Info](#)
  - [호환성](#)
  - [프로퍼티](#)
  - [성능 분](#)

# New Features

---

**BUG-49963 aku(Altibase Kubernetes Utility)가 추가되었습니다.**

**module**

rp

**Category**

Usability

**재현 빈도**

Always

**설명**

aku(Altibase Kubernetes Utility)는 쿠버네티스의 스테이트풀셋(Statefulset)에서 스케일링(scaling)할 때 파드(Pod) 생성 및 종료에 따라 Altibase의 데이터를 동기화하거나 동기화 정보를 초기화하는 등의 작업을 수행할 수 있게 도와주는 유틸리티입니다. 보다 자세한 내용은 [Utilities Manual-3.aku](#)를 참고하시기 바랍니다.

# Fixed Bugs

**BUG-49910** INSERT문의 바인드 파라미터를 LOB 데이터 타입으로 바인드할 때 INSERT문 실행이 실패했음에도 레코드가 삽입되는 현상을 수정합니다.

module

qp-dml-execute

Category

Functional Error

재현 빈도

Always

설명

INSERT문의 바인드 파라미터를 LOB 데이터 타입으로 바인드할 때 INSERT 수행이 실패했다는 에러가 발생하지만 실제로는 레코드가 삽입되는 현상을 수정합니다.

이 버그는 바인드 파라미터의 SQL 데이터 타입을 실제 컬럼의 데이터 타입과 다른 LOB 데이터 타입으로 바인드할 때 발생합니다. 이 버그가 반영된 Altibase 서버 7.1.0.8.1 이상에서 버그 조건에 해당하는 같은 동작 수행 시 SQL 수행 결과가 달라집니다.

재현 방법

- 재현 절차

```
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)*/
```

```

        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);

```

- 수행 결과

```
$ demo_ex2
```

```
Error : 187 : SQLExecute
```

```
Diagnostic Record 1
```

```
SQLSTATE      : HY000
```

```
Message text : LobLocator cannot span the transaction 0.
```

```
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
```

- 예상 결과

```
$ 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

없음

## 변경사항

- Performance view
- Property
- Compile Option
- Error Code

**BUG-49911 DatabaseMetaData.getColumns 메소드의 IS\_AUTOINCREMENT, IS\_GENERATEDCOLUMN 컬럼값 반환 시 SQLException: Invalid column name 에러가 발생합니다.**

## module

mm-jdbc

## Category

Functionality

## 재현 빈도

Frequency

## 설명

DatabaseMetaData.getColumns 메소드의 IS\_AUTOINCREMENT, IS\_GENERATEDCOLUMN 컬럼값 반환 시 SQLException: Invalid column name 에러가 발생하는 현상을 수정합니다.

이 버그는 아래 버전에 해당하는 JDBC 드라이버를 사용할 때 발생합니다.

- JDBC 4.2 API를 부분 지원하는 Altibase 7.1 JDBC 드라이버(Altibase42.jar)
- Altibase 7.2 JDBC 드라이버

추가로, 이 버그에서는 getProcedures() 메소드의 반환 결과에서 SPECIFIC\_NAME, PROCEDURE\_TYPE 컬럼의 반환 순서를 JDBC 4.2 API 명세에서 정의한 순서대로 반환하도록 수정하였습니다. 애플리케이션 코드에 따라 수행 결과가 달라질 수 있습니다. 이 버그 반영 전/후 SPECIFIC\_NAME, PROCEDURE\_TYPE 컬럼의 반환 순서는 아래와 같습니다.

버그 반영 전	버그 반영 후
rs.getString(8) ==> SPECIFIC_NAME rs.getString(9) ==> PROCEDURE_TYPE	rs.getString(8) => PROCEDURE_TYPE rs.getString(9) => SPECIFIC_NAME

본 버그를 적용하려면 Altibase JDBC 드라이버를 패치해야 합니다.

## 재현 방법

- 재현 절차
- 수행 결과
- 예상 결과

## Workaround

없음

## 변경사항

- Performance view
- Property
- Compile Option
- Error Code

**BUG-49926 MEMORY\_ALLOCATOR\_TYPE 프로퍼티의 최대값을 변경합니다.**

## module

id

## Category

Fatal

## 재현 빈도

Always

## 설명

MEMORY\_ALLOCATOR\_TYPE 프로퍼티의 최대값을 1에서 0으로 변경합니다. 이 버그가 적용된 Altibase 서버 7.1.0.8.1 이상에서 Altibase 서버 프로퍼티 파일(altibase.properties)에 MEMORY\_ALLOCATOR\_TYPE=1을 추가한 경우 Altibase 서버 구동 시 Property [MEMORY\_ALLOCATOR\_TYPE] 1 Overflowed the Value Range.(0~0) 에러가 발생합니다.

## 재현 방법

- 재현 절차
- 수행 결과
- 예상 결과

## Workaround

없음

## 변경사항

- Performance view
- Property
- Compile Option
- Error Code

**BUG-49939 GROUP BY GROUPING SETS 절과 ORDER BY NULLS FIRST 절 또는 ORDER BY NULLS LAST 절을 같이 사용할 때 ERR-31001 : SQL syntax error 에러가 발생합니다.**

## module

qp-select

## Category

Functional Error

## 재현 빈도

Always



## 설명

GROUP BY GROUPING SETS 절과 ORDER BY NULLS FIRST 절 또는 ORDER BY NULLS LAST 절을 같이 사용할 때 ERR-31001 : SQL syntax error 에러가 발생하는 현상을 수정합니다.

## 재현 방법

- 재현 절차

```
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;
```

- 수행 결과

```
[ERR-31001 : SQL syntax error

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

]
```

- 예상 결과

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

아래와 같이 쿼리를 변환하여 버그를 회피할 수 있습니다.

```
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;
```

## 변경사항

- Performance view
- Property
- Compile Option
- Error Code

## BUG-49940 ALTER TABLE ~ ADD COLUMN 수행 시 컬럼의 FIXED/VARIABLE 옵션을 결정하는 프로퍼티를 추가합니다.

### module

qp-ddl-dcl-execute

### Category

Functional Error

### 재현 빈도

Always

### 설명

ALTER TABLE ~ ADD COLUMN 수행 시 컬럼의 FIXED/VARIABLE 옵션을 결정하는 프로퍼티를 추가합니다.

이 버그는 메모리 테이블에만 영향이 있습니다.

- 이름

\_\_FORCE\_VARIABLE\_FOR\_DDL\_COLUMN

- 설명

메모리 테이블에 ALTER TABLE ~ ADD COLUMN 수행 시 컬럼의 FIXED/VARIABLE 옵션을 결정합니다.

0 :

추가할 컬럼의 크기가 IN ROW 절(?) 또는 MEMORY\_VARIABLE\_COLUMN\_IN\_ROW\_SIZE에서 지정한 크기보다 작거나 같으면 FIXED 컬럼으로 생성하고 실시간 DDL이 동작하지 않습니다.

실시간 DDL이 동작하지 않으면 Altibase 서버는 ADD COLUMN 작업을 아래와 같이 처리합니다.

- ALTER TABLE ~ ADD COLUMN 수행 시 내부적으로 테이블을 재구성합니다.
- ALTER TABLE ~ ADD COLUMN 수행 시 대상 테이블에 X 잠금을 획득하므로 ADD COLUMN이 완료할 때까지 테이블에 접근 할 수 없습니다.
- ALTER TABLE ~ ADD COLUMN 수행 시 데이터 크기에 비례하여 메모리 사용량이 증가합니다.

1 : 컬럼 크기에 상관없이 무조건 VARIABLE로 컬럼을 생성합니다. ADD COLUMN 수행 시 실시간 DDL이 동작합니다.

실시간 DDL은 메모리 테이블을 대상으로 다른 트랜잭션에 영향을 주지 않고 ADD COLUMN 작업이 수행되는 것을 말합니다.

- 기본값

1

- 속성

읽기 전용, **비공개**

- 참고 사항

이 버그에서 추가된 프로퍼티를 적용하려면 Altibase 서버 설정 파일에  
\_FORCE\_VARIABLE\_FOR\_DDL\_COLUMN = 1 을 추가하고 Altibase 서버를 재시작해야 합니다. 값을 0으로  
변경 시 ADD COLUMN 처리 방식이 변경되는 것을 주의해야 합니다.

이 프로퍼티는 비공개 프로퍼티로 매뉴얼에 설명을 추가하지 않습니다.

## 재현 방법

- 재현 절차
- 수행 결과
- 예상 결과

## Workaround

없음

## 변경사항

- Performance view
- Property
- Compile Option
- Error Code

**BUG-49946 ado.net에서 PSM 수행 시 SqlDataReader.NextResult()에서 잘못된 결과를 반환합니다.**

## module

ux-win-adonet

## Category

Functional Error

## 재현 빈도

Always

## 설명

ado.net에서 PSM 수행 시 DbDataReader.NextResult()에서 잘못된 결과를 반환하는 문제를 수정합니다. 버그 조건을 만족하는 애플리케이션 수행 시 버그 반영 전/후 결과가 달라질 수 있습니다.

## 재현 방법

- 재현 절차

```
create table proctest_t1 (c1 integer, c2 varchar(10))
insert into proctest_t1 values (11, 'a1')
insert into proctest_t1 values (12, 'a2')
insert into proctest_t1 values (13, 'a3')
create table proctest_t2 (c1 integer, c2 varchar(10))
insert into proctest_t2 values (21, 'b1')
insert into proctest_t2 values (22, 'b2')
insert into proctest_t2 values (23, 'b3')
create typeset proctest_type
as
type proctest_cur is ref cursor;
end
create procedure proctest_resultsets
(p1 out proctest_type.proctest_cur,
 p2 out proctest_type.proctest_cur)
as
sql_stmt varchar(200)
begin
sql_stmt := 'SELECT * FROM proctest_t1';
open p1 for sql_stmt;
sql_stmt := 'SELECT * FROM proctest_t2';
open p2 for sql_stmt;
end
using (DbCommand sCmd = Connection.CreateCommand())
{
    sCmd.CommandType = CommandType.Text;
    sCmd.CommandText = "EXEC proctest_resultsets";
    using (DbDataReader sReader = sCmd.ExecuteReader())
    {
        AssertEquals(true, sReader.NextResult());
        AssertEquals(false, sReader.NextResult());
    }
}
```

```
sReader.NextResult() 결과 false
```

- 예상 결과

```
sReader.NextResult() 결과 true
```

## Workaround

없음

## 변경사항

- Performance view
- Property
- Compile Option
- Error Code

**BUG-49960 getColumnLabel()으로 한글로 된 컬럼의 이름을 가져오면 한글이 깨지고 SQLException: Invalid column name 에러가 발생합니다.**

## module

mm-jdbc

## Category

Functional Error

## 재현 빈도

Always

## 설명

Altibase 서버와 클라이언트의 캐릭터셋이 다를 때 JDBC에서 한글로 된 컬럼의 이름을 가져오면 한글이 깨지는 현상을 수정합니다. 이 버그는 ResultSetMetaData 인터페이스의 다음 메소드들을 사용할 때 영향이 있습니다.

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

본 버그를 적용하려면 Altibase JDBC 드라이버를 패치해야 합니다.

## 재현 방법

- 재현 절차

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

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

```
### 예제 코드 CharacterSetTest.java 일부
[source encoding = utf8]
Connection sCon = getAltConnection();
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
```

- 수행 결과

```
$ java CharacterSetTest
Col1 Name==>占시뤼엡1
Exception in thread "main" java.sql.SQLException: Invalid column name: 而ㅁ??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
    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)
```

- 예상 결과

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

## Workaround

`-Dfile.encoding=euc-kr` 옵션을 사용하여 버그를 회피할 수 있습니다.

## 변경사항

- 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

Altibase 7.1 패치 버전별 히스토리는 [Version Histories](#) 에서 확인할 수 있다.

## 호환성

### Database binary version

데이터베이스 바이너리 버전은 변경되지 않았다.

데이터베이스 바이너리 버전은 데이터베이스 이미지 파일과 로그파일의 호환성을 나타낸다. 이 버전이 다른 경우의 패치(업그레이드 포함)는 데이터베이스를 재구성해야 한다.

### Meta Version

메타 버전은 변경되지 않았다.

패치를 롤백하려는 경우, [메타다운그레이드](#)를 참고한다.

### CM protocol Version

통신 프로토콜 버전은 변경되지 않았다.

### Replication protocol Version

Replication 프로토콜 버전은 변경되지 않았다.

## 프로퍼티

추가된 프로퍼티

변경된 프로퍼티

삭제된 프로퍼티

## 성능 뷰

추가된 성능 뷰



변경된 성능 뷰

삭제된 성능 뷰