Chapter 2. SQL Return Codes

The information in this chapter is General-use Programming Interface and Associated Guidance Information, as defined in “Appendix E. Notices” on page 1331.

Introduction

Structured Query Language (SQL) return codes, and the tokens referred to in the explanations, are returned in the SQL communication area (SQLCA).

The SQLCA is an area in the application program (defined by the application program) for the use of DB2. It is described in Appendix C of *DB2 SQL Reference* .

The SQL return code is returned in the SQLCODE field of the SQLCA. The tokens are returned in the SQLERRM field. If there are several tokens, they appear sequentially in SQLERRM in the order in which they appear in the message text. The tokens are separated by the character X'FF'.

A token appears in the message text in lowercase letters. When an SQL return code is returned through SPUFI, the tokens have been substituted into the message text and the text is displayed. The substitution of tokens into message text is performed by a DB2 module named DSNTIAR. This module can also be used by application programs. Refer to Part 2 of *DB2 Application Programming and SQL Guide* for more information about DSNTIAR.

Checking the execution of SQL statements

An application program containing executable SQL statements must either provide a structure named SQLCA or a stand-alone integer variable named SQLCODE (SQLCOD in FORTRAN). An SQLCA may be provided by using the INCLUDE SQLCA statement. INCLUDE SQLCA must not be used if a stand-alone SQLCODE is provided.

The SQLCA includes an integer variable named SQLCODE. The option of providing a stand-alone SQLCODE instead of an SQLCA allows for conformance with the ISO/ANSI SQL standard. The use of this option must be indicated to the precompiler.

SQLCODE

Regardless of whether the application program provides an SQLCA or a stand-alone variable, SQLCODE is set by DB2 after each SQL statement is executed. DB2 conforms to the ISO/ANSI SQL standard as follows:

- If SQLCODE = 0, execution was successful.
- If SQLCODE > 0, execution was successful with a warning.
- If SQLCODE < 0, execution was not successful.
- SQLCODE = 100, ″no data″ was found. For example, a FETCH statement returned no data because the cursor was positioned after the last row of the result table.

SQLSTATE

SQLSTATE is also set by DB2 after the execution of each SQL statement. Thus, application programs can check the execution of SQL statements by testing SQLSTATE instead of SQLCODE. SQLSTATE (SQLSTT in FORTRAN) is a 5-byte character string variable in the SQLCA.

SQLSTATE provides application programs with common codes for common error conditions (the values of SQLSTATE are product-specific only if the error or warning is product-specific). Furthermore, SQLSTATE is designed so that application programs can test for specific errors or classes of errors. The coding scheme is the same for all IBM relational database products. See “Appendix C. SQLSTATE Values—Common Error Codes” on page 1301 for more information and a complete list of the possible values of SQLSTATE.

Successful execution SQL code

000 SUCCESSFUL EXECUTION

Explanation: SQL code 000 signifies the unqualified successful execution or successful execution of an SQL statement with one or more warnings. If SQLWARN0 is blank, then no warning situation exists, although SQLWARNx fields may return other information. If SQLWARN0 = W, at least one of the other warning indicators in the SQLCA has been set to indicate a warning condition. For example, SQLWARN1 is used to indicate that a value of a string column was truncated when assigned to a host variable. SQLWARNx fields are described in Appendix C of *DB2 SQL Reference*.

SQLSTATE: 00000 for unqualified successful execution

SQLSTATE: 01ddd for successful execution with warning 'ddd'.

+012 THE UNQUALIFIED COLUMN NAME *column-name* WAS INTERPRETED AS A CORRELATED REFERENCE

Explanation: The column name does not identify a column of a table or view in the FROM clause of the subquery. However, it does identify a column of a table or view in a FROM clause at a higher level in the statement.

System Action: The column name is interpreted as a correlated reference.

Programmer Response: If DB2’s interpretation of the column name was not what you intended, rewrite the SQL statement and submit it again. If you intend the column name to refer to a table named at a higher level, we advise rewriting the statement anyway, using a table name or correlation name as a qualifier for the column name. The unqualified column name could be interpreted differently if you do a rebind after altering one of the tables to which you refer.

SQLSTATE: 01545

+098 A DYNAMIC SQL STATEMENT ENDS WITH A SEMICOLON.

Explanation: The statement string of a PREPARE or EXECUTE IMMEDIATE statement is a valid dynamic SQL statement, but it ends with a semicolon.

System Action: The semicolon and any subsequent text are ignored.

Programmer Response: Check that the semicolon is being used as a statement terminator.

SQLSTATE: 01568

+100 ROW NOT FOUND FOR FETCH, UPDATE OR DELETE, OR THE RESULT OF A QUERY IS AN EMPTY TABLE

Explanation: One of the following conditions occurred:

- No row met the search conditions specified in an UPDATE or DELETE statement.
- The result of a SELECT INTO statement was an empty table.
- A FETCH statement was executed when the cursor was positioned after the last row of the result table.
- The result of the subselect of an INSERT statement is empty.
- A scrollable cursor is positioned before the first row of a table.

When a SELECT statement is executed using SPUFI, this SQLCODE indicates normal completion.

System Action: No data was retrieved, updated, or deleted.

SQLSTATE: 02000

+110 **SQL UPDATE TO A DATA CAPTURE TABLE NOT SIGNALED TO ORIGINATING SUBSYSTEM**

Explanation: DataPropagator (DPropNR) exit routine issued an SQL INSERT, UPDATE, or DELETE statement to a table defined with DATA CAPTURE CHANGES. Since data capture is already in progress, notification is not sent back to the originating IMS subsystem.

System Action: DB2 changes the data and records the change in the log. DB2 does not notify DPropNR's exit routine of the change, because doing so might cause the same change to be made again.

SQLSTATE: 01561

+111 **THE SUBPAGES OPTION IS NOT SUPPORTED FOR TYPE 2 INDEXES**

Explanation: You cannot use the SUBPAGES option for type 2 indexes.

System Action: The option is ignored and processing continues.

Programmer Response: Remove the SUBPAGES option to get rid of the warning.

SQLSTATE: 01590

+117 **THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS**

Explanation: The number of insert values in the value list of the INSERT statement is not the same as the number of object columns specified.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, specify one and only one value for each of the specified object columns.

SQLSTATE: 01525

+162 **TABLESPACE** *database-name.tablespace-name* **HAS BEEN PLACED IN CHECK PENDING**

Explanation: The indicated table space is in check pending status because the ALTER TABLE statement was used to specify a referential constraint or a check constraint (while special register CURRENT RULES = 'DB2') on a populated table. The table space is not generally available until the check pending status is removed from the table space.

System Action: The table space was placed in check pending status.

Programmer Response: Run the CHECK DATA utility. The enforcement of the referential constraint or the check constraint is deferred until the CHECK DATA utility is run.

SQLSTATE: 01514

+203 **THE QUALIFIED COLUMN NAME** *column-name* **WAS RESOLVED USING A NON-UNIQUE OR UNEXPOSED NAME**

Explanation: The table designator selected to resolve a qualified column name is one of the following:

- An unexposed name
- An exposed name that has an exposed duplicate in the same FROM clause
- An exposed name that has an unexposed duplicate which appears before the selected name in the ordered list of names to which the qualifier is compared

Therefore, the statement does not conform to the guidelines for using only unique exposed names as qualifiers or it is possible that the column reference was not resolved to the intended instance of the table or view.

System Action: DB2 uses the selected name to resolve the reference.

Programmer Response: If DB2's resolution of the qualifier was not what you intended, rewrite the SQL statement and submit it again. The rules used to resolve column name qualifiers are given in Chapter 3 of *DB2 SQL Reference*.

SQLSTATE: 01552

+204 *name* **IS AN UNDEFINED NAME**

Explanation: The object identified by 'name' is not defined in the DB2 subsystem. This return code can be generated for any type of DB2 object.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, determine that the object name was correctly specified in the SQL statement (including any required qualifiers). If so, ensure that the object exists in the system before resubmitting the statement.

SQLSTATE: 01532

+206 *column-name* **IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE**

Explanation: This return code is used to report one of these errors:

- In the case of an INSERT or UPDATE statement, the specified column is not a column of the table or view that was specified as the object of the insert or update.
- In the case a SELECT or DELETE statement, the specified column is not a column of any of the tables or views identified in a FROM clause in the statement.
- There is a correlated reference in GROUP BY.
- There is an unresolved qualified reference in HAVING.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, verify that the column and table names are specified correctly in the SQL statement. In the case of a SELECT statement, check to be sure that all of the required tables were named in the FROM clause.

SQLSTATE: 01533

+218 **THE SQL STATEMENT REFERENCING A REMOTE OBJECT CANNOT BE EXPLAINED**

Explanation: The user has used EXPLAIN(YES) on the bind subcommand to bind an application which has SQL statement referencing a remote object or the user has a static EXPLAIN SQL statement which references a remote object in the application program. EXPLAIN on a remote object is not supported by DB2.

It is issued at BIND time, and only with VALIDATE(RUN).

System Action: The plan or package will be bound successfully, but no information will be filled in the user's PLAN-TABLE for the SQL statement referencing a remote object. An SQLCODE -512 will be issued at run time if the EXPLAIN statement is found to explain a remote object.

SQLSTATE: 01537

+219 **THE REQUIRED EXPLANATION TABLE** *table-name* **DOES NOT EXIST**

Explanation: The EXPLAIN statement assumes the existence of the explanation table and it is not defined in the DB2 subsystem as a base table. Refer to Chapter 6 of *DB2 SQL Reference* for more information.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, determine whether the required explanation table does exist. If not, create the required table.

SQLSTATE: 01532

+220 **THE COLUMN** *column-name* **IN EXPLANATION TABLE** *table-name* **IS NOT DEFINED PROPERLY**

Explanation: An error occurred during the insertion of a row into the explanation table. The table is improperly defined for the following reasons:

- A column is missing.
- Columns are defined in the wrong order.
- The table contains an extra column.
- A column description is invalid because of its name, data type, length, or null attributes.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, correct the definition of the required explanation table. Refer to Chapter 6 of *DB2 SQL Reference* for information about defining an explanation table.

SQLSTATE: 01546

+222 **HOLE DETECTED USING** *cursor-name*

Explanation: DB2 could not FETCH data for cursor *cursor-name* because DB2 has detected a delete hole or an update hole. DB2 detects these holes when DB2 tries to refetch the current row of the result table for cursor *cursor-name*, and cannot locate the corresponding row of the underlying table.

A delete hole occurs when the corresponding row of the underlying table has been deleted.

An update hole occurs when the corresponding row of the underlying table has been updated, and the updated row no longer satisfies the search condition that is specified in the SELECT statement of the cursor.

System Action: The statement cannot be processed. The cursor is repositioned on the hole.

Programmer Response: Correct the application program to handle this situation or change isolation levels so the base row cannot be deleted during the cursor operation.

SQLSTATE: 02502

+231 **CURSOR POSITION OF CURSOR *cursor-name* IS NOT VALID FOR FETCH OF THE CURRENT ROW**

Explanation: The cursor was not positioned on a row, and either FETCH CURRENT or FETCH RELATIVE 0 was specified. No data was fetched.

cursor-name
Name of the cursor used for the FETCH statement.

System Action: The statement cannot be processed. The cursor position is unchanged.

Programmer Response: Correct the application program to establish position before issuing this FETCH statement.

SQLSTATE: 02000

+236 **SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED FOR *integer3* COLUMNS**

Explanation: The value of the SQLN field of the SQLDA should be at least as large as the number of columns being described. *integer3* is the number of columns being described.

In the case that USING BOTH has been specified, twice as many SQLVAR entries are needed as the number of columns being described.

The number of SQLVAR entries that are needed to return all of the information about the columns is *integer2*.

Note: in the case of DESCRIBE INPUT, each reference to *column* would actually be *parameter*.

System Action: The SQLDAID 7th byte has been set to ″on″ with a value of 2 indicating that 2 SQLVAR entries are needed for each column. DB2 has not set any SQLVAR entries.

Programmer Response: Increase the value of the SQLN field in the SQLDA to the value indicated in the message (making sure the SQLDA is large enough to support that amount) and resubmit the statement.

SQLSTATE: 01005

+237 **SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A DISTINCT TYPE**

Explanation: Given that at least one of the columns being described is a distinct type, space should be provided for the *extended* SQLVAR entries in addition to the *base* SQLVAR entries. The value of SQLN, *integer1*, indicates that there are not enough SQLVAR entries for the base and extended SQLVAR entries.

The total number of SQLVAR entries that are required depends on whether USING BOTH was specified (*n* is the number of columns being described):

- With USING BOTH, space should be allocated for *3n* SQLVAR entries.
- Otherwise, space should be allocated for *2n* SQLVAR entries.

The number of SQLVAR entries that are needed to return all of the information about the columns is *integer2*.

Note: in the case of DESCRIBE INPUT, each reference to *column* would actually be *parameter*.

System Action: DB2 has set the SQLDAID 7th byte flag ″on″ because sufficient space was not provided for the *extended* SQLVAR entries. The value of the 7th byte flag indicates how many SQLVAR entries are needed for each column. Additionally, because there were enough SQLVAR entries for the *base* SQLVARs, DB2 has set the fields of the *base* SQLVAR entries. However, DB2 has not set any *extended* SQLVAR entries.

Programmer Response: If there is no need for the additional information about the distinct type(s), then no action is required unless USING BOTH was specified (in which case additional SQLVAR entries must be provided for the labels).

If the distinct type information is needed, the value of the SQLN field in the SQLDA should be increased to *integer2* (after making sure that the SQLDA is large enough to support that amount) and the statement should be resubmitted.

SQLSTATE: 01594

+238 **SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* SQLVAR ENTRIES ARE NEEDED FOR *integer3* COLUMNS BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A LOB**

Explanation: Given that at least one the columns being described is a LOB, space must be provided for the *extended* SQLVAR entries in addition to the *base* SQLVAR entries. The value of SQLN, *integer1*, indicates that there are not enough SQLVAR entries for the base and extended SQLVAR entries. One or more of the columns being described may be a distinct type.

The total number of SQLVAR entries that are required depends on whether USING BOTH was specified or not (*n* is the number of columns being described which is *integer3*), and whether the columns include any distinct types:

- With USING BOTH, and one or more distinct types, space should be allocated for *3n* SQLVAR entries.

- Otherwise, space should be allocated for *2n* SQLVAR entries.

The number of SQLVAR entries that are needed to return all of the information about the columns is *integer2*.

Note: in the case of DESCRIBE INPUT, each reference to *column* would actually be *parameter*.

System Action: DB2 has set the SQLDAID 7th byte flag ″on″ because sufficient space was not provided for the *extended* SQLVAR entries. The value of the 7th byte flag indicates how many SQLVAR entries are needed for each column. DB2 has not set any SQLVAR entries.

Programmer Response: Increase the value of the SQLN field in the SQLDA to *integer2* (after making sure that the SQLDA is large enough to support that amount) and resubmit the statement.

SQLSTATE: 01005

+239 **SQLDA INCLUDES** *integer1* **SQLVAR ENTRIES, BUT** *integer2* **ARE REQUIRED FOR** *integer3* **COLUMNS BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A DISTINCT TYPE**

Explanation: Given that at least one of the columns being described is a distinct type, space should be provided for the *extended* SQLVAR entries in addition to the *base* SQLVAR entries. The value of SQLN, *integer1*, indicates that there are not enough SQLVAR entries for the base and extended SQLVAR entries.

The total number of SQLVAR entries that are required depends on whether USING BOTH was specified or not (*n* is the number of columns being described which is *integer3*),

- With USING BOTH, space should be allocated for *3n* SQLVAR entries.
- Otherwise, space should be allocated for *2n* SQLVAR entries.

The number of SQLVAR entries that are needed to return all of the information about the columns is *integer2*.

Note: in the case of DESCRIBE INPUT, each reference to *column* would actually be *parameter*.

System Action: DB2 has set the SQLDAID 7th byte flag ″on″ because sufficient space was not provided for the *extended* SQLVAR entries. The value of the 7th byte flag indicates how many SQLVAR entries are needed for each column. DB2 has not set any SQLVAR entries.

Programmer Response: If the distinct type information is needed, the value of the SQLN field in the SQLDA should be increased to *integer2* (after making sure the SQLDA is large enough to support that amount) and the statement should be resubmitted.

If there is no need for the additional information about the distinct type(s) in the result set, then it is possible to resubmit the statement only providing enough SQLVAR entries to accommodate the number of columns being described (i.e. provide the necessary number of *base* SQL entries).

SQLSTATE: 01005

+304 **A VALUE WITH DATA TYPE** *data-type1* **CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION** *position-number* **WITH DATA TYPE** *data-type2*

Explanation: A FETCH or SELECT into a host variable list or structure, position number *position-number* failed because the host variable having data type *data-type2* was not large enough to hold the retrieved value having data type *data-type1*.

System Action: The FETCH or SELECT could not return the data for the indicated SELECT item, the indicator variable is set to negative two (-2) to indicate a null value returned. Processing continues.

Programmer Response: Verify that table definitions are current, and that the host variable has the proper data type. See the explanation for SQLCODE -405 for ranges of SQL data types.

SQLSTATE: 01515

+331 **THE NULL VALUE HAS BEEN ASSIGNED TO A HOST VARIABLE BECAUSE THE STRING CANNOT BE TRANSLATED. REASON** *reason-code***, CHARACTER** *code-point***, HOST VARIABLE** *position-number*

Explanation: A string assigned to a host variable had to be translated from its coded character set to the coded character set of the host variable and an error occurred during the translation. The *position-number* is the ordinality of the host variable in the SQLDA. See the description of SQLCODE -331 for further information including the meaning of the *reason-code* and *code-point*.

System Action: The host variable is unchanged and its indicator variable is set to -2 to indicate that a null value is returned. Execution of the statement continues as if the translation error had not occurred.

SQLSTATE: 01520

+335 **DB2 CONVERTED A HOST VARIABLE, PARAMETER, OR COLUMN NUMBER *var-num var-name-or-num* TO COLUMN NAME, HOST VARIABLE, OR EXPRESSION NUMBER *col-name-or-num* FROM *from ccsid* TO *to-ccsid*, AND RESULTING IN SUBSTITUTION CHARACTERS.**

Explanation: A translation error occurred during the conversion of a string to a different coded character set. One or more substitution characters have been placed in the string during the conversion process.

System Action: DB2 processes the statement successfully.

Programmer Response: DB2 processes the SQL statement, but used substitution characters instead one or more characters as a result of character conversion from *from ccsid* to *to-ccsid*. If substitution is acceptable, then no action is necessary. If substitution is not acceptable, then issue a ROLLBACK. Ensure that data being provided to DB2 is convertible from *from ccsid* to *to-ccsid* without data loss.

SQLSTATE: 01517

+339 **THE SQL STATEMENT HAS BEEN SUCCESSFULLY EXECUTED, BUT THERE MAY BE SOME CHARACTER CONVERSION INCONSISTENCIES**

Explanation: The application is connected to a DB2 Version 2 Release 3 database server. The SQL statement is using an alias or three-part name, which refers to another DB2 subsystem that is at the Version 2 Release 2 level. DB2 Version 2 Release 2 does not support character conversion. Since the execution of SQL statements from an EBCDIC DRDA requester to an EBCDIC Version 2 Release 2 DB2 server could require character conversion, a warning is generated. If the requester system CCSID is inconsistent with the DB2 Version 2 Release 2 environment, most EBCDIC character code points match to the same character. Only certain special characters typically lead to data integrity concerns. Therefore, a warning is generated.

System Action: The statement is successfully executed.

Programmer Response: If the DB2 Version 2 Release 2 table is accessed from other environments (different CCSID), incorrect results might occur (relative to what the other environment might expect). You should understand what characters might not be consistent with the DB2 Version 2 Release 2 environment (its inherent CCSID) and avoid use of those characters or understand the exposure you face if you use them.

System Programmer Response: If the application must refer to the Version 2 Release 2 subsystem data, the Version 2 Release 2 DB2 subsystem can be migrated to Version 2 Release 3 where character conversion is supported.

SQLSTATE: 01569

+394 **USER SPECIFIED OPTIMIZATION HINTS USED DURING ACCESS PATH SELECTION**

Explanation: Normal access path selection was bypassed in favor of an access path specified in the PLAN_TABLE.

System Action: Processing continues normally.

Programmer Response: Ensure that the access path is correct and produces the correct results.

SQLSTATE: 01629

+395 **USER SPECIFIED OPTIMIZATION HINTS ARE INVALID (REASON CODE = *reason-code*). THE OPTIMIZATION HINTS ARE IGNORED.**

Explanation: The optimization hints specified for this query are invalid. A*reason-code* in the following table can help identify why the hints were invalid:

reason-code	Description
2	TABNO is invalid.
3	TNAME is not specified.
4	TNAME is ambiguous.
5	TABNO doesn't agree with TNAME.
6	QBLOCKNO doesn't agree with TNAME.
7	PAGE_RANGE is invalid.
8	PREFETCH is invalid.
9	METHOD is invalid.
10	SORTN_JOIN is invalid.
11	SORTC_JOIN is invalid.
12	ACCESSTYPE is invalid.
13	ACCESSCREATOR or ACCESSNAME is invalid.
14	TYPE 1 index can't be used with isolation level 'UR'.
15	Specified index can't be used as requested.
16	Multi-index access can't be done.
17	Invalid ACCESSTYPE combination.
18	METHOD specified for first table accessed.
19	Nested-loop join can't be done as requested.
20	Merge join can't be done as requested.

21 Hybrid join can't be done as requested.

22 PARALLELISM_MODE requested can't be done.

23 PARALLELISM_MODE is invalid.

24 ACCESS_DEGREE is invalid.

25 JOIN_DEGREE is invalid.

26 A table is missing.

27 A table is redundant.

28 PRIMARY_ACCESSTYPE is invalid.

29 ACCESS_PGROUP_ID is not specified.

30 JOIN_PGROUP_ID is not specified.

31 PARALLELISM_MODE is not specified.

32 CREATOR or TNAME is invalid.

33 Join sequence is incorrect.

34 Full outer join requires merge join method.

35 WHEN_OPTIMIZE is invalid or inconsistent.

99 Unexpected error.

System Action: The user-specified optimization hints are ignored. The access path is determined without the use of hints and processing continues normally.

Programmer Response: Correct the problem with the optimization hints, or disable their use for this query.

SQLSTATE: 01628

+402 LOCATION *location* **IS UNKNOWN**

Explanation: A remote object is referenced and either the table SYSIBM.LOCATIONS is not defined or the referenced 'location' matches no entry in the SYSIBM.LOCATIONS.LOCATION column.

System Action: For the CREATE ALIAS statement, the alias is created. For binding a plan or package with the VALIDATE(RUN) option, the plan or package is created.

SQLSTATE: 01521

+403 THE LOCAL OBJECT REFERENCED BY THE CREATE ALIAS STATEMENT DOES NOT EXIST

Explanation: The local object referenced by the CREATE ALIAS statement does not exist when creating the alias.

System Action: The alias is created.

SQLSTATE: 01522

+434 OPTION *keyword* **IS A DEPRECATED FEATURE**

Explanation: *keyword* is a deprecated feature that will not be supported in releases following DB2 Version 7. It is accepted, but continued use of this keyword is not recommended.

For indexes, use type 2 indexes rather than type 1 indexes to avoid any incompatibilities.

System Action: Processing continues normally.

Programmer Response: No change is required for the current release. However, you should change your SQL statement and remove this feature to prepare for future releases when this feature is not supported.

SQLSTATE: 01608

+445 VALUE *value* **HAS BEEN TRUNCATED**

Explanation: The value *value* was truncated by a cast function, which was called to transform the value in some way. This is a warning situation. The cast function is a result of

- a CAST specification
- a built-in function such as CHAR, VARCHAR, etc.
- a CAST FROM specification on the CREATE FUNCTION statement that created the function
- a user-defined function that is sourced on another function and the result needed to be transformed.

If 'value' has the 'for bit data' subtype, then the 'value' is printed as a hexadecimal string in quotes followed by an X.

System Action: The value has been truncated.

Programmer Response: Ensure that the output is as expected and that the truncation has not caused any unexpected consequences.

SQLSTATE: 01004

+462 EXTERNAL FUNCTION OR PROCEDURE *name* **(SPECIFIC NAME** *specific-name***) HAS RETURNED A WARNING SQLSTATE, WITH DIAGNOSTIC TEXT** *text*

Explanation: An SQLSTATE of the form *01Hxx* was returned to DB2 by user-defined function or procedure *name*, along with message text *text*.

System Action: Processing continues.

Programmer Response: See your database administrator, or the author of the function or procedure to find out the meaning of the warning. The significance of the bad SQLSTATE to the invoking application can be learned from the author of the function or procedure.

SQLSTATE: 01Hxx

+464 **PROCEDURE** *proc* **RETURNED** *num* **QUERY RESULT SETS, WHICH EXCEEDS THE DEFINED LIMIT** *integer*

Explanation: The stored procedure named by *proc* completed normally. However, the stored procedure exceeded the defined limit on the number of query result sets the procedure can return.

- *num* identifies the number of query result sets returned by the stored procedure.
- *integer* identifies the defined limit on the number of query result sets for the stored procedure.

Only the first *integer* query result sets are returned to the SQL program that issued the SQL CALL statement.

The possible causes are as follows:

- The stored procedure is unable to return *num* result sets due to the limit defined for the procedure.
- The stored procedure is unable to return *num* result sets due to the DRDA limitations imposed by the client. The DRDA client establishes this limit with the MAXRSLCNT DDM code point.

System Action: The SQL statement is successful. The SQLWARN9 field is set to 'Z'.

SQLSTATE: 0100E

+466 **PROCEDURE** *proc* **RETURNED** *num* **QUERY RESULTS SETS**

Explanation: The stored procedure named by *proc* completed normally. The procedure returned the number of SQL query result sets specified in *num*.

System Action: The SQL statement is successful. The SQLWARN9 field is set to 'Z'.

SQLSTATE: 0100C

+494 **NUMBER OF RESULT SETS IS GREATER THAN NUMBER OF LOCATORS**

Explanation: The number of result set locators specified on the ASSOCIATE LOCATORS statement is less than the number of result sets returned by the stored procedure. The first ″n″ result set locator values are returned, where ″n″ is the number of result set locator variables specified on the SQL statement.

System Action: The SQL statement is successful. The SQLWARN3 field is set to 'Z'.

Programmer Response: Increase the number of result set locator variables specified on the SQL statement.

SQLSTATE: 01614

+495 **ESTIMATED PROCESSOR COST OF** *estimate-amount1* **PROCESSOR SECONDS (***estimate-amount2* **SERVICE UNITS) IN COST CATEGORY** *cost-category* **EXCEEDS A RESOURCE LIMIT WARNING THRESHOLD OF** *limit-amount* **SERVICE UNITS**

Explanation: The prepare of a dynamic INSERT, UPDATE, DELETE, or SELECT SQL statement resulted in a cost estimate that exceeded the warning threshold value specified in the resource limit specification table (RLST). This warning is also issued if DB2's cost category value was "B", and the default action specified in the RLF_CATEGORY_B column in the RLST is to issue a warning.

estimate_amount1
The cost estimate (in processor seconds) if the prepared INSERT, UPDATE, DELETE or SELECT statement were to be executed.

estimate_amount2
The cost estimate (in service units) if the prepared INSERT, UPDATE, DELETE or SELECT statement were to be executed.

cost-category
DB2's cost-category for this SQL statement. The possible values are A or B.

limit-amount
The warning threshold (in service units) specified in the RLFASUWARN column of the RLST.

System Action: The prepare of the dynamic INSERT, UPDATE, DELETE, or SELECT statement was successful. An SQLCODE -905 might be issued if the execution of the prepared statement exceeds the ASUTIME value specified in the RLST.

Programmer Response: Ensure that there is application logic to handle the warning to either allow the statement to execute or to stop the statement from being executed. If this SQLCODE was returned because the cost category value is "B", it might be that the statement is using parameter markers or that some statistics are not available for the referenced tables and columns. Make sure the administrator has run the utility RUNSTATS on the referenced tables. It might also be that UDFs will be invoked when the statement is executed, or for INSERT, UPDATE, or DELETE statements that triggers are defined on the changed table. Check the DSN_STATEMNT_TABLE or the IFCID 22 record for this statement to find the reasons this SQL statement has been put in cost category "B".

User Response: If the warning is caused by an SQL statement that is consuming too much processor resource, attempt to rewrite the statement to perform more efficiently. Another option is to ask the administrator to increase the warning threshold value in the RLST.

SQLSTATE: 01616

+535 **THE RESULT OF THE POSITIONED UPDATE OR DELETE MAY DEPEND ON THE ORDER OF THE ROWS**

Explanation: A positioned update of a primary key or a delete from a table with a self-referencing constraint was requested.

System Action: DB2 executes the UPDATE or DELETE statement and the contents of the table are changed.

SQLSTATE: 01591

+541 **THE REFERENTIAL OR UNIQUE CONSTRAINT** *name* **HAS BEEN IGNORED BECAUSE IT IS A DUPLICATE**

Explanation: A FOREIGN KEY clause uses the same key and parent table as another FOREIGN KEY clause, or a UNIQUE clause uses the same column list as another UNIQUE or PRIMARY KEY clause. In either case, the duplicate clause is ignored.

name is either the foreign key name or the unique constraint name.

System Action: DB2 continues processing.

Programmer Response: If the duplication is an error, correct the statement and execute it again.

SQLSTATE: 01543

+551 *auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation* **ON OBJECT** *object-name*

Explanation: Authorization ID *auth-id* has attempted to perform the specified *operation* on object *object-name* without having been granted the proper authority to do so. This error might also occur if the specified object does not exist, or if the object is a read-only view (for UPDATE or INSERT). Additionally, the error may occur if *auth-id* is trying to create a table or view with an authorization ID other than its own. You may create a table or view from an *auth-id* other than your own only if your authorization ID is SYSADM, DBADM, or DBCTRL.

If this error occurs while DB2 is creating or altering a table involving referential constraints, this code reports that the user does not have the necessary ALTER privilege to perform a FOREIGN KEY, DROP FOREIGN KEY, DROP PRIMARY KEY, or DROP UNIQUE operation. The *object-name* identifies the object table of the CREATE or ALTER TABLE statement, not the table for which the user lacks the ALTER privilege.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, ensure that *auth-id* has been granted the authority to perform the desired operation, that *object-name* exists, and that *auth-id* is not trying to create a table with a different authorization ID.

SQLSTATE: 01548

+552 *auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation*

Explanation: Authorization ID 'auth-id' has attempted to perform the specified 'operation' without having been granted the authority to do so.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, ensure that the authorization-ID has been granted the authority necessary to perform the desired operation.

SQLSTATE: 01542

+558 **THE WITH GRANT OPTION IS IGNORED**

Explanation: The GRANT statement contained one of the following situations:

- PUBLIC was within the list of 'grantee' authorization IDs.
- BINDAGENT privilege was being granted.
- ANY package privilege on collection-id.* was being granted.

The WITH GRANT option may not be used in the above situations.

System Action: The offending privilege(s) in the authorization specification are granted without the GRANT option. If the grantee is PUBLIC, all the privileges in the authorization specification are granted without the GRANT option.

SQLSTATE: 01516

+561 **THE ALTER, INDEX, REFERENCES, AND TRIGGER PRIVILEGES CANNOT BE GRANTED PUBLIC AT ALL LOCATIONS**

Explanation: You specified a GRANT statement with either an ALL or ALL PRIVILEGES keyword. ALL and ALL PRIVILEGES imply the granting of ALTER, INDEX, REFERENCES, AND TRIGGER privileges that cannot be granted to a remote user.

System Action: DB2 executes the GRANT statement.

However, it does not grant the ALTER, INDEX, REFERENCES, or TRIGGER privileges to PUBLIC*.

SQLSTATE: 01523

+562 **A GRANT OF A PRIVILEGE WAS IGNORED BECAUSE THE GRANTEE ALREADY HAS THE PRIVILEGE FROM THE GRANTOR**

Explanation: At least one of the privileges in the GRANT statement was ignored because the privilege was already granted to the grantee by the grantor.

System Action: The privileges previously granted are ignored; all others are granted.

SQLSTATE: 01560

+585 **THE SCHEMA NAME** *schema-name* **APPEARS MORE THAN ONCE IN THE CURRENT PATH**

Explanation: The current path includes *schema name* more than once.

System Action: The statement is executed.

SQLSTATE: 01625

+599 **COMPARISON FUNCTIONS ARE NOT CREATED FOR A DISTINCT TYPE BASED ON A LONG STRING DATA TYPE**

Explanation: Comparison functions are not created for a distinct type based on a long string data type (BLOB, CLOB, DBCLOB) since the corresponding function are not available for these built-in data types.

System Action: The statement is processed successfully.

Programmer Response: No action is required.

SQLSTATE: 01596

+610 **A CREATE/ALTER ON OBJECT** *object-name* **HAS PLACED OBJECT IN** *utility* **PENDING**

Explanation: The identified object is in one of the following states:

- REBUILD pending for an index

  The index is in REBUILD PENDING status because CREATE INDEX with DEFER was specified on a populated table. The index is not generally available until the index is removed from the REBUILD pending state.
- REORG pending for a table space partition

  The table space is in REORG pending because ALTER INDEX was used to change the limit key values. The table space partition is not generally available until the REORG pending status is removed.
- REORG pending for a table space

  The table space is in REORG pending because ALTER TABLE was used to add an identity column to a populated table. The table space is not generally available until the REORG pending status is removed.

System Action: The object was placed in the indicated pending status.

Programmer Response: The following actions may be taken:

- For REBUILD pending on an index, use the REBUILD INDEX utility to rebuild the index and remove the REBUILD pending status.
- For REORG pending on a table space partition, perform the following steps:
  1. Issue a DISPLAY DATABASE command for the table space to identify which partitions are in REORG pending.
  2. Run the REORG utility on the partitions that are in REORG pending.
- For REORG pending on a table space, run the REORG utility on the table space.

SQLSTATE: 01566

+645 **WHERE NOT NULL IS IGNORED BECAUSE THE INDEX KEY CANNOT CONTAIN NULL VALUES**

Explanation: The WHERE NOT NULL clause is ignored on the CREATE INDEX statement because the index key is defined on columns that cannot contain null values.

System Action: The option is ignored; processing continues.

Programmer Response: Remove the WHERE NOT NULL clause to get rid of the warning.

SQLSTATE: 01528

+650 **THE TABLE BEING CREATED OR ALTERED CANNOT BECOME A DEPENDENT TABLE**

Explanation: This table is defined with the maximum number of columns. It cannot be altered to be a dependent table later.

System Action: The table is created. Check to see if the table will become a dependent table at a later time. If yes, drop and recreate the table with fewer than 750 columns.

SQLSTATE: 01538

+653 **TABLE** *table-name* **IN PARTITIONED TABLESPACE** *tspace-name* **IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT BEEN CREATED**

Explanation: An attempt has been made to insert or manipulate data in or create a view on a partitioned table (that is, a table residing in a partitioned table space) before the partitioned index for that table has been created.

A table residing in a partitioned table space cannot be referenced in any SQL manipulative statement or a CREATE VIEW statement before the partitioned index for that table has been created.

System Action: A valid plan or package will be created if no errors are detected. The statement is bound dynamically on each execution of the statement.

Programmer Response: For better performance, rebind the plan or package after correcting the statement. To correct the statement, verify that the correct table was specified in the statement. If so, ensure that the partitioned index for the table has been created successfully before attempting to execute any SQL manipulative statements that reference that table.

SQLSTATE: 01551

+655 **STOGROUP stogroup_name HAS BOTH SPECIFIC AND NON-SPECIFIC VOLUME IDS. IT WILL NOT BE ALLOWED IN FUTURE RELEASES**

Explanation: The CREATE/ALTER STOGROUP statement has caused the STOGROUP with 'stogroup_name' to have both specific and non-specific ('*') volume Ids. This warning code is used to specify that the mixing of different types of volume Ids will not be allowed in future releases.

System Action: DB2 continues processing.

Programmer Response: Plan to use either specific or non-specific volume ids to avoid future release migration impact. ALTER STOGROUP may be used to drop all specific volume ids or all non-specific volume ids.

SQLSTATE: 01597

+658 **THE SUBPAGES VALUE IS IGNORED FOR THE CATALOG INDEX** *index-name*

Explanation: Only SUBPAGES 1 is allowed for this catalog index.

System Action: The index was altered successfully using SUBPAGES 1. If you are also altering the TYPE option to a new value, the index is placed in recovery pending status.

SQLSTATE: 01600

+664 **THE INTERNAL LENGTH OF THE LIMIT-KEY FIELDS FOR THE PARTITIONED INDEX** *index-name* **EXCEEDS THE LENGTH IMPOSED BY DB2**

Explanation: The ALTER INDEX statement can change a limit key for the partitioned index (that is, the cluster index for a table residing in a partitioned table space), and the length of the limit key exceeds the permitted maximum.

DB2 restricts the internal length of the limit keys for a partitioned index to a maximum of 40 bytes. The sum of the internal lengths of the limit keys specified in the PART clause of the ALTER INDEX statement exceeds that 40-byte maximum. The limit key was truncated to 40 bytes.

System Action: The specified index was altered but the limit key was truncated to 40 bytes.

SQLSTATE: 01540

+738 **DEFINITION CHANGE OF** *object object_name* **MAY REQUIRE SIMILAR CHANGE ON READ-ONLY SYSTEMS**

Explanation: A change was made to the definition of the specified object that may also require a similar change to any read-only shared systems.

System Action: The statement is successfully executed.

Programmer Response: Check the read-only shared systems that have the specified object defined, and determine if a change must be made to the object on those systems.

SQLSTATE: 01530

+799 **A SET STATEMENT REFERENCES A SPECIAL REGISTER THAT DOES NOT EXIST AT THE SERVER SITE**

Explanation: A DB2 server received a SET statement that it does not understand.

System Action: The SET SPECIAL REGISTER request is ignored.

Programmer Response: This SQLCODE can be returned to an application for any SQL statement. This SQLCODE may be masked by other negative SQLCODEs that the SQL statement receives. Processing continues at the server.

SQLSTATE: 01527

+802 **EXCEPTION ERROR** *exception-type* **HAS OCCURRED DURING** *operation-type* **OPERATION ON** *data-type* **DATA, POSITION** *position-number*

Explanation: The exception error *exception-type* occurred while doing an ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION, NEGATION, or BUILT-IN FUNCTION operation on a field whose *data-type* is DECIMAL, FLOAT, SMALLINT, or INTEGER. The error occurred while processing an arithmetic expression in the SELECT list of an outer SELECT statement, and the position in the select list is denoted by *position-number*. The possible exception types are FIXED POINT OVERFLOW, DECIMAL OVERFLOW, DIVIDE EXCEPTION, EXPONENT OVERFLOW, ZERO DIVIDE, or OUT OF RANGE. The data type displayed in the message may indicate the data type of the temporary internal copy of the data, which may differ from the actual column or literal data type due to conversions by DB2.

A fixed point overflow can occur during any arithmetic operation on either INTEGER or SMALLINT fields.

A decimal overflow exception can occur when one or more nonzero digits are lost because the destination field in any decimal operation is too short to contain the result.

A divide exception can occur on a decimal division operation when the quotient exceeds the specified data-field size. A zero divide exception occurs on any division by zero.

An exponent overflow can occur when the result characteristic of any floating-point operation exceeds 127 and the result fraction is not zero, i.e. the magnitude of the result exceeds approximately 7.2E+75.

Any of the exceptions/overflows can occur during the processing of a built-in function. If the *operation-type* is FUNCTION then the error occurred while processing either an input, intermediate, or final value. The cause could be that the value of a parameter is *out of range*.

Note: Parts of *exception-type*, *data-type*, *operation-type* , and *position-number* may or may not be returned in SQLCA, depending upon when the error was detected.

System Action: For each expression in error the indicator variable is set to negative two (-2) to indicate a null value returned. The data variable is unchanged. Execution of the statement continues with all nonerror columns and expressions of the outer SELECT list being returned. If the statement is cursor controlled then the CURSOR will remain open.

Programmer Response: Examine the expression for which the warning occurred to see if the cause (or the likely cause) of the problem can be determined. The problem may be data-dependent, in which case it will be necessary to examine the data that was being processed at the time the error occurred.

See the explanation of SQLCODE -405 for allowed ranges of numeric data types.

SQLSTATE: 01519

+806 **BIND ISOLATION LEVEL RR CONFLICTS WITH TABLESPACE LOCKSIZE PAGE OR LOCKSIZE ROW AND LOCKMAX 0**

Explanation: The specification of isolation level RR is incompatible with the LOCKSIZE PAGE or LOCKSIZE ROW and LOCKMAX 0 specification for a table space accessed by the application. Table space locking is used to protect the integrity of the application.

System Action: A valid package/plan is created if no errors are detected. Table space locking is used. RR isolation level is preserved.

Programmer Response: If you do not want table space locking, use isolation level UR, CS, or RS.

SQLSTATE: 01553

+807 **THE RESULT OF DECIMAL MULTIPLICATION MAY CAUSE OVERFLOW**

Explanation: An arithmetic expression contains a decimal multiplication that may cause an overflow condition when the statement is executed. The problem may be corrected by restructuring the arithmetic expression so that decimal multiplication precedes decimal division or by changing the precision and scale of the operands in the arithmetic expression. Refer to Chapter 3 of *DB2 SQL Reference* for the precision and scale of the decimal multiplication and division results.

System Action: A valid package will be created if no errors are detected.

SQLSTATE: 01554

+863 **THE CONNECTION WAS SUCCESSFUL BUT ONLY SBCS WILL BE SUPPORTED**

Explanation: The target AS supports only the DB2 SBCS CCSID. The DB2 Mixed CCSID or GRAPHIC CCSID or both are not supported by the target AS. DB2 character data sent to the target AS must be restricted to SBCS.

System Action: The CONNECT statement is successful. The release level of the target AS has been placed into the SQLERRP field of the SQLCA (see *DB2 SQL Reference* for the CONNECT statement).

Programmer Response: Do not execute any SQL statements which pass either mixed data or graphic data as input host variables.

SQLSTATE: 01539

+883 ROLLBACK TO SAVEPOINT OCCURED WHEN THERE WERE OPERATIONS THAT CANNOT BE UNDONE, OR AN OPERATION THAT CANNOT BE UNDONE OCCURRED WHEN THERE WAS A SAVEPOINT OUTSTANDING

Explanation: The operations that are referred to are updates (inserts into or deletes from) a created global temporary table. If this SQL warning code is received as the result of a ROLLBACK TO SAVEPOINT statement, the rollback is performed; however, the changes that were made to the temporary table are not undone. If this SQL warning code is received as the result of an operation to a created global temporary table, the operation is performed; however, be advised that a savepoint is outstanding, and the update will not be backed out if a rollback to the savepoint is performed.

System Action: The SQL statement is processed.

Programmer Response: Verify that this is what you meant.

SQLSTATE: 01640

+2000 TYPE 1 INDEXES WITH SUBPAGES GREATER THAN 1 CANNOT BECOME GROUP BUFFER POOL DEPENDENT IN A DATA SHARING ENVIRONMENT

Explanation: A SUBPAGES value of greater than 1 was specified on a CREATE INDEX or ALTER INDEX statement in a data sharing environment. Type 1 indexes with a SUBPAGES value greater than 1 can only be accessed in a data sharing environment in non-group buffer pool dependent mode. Any read or update request that would cause the index to become group buffer pool dependent will be denied with a 'resource unavailable' condition. Only type 2 indexes or type 1 indexes with SUBPAGES 1 can become group buffer pool dependent.

System Action: The statement is successfully executed. If ALTER INDEX was performed, then the index is placed in recovery pending status.

SQLSTATE: 01638

+20002 THE GBPCACHE SPECIFICATION IS IGNORED, *bpname* DOES NOT ALLOW CACHING

Explanation: This message is issued in response to a CREATE or ALTER of a table space or index is which a buffer pool is named that corresponds to a group buffer pool that is defined with GBPCACHE NO. The corresponding group buffer pool is used only for cross-invalidation. It contains no data entries. All reads and writes are from and to DASD.

System Action: The statement is processed.

User Response: If you want to use one of the GBPCACHE options other than NONE, you must alter the table space or index to use a group buffer pool that is defined with GBPCACHE YES.

SQLSTATE: 01624

+20007 USE OF OPTIMIZATION HINTS IS DISALLOWED BY A DB2 SUBSYSTEM PARAMETER. THE SPECIAL REGISTER 'OPTIMIZATION HINT' IS SET TO THE DEFAULT VALUE OF BLANKS.

Explanation: DB2 is not enabled to use optimization hints. The special register OPTIMIZATION HINT is set to the default value of BLANKS.

System Action: Processing continues normally using the default OPTIMIZATION HINT value. The user-specified optimization hints are ignored. The access path is determined without the use of hints and processing continues normally.

Programmer Response: Enable the use of OPTIMIZATION HINT by changing the value of OPTIMIZATION HINTS on the DB2 Installation panel DSNTIP4.

If, after further consideration, you do not want to use an OPTIMIZATION HINT, change or remove the SET CURRENT OPTIMIZATION HINT statement.

SQLSTATE: 01602

+20122 DEFINE NO OPTION IS NOT APPLICABLE IN THE CONTEXT SPECIFIED

Explanation: The DEFINE NO option was specified, however it is not applicable in the context specified. DEFINE NO was specified in one of the following situations:

- a CREATE INDEX statement that included the VCAT clause
- a CREATE INDEX statement for a non-empty table
- a CREATE LOB TABLESPACE statement
- a CREATE TABLESPACE statement that included the VCAT clause

System Action: DB2 ignored the DEFINE NO option and created the object with the DEFINE YES option instead.

SQLSTATE: 01644

+30100 **OPERATION COMPLETED SUCCESSFULLY BUT A DISTRIBUTION PROTOCOL VIOLATION HAS BEEN DETECTED. ORIGINAL SQLCODE=***original-sqlcode* **AND ORIGINAL SQLSTATE=***original-sqlstate*

Explanation: The application requested operation (either COMMIT or ROLLBACK) has completed successfully but the response from the remote server and the SQLCODE that was returned from the remote server are inconsistent. For example, the reply message from the remote server indicates that a COMMIT operation completed successfully but the SQLCODE returned from the AS was negative.

System Action: An alert was generated. A DSNL031I message may have been written to the console. Refer to the description of this message for further information.

The SQLCODE returned by the remote server is replaced with +30100 and the SQLSTATE returned by the remote server is replaced with 01558.

The SQLCODE and SQLSTATE values that were returned from the AS are stored in the SQLERRM field in a string of the following format:

```
'original-sqlcode 'FF'X original-sqlstate'
```

Programmer Response: Notify the System Programmer for assistance in analyzing the trace data that was generated.

SQLSTATE: 01558

Error SQL codes

-007 **STATEMENT CONTAINS THE ILLEGAL CHARACTER** *character*

Explanation: The specified 'character' is not a valid character in SQL statements.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax and resubmit the statement. Refer to Chapter 3 of *DB2 SQL Reference* for information about the valid SQL character set.

SQLSTATE: 42601

-010 **THE STRING CONSTANT BEGINNING** *string* **IS NOT TERMINATED**

Explanation: The statement contains a string constant, beginning with 'string', that is not terminated properly.

System Action: The statement cannot be executed.

Programmer Response: Examine the statement for missing quotation marks or apostrophes in the indicated string constant.

SQLSTATE: 42603

-029 **INTO CLAUSE REQUIRED**

Explanation: SELECT statements embedded in an application program must have an INTO clause to denote where the results of the SELECT are to be placed. Dynamic SELECT statements do not permit the INTO clause.

System Action: The statement cannot be executed.

Programmer Response: Add the INTO clause to the SELECT statement and precompile the application program again.

SQLSTATE: 42601

-060 **INVALID** *type* **SPECIFICATION :** *spec*

Explanation: 'type' is either LENGTH or SCALE. 'spec' is the specified length or scale. Length or scale must be specified by an unsigned integer constant and the value must be in the range allowed by the data type.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement. Refer to Chapter 3 of *DB2 SQL Reference* for rules for length and scale.

SQLSTATE: 42815

-079 **QUALIFIER FOR DECLARED GLOBAL TEMPORARY TABLE** *table-name* **MUST BE SESSION, NOT** *qualifier*

Explanation: The qualifier for a declared temporary table must be SESSION. The DECLARE GLOBAL TEMPORARY TABLE statement defines a new temporary table named *table-name* with an explicit qualifier of *qualifier*. Specifying a qualifier other than SESSION is not allowed.

System Action: The statement was not executed.

Programmer Response: Change the statement in one of the following ways:

- Change the qualifier to SESSION.
- Remove the qualifier, and let DB2 default it to SESSION.

SQLSTATE: 428EK

-084 **UNACCEPTABLE SQL STATEMENT**

Explanation: This SQL statement is unacceptable to DB2. One of the following has occurred:

- An attempt has been made to PREPARE or EXECUTE IMMEDIATE an SQL statement that

cannot be prepared; refer to the proper SQL statement in *DB2 SQL Reference*

- The embedded SQL statement is not an SQL statement supported by DB2.
- The statement referenced an undeclared cursor.
- An attempt was made to prepare an ALLOCATE CURSOR statement but the statement identifier is already associated with a declared cursor.

System Action: The statement cannot be executed.

Programmer Response: If the situation involves an SQL statement that cannot be prepared, the problem is in the source of the SQL statement, not the application program. Thus, no action is necessary unless the source of the SQL statement is the application program itself.

If the situation involves an SQL statement that is not supported by DB2, remove it from the application program and precompile again.

If the situation involves an invalid PREPARE of an ALLOCATE CURSOR statement, change the application program to use a statement identifier that is not associated with a declared cursor.

SQLSTATE: 42612

-097 THE USE OF LONG VARCHAR OR LONG VARGRAPHIC IS NOT ALLOWED IN THIS CONTEXT

Explanation: The statement attempted to use the LONG VARCHAR or LONG VARGRAPHIC syntax. This syntax cannot be used for the following statements:

- CAST syntax
- CREATE DISTINCT TYPE
- CREATE FUNCTION
- CREATE PROCEDURE
- ALTER FUNCTION
- COMMENT ON FUNCTION
- GRANT EXECUTE ON FUNCTION
- REVOKE EXECUTE ON FUNCTION
- DROP

Use the VARCHAR or VARGRAPHIC syntax specifying an explicit length as required.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement.

SQLSTATE: 42601

-101 THE STATEMENT IS TOO LONG OR TOO COMPLEX

Explanation: DB2 cannot process the statement because it exceeds the system limits for length or complexity. Enabling parallelism will increase the complexity of the statement.

System Action: DB2 cannot process the statement.

Programmer Response: Divide the statement into shorter or less complex SQL statements.

If the statement enables parallelism, try disabling parallelism. You may do this by using the DEGREE(1) bind option for static SQL, or by setting the CURRENT DEGREE special register to '1' for dynamic SQL.

SQLSTATE: 54001

-102 LITERAL STRING IS TOO LONG. STRING BEGINS *string*

Explanation: The string constant beginning with 'string' has a length greater than 255 characters or 124 graphic characters. Character strings with lengths greater than 255 and graphic strings with lengths greater than 124 can be specified only through assignment from host variables.

For special registers, the allowable length depends on the particular special register. See Chapter 3 of the SQL Reference to determine the maximum length of a value for a special register.

Two consecutive string delimiters are used to represent one string delimiter within the character string, but these count as 2 bytes when calculating the lengths of character string constants.

System Action: The statement cannot be executed.

Programmer Response: The requested function is not available interactively. If the error occurred in the context of an SQL statement embedded in an application program, the desired result can be achieved by assigning the long string to a host variable, and substituting that variable for the string literal in the SQL statement.

SQLSTATE: 54002

-103 *literal* **IS AN INVALID NUMERIC LITERAL**

Explanation: The indicated 'literal' begins with a digit, but is not a valid integer, decimal, or float literal.

System Action: The statement cannot be executed.

Programmer Response: Correct the invalid literal.

SQLSTATE: 42604

-104 ILLEGAL SYMBOL *"token"***. SOME SYMBOLS THAT MIGHT BE LEGAL ARE:** *token-list*

Explanation: A syntax error was detected where the symbol *"token"* occurs in the SQL statement. The list of symbols that might be legal shows some alternate symbols that could possibly be correct at that point, if the preceding part of the statement is entirely correct.

However, the preceding part of the statement might be incorrect. For example, if an important keyword is

omitted, DB2 detects the error later, and not always immediately after the point where the keyword should appear. The list of alternate symbols are only suggestions. Some of those symbols might not even be legal for statements to be executed by DB2. Those symbols are possibly correct for statements sent to other database management systems.

This SQL code will also be issued if the RELEASE TO SAVEPOINT statement is specified without a savepoint name.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement and execute it again.

SQLSTATE: 42601

-105 INVALID STRING

Explanation: The statement contains an invalid string. It is neither a character string nor a graphic string.

System Action: The statement cannot be executed.

Programmer Response: Specify the correct format of the string. Check for a graphic string, paired delimiters, the character G or N, and an even number of bytes within the string.

SQLSTATE: 42604

-107 THE NAME *name* IS TOO LONG. MAXIMUM ALLOWABLE SIZE IS *size*

Explanation: The *name* is too long. The maximum permissible length for names of that type is indicated by *size*.

Names for the following cannot contain more than 128 characters:

- Savepoint-name

Names for the following cannot contain more than 64 characters:

- Version-id

Names for the following cannot contain more than 18 characters (20 including SQL escape characters, if present):

- SQL columns
- SQL tables
- SQL views
- SQL indexes
- SQL aliases
- SQL synonyms
- Collection-ids
- Check constraints
- Functions
- Stored procedures
- User-defined types

Names for the following cannot contain more than 16 characters:

- Location-name

The following cannot contain more than 8 characters:

- Table qualifiers
- View qualifiers
- Library member names specified in an INCLUDE statement
- Storage group names
- Database names
- Table space names
- Application plans
- Database request modules (DBRMs)
- Referential constraint names specified in CREATE or ALTER TABLE statements
- Package-ids
- Schema names
- Trigger names

Host variable names cannot contain more than 64 characters. Volume serial numbers cannot contain more than 6 characters. Labels cannot contain more than 30 characters.

System Action: The statement cannot be processed.

Programmer Response: Choose a shorter name for the object.

SQLSTATE: 42622

-108 THE NAME *name* IS QUALIFIED INCORRECTLY

Explanation: The name *name* is improperly qualified.

A target name on the RENAME statement may not have a qualifier.

System Action: The statement cannot be executed.

Programmer Response: Remove the qualifier and reissue the statement.

SQLSTATE: 42601

-109 *clause* CLAUSE IS NOT PERMITTED

Explanation: The indicated clause is not permitted in the context in which it appears in this SQL statement for the following reasons:

- A subselect cannot have an INTO clause.
- A CREATE VIEW statement cannot have INTO, ORDER BY, or FOR UPDATE clauses.
- An embedded SELECT statement cannot have ORDER BY or FOR UPDATE clauses
- SELECT statements used in cursor declarations cannot have an INTO clause.
- A RAISE_ERROR function can only be used as a select list item if it is cast to some data type using the CAST specification.

- DESCRIBE INPUT statement can not have USING clause.
- QUERYNO cannot be specified as part of an EXPLAIN statement when the EXPLAIN statement contains an 'explainable-sql-statement'.
- The table being updated in a POSITIONED UPDATE statement cannot be assigned a correlation name.

If the clause is part of a CREATE INDEX, CREATE TABLE, CREATE TABLESPACE or ALTER TABLESPACE statement, see the appropriate section of the SQL Reference for a description of the valid use of clauses for the statement.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement.

SQLSTATE: 42601

-110 **INVALID HEXADECIMAL LITERAL BEGINNING** *string*

Explanation: The literal beginning with the specified 'string' contains one or more characters that are not valid hexadecimal digits.

System Action: The statement cannot be executed.

Programmer Response: Correct the invalid literal.

SQLSTATE: 42606

-111 **A COLUMN FUNCTION DOES NOT INCLUDE A COLUMN NAME**

Explanation: The specification of a column function (AVG, MAX, MIN, or SUM) was invalid because such functions must include a column name in the operand. In a trigger definition, a transition variable specification does not qualify as a column name for this purpose.

System Action: The statement cannot be executed.

Programmer Response: A column name must be specified as an operand to the function. Refer to Chapter 4 of *DB2 SQL Reference* for information about the proper usage of column functions.

SQLSTATE: 42901

-112 **THE OPERAND OF A COLUMN FUNCTION IS ANOTHER COLUMN FUNCTION**

Explanation: The operand of a column function can be either an expression or DISTINCT followed by an expression. The operand cannot be another column function.

System Action: The statement cannot be executed.

Programmer Response: Correct the function specification. Refer to Chapter 4 of *DB2 SQL Reference* for information about the proper usage of column functions.

SQLSTATE: 42607

-113 **INVALID CHARACTER FOUND IN** *string***, REASON CODE** *nnn*

Explanation: The *string* contains an invalid character. It can be an SQL ordinary identifier name, a host variable name, or a DBCS comment.

For SBCS SQL ordinary identifiers, names of buffer pools, databases, plans, and storage groups must contain only uppercase alphabetic or national characters and numerics when CHARSET is KATAKANA; the first character must be alphabetic or national.

The following reason codes apply to SBCS identifiers:

000 An invalid character was found in the SBCS identifier (including the case in which a DBCS identifier was used where only an SBCS identifier is allowed.)

The following reason codes apply to DBCS identifiers or comments:

101 An odd number of bytes exists between the shift-out and the shift-in character.

102 Either a shift-in or shift-out character is missing.

103 DBCS blanks X'4040' are not allowed.

104 There are no characters between the shift-out and the shift-in characters.

105 Shift-out cannot be the first byte of the DBCS character between the shift-out and the shift-in characters.

System Action: Processing is terminated.

User Response: Correct the name.

SQLSTATE: 42602

-114 **THE LOCATION NAME** *location* **DOES NOT MATCH THE CURRENT SERVER**

Explanation: A 3-part SQL procedure name was provided for one of the following SQL statements:

```
ASSOCIATE LOCATORS
CALL
DESCRIBE PROCEDURE
```

The first part of the SQL procedure name, which specifies the location where the stored procedure resides, did not match the value of the SQL CURRENT SERVER special register.

System Action: The statement cannot be executed.

Programmer Response: Take one of these actions to resolve the mismatch:

- Change the location qualifier to match the CURRENT SERVER special register.
- Issue an SQL CONNECT to the location where the stored procedure resides before issuing the SQL statement. Ensure that the SQL CALL statement is issued before the ASSOCIATE LOCATORS or DESCRIBE PROCEDURE.
- Bind the package containing the 3-part SQL procedure name with the BIND option DBPROTOCOL(DRDA). With this option, DB2 implicitly uses the DRDA protocol for remote access to the stored procedure.
- Correct the statements so that the exact syntax used to specify the procedure name on the CALL statement be the same as that on the ASSOCIATE LOCATOR and/or DESCRIBE PROCEDURE. If an unqualified name is used to CALL the procedure, the 1-part name must also be used on the other statements. If the CALL statement is made with a 3-part name, and the current server is the same as the location in the 3-part name, the ASSOCIATE LOCATOR or DESCRIBE procedure can omit the location.

SQLSTATE: 42961

-115 A PREDICATE IS INVALID BECAUSE THE COMPARISON OPERATOR *operator* IS FOLLOWED BY A PARENTHESIZED LIST OR BY ANY OR ALL WITHOUT A SUBQUERY

Explanation: A simple comparison like '>' must not be followed by a list of items. ANY and ALL comparisons must be followed by a subselect, rather than an expression or a list of items.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement. Refer to Chapter 6 of *DB2 SQL Reference* for information about the syntax of SQL statements.

SQLSTATE: 42601

-117 THE NUMBER OF VALUES ASSIGNED IS NOT THE SAME AS THE NUMBER OF SPECIFIED OR IMPLIED COLUMNS

Explanation: The number of insert values in the value list of the INSERT statement is not the same as the number of object columns specified. Alternatively, the number of values on the right side of an assignment in a SET assignment statement or the SET clause of an UPDATE statement does not match the number of columns on the left side.

System Action: The statement cannot be executed. No data was inserted into the object table.

Programmer Response: Correct the statement to specify one and only one value for each of the specified object columns.

SQLSTATE: 42802

-118 THE OBJECT TABLE OR VIEW OF THE DELETE OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE

Explanation: The table or view specified as the object of a DELETE or UPDATE statement also appears in the FROM clause of a subselect within the statement.

The table or view that is the object of a UPDATE or DELETE cannot also be used to supply the values to be inserted or to qualify the rows to be updated or deleted.

System Action: The statement cannot be executed. No data was updated or deleted.

Programmer Response: The implied function is not supported by DB2. It may be possible to obtain the desired result by creating a temporary copy of the object table or view and addressing the subselect to that copy. Refer to Chapter 6 of *DB2 SQL Reference* for information about the syntax of SQL statements.

SQLSTATE: 42902

-119 A COLUMN IDENTIFIED IN A HAVING CLAUSE IS NOT INCLUDED IN THE GROUP BY CLAUSE

Explanation: A column identified in a HAVING clause (possibly within a scalar function) does not appear in the GROUP BY clause. Columns specified in a HAVING clause must appear within column functions or also be specified in the GROUP BY clause.

System Action: The statement cannot be executed.

Programmer Response: The implied function is not supported by DB2. Refer to Chapter 5 of *DB2 SQL Reference* for information about the proper usage of HAVING and GROUP BY clauses.

SQLSTATE: 42803

-120 A WHERE CLAUSE, SET CLAUSE, VALUES CLAUSE, OR A SET HOST-VARIABLE STATEMENT INCLUDES A COLUMN FUNCTION

Explanation: A column function or a user-defined function that is sourced on a column function is not permitted in a SET clause, a VALUES clause, or a SET assignment statement. A column function or a user-defined function that is sourced on a column function is allowed in a WHERE clause only if the WHERE clause appears within a subquery of a HAVING clause.

System Action: The statement cannot be executed.

Note: The 'column-name' may or may not be returned in SQLCA, depending on the nature of the error occurring in the SQL statement.

Programmer Response: The implied function is not supported by DB2. Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on operands that can be specified within WHERE, SET and VALUES clauses and SET assignment statements.

SQLSTATE: 42903

-121 THE COLUMN *name* IS IDENTIFIED MORE THAN ONCE IN THE INSERT OR UPDATE OR SET TRANSITION VARIABLE STATEMENT

Explanation: The same column 'name' is specified more than once, either in the list of object columns of an INSERT statement, in the SET clause of an UPDATE statement, or in a SET transition variable statement.

System Action: The statement cannot be executed. No data was inserted or updated in the object table.

Programmer Response: Correct the syntax of the statement so that each column name is specified only once.

SQLSTATE: 42701

-122 A SELECT STATEMENT WITH NO GROUP BY CLAUSE CONTAINS A COLUMN NAME AND A COLUMN FUNCTION IN THE SELECT CLAUSE OR A COLUMN NAME IS CONTAINED IN THE SELECT CLAUSE BUT NOT IN THE GROUP BY CLAUSE

Explanation: The SELECT statement contains one of these errors:

- The statement contains a column name and a column function in the SELECT clause, but no GROUP BY clause.
- A column name is contained in the SELECT clause (possibly within a scalar function) but not in the GROUP BY clause.

  Note: A HAVING clause specified without a GROUP BY clause implies a GROUP BY with no columns. Thus, no column names are allowed in the SELECT clause.
- A *sort-key-expression* was specified in the ORDER BY clause, the result table contains grouped data, but the *select-clause* and ORDER BY clause contain a mixture of grouped data and non-grouped data.

System Action: The statement cannot be executed.

Programmer Response: You can correct the statement by:

- including the columns in the GROUP BY clause that are in the SELECT clause, or
- removing the columns from the SELECT clause.

Refer to Chapter 5 of *DB2 SQL Reference* for information about the use of GROUP BY clauses in SQL statements.

SQLSTATE: 42803

-123 THE PARAMETER IN POSITION *n* IN THE FUNCTION *name* MUST BE A CONSTANT OR KEYWORD

Explanation: The parameter in position *n* in the function *name* is not a constant when it is required to be a constant or a keyword when it is required to be a keyword.

System Action: The statement could not be processed.

Programmer Response: Action: Ensure that each argument of the function conforms to the definition of the corresponding parameter.

SQLSTATE: 42601

-125 AN INTEGER IN THE ORDER BY CLAUSE DOES NOT IDENTIFY A COLUMN OF THE RESULT

Explanation: The ORDER BY clause in the statement contains a column number that is either less than one, or greater than the number of columns of the result table (the number of items in the SELECT clause).

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the ORDER BY clause such that each column identifier properly denotes a column of the result table.

SQLSTATE: 42805

-126 THE SELECT STATEMENT CONTAINS BOTH AN UPDATE CLAUSE AND AN ORDER BY CLAUSE

Explanation: The SELECT statement in the declaration for a cursor contains both an UPDATE clause and an ORDER BY clause. Unless you use a scrollable cursor, an ORDER BY clause cannot be specified in the declaration for a cursor that is to be used for UPDATE.

System Action: The statement cannot be processed. The cursor remains undefined in the application program.

Programmer Response: The implied function is not supported by DB2. A cursor that is to be used for update cannot be defined to fetch the rows of the object table in a specific order.

Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on the declarations for cursors to be used for update.

SQLSTATE: 42829

-127 DISTINCT IS SPECIFIED MORE THAN ONCE IN A SUBSELECT

Explanation: The DISTINCT qualifier can be used only once in a SELECT statement or a subselect.

System Action: The statement cannot be executed.

Programmer Response: The implied function is not supported by DB2. Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on the use of the DISTINCT qualifier.

SQLSTATE: 42905

-128 INVALID USE OF NULL IN A PREDICATE

Explanation: The use of NULL in the search condition does not conform to the rules of SQL syntax.

System Action: The statement cannot be executed.

Programmer Response: The implied function is not supported by DB2. Refer to Chapter 3 of *DB2 SQL Reference* for information about the proper use of the NULL operand.

SQLSTATE: 42601

-129 THE STATEMENT CONTAINS TOO MANY TABLE NAMES

Explanation: A subselect (including all subqueries) can have a maximum of 225 references to table names.

System Action: The statement cannot be executed.

Programmer Response: Break the SQL statement into two or more simpler statements with less than 225 table references in each. The count will include the number of base table occurrences from each table or view on the FROM list. Refer to Chapter 5 of *DB2 SQL Reference* for the definition of a subselect.

SQLSTATE: 54004

-130 THE ESCAPE CLAUSE CONSISTS OF MORE THAN ONE CHARACTER, OR THE STRING PATTERN CONTAINS AN INVALID OCCURRENCE OF THE ESCAPE CHARACTER

Explanation: The ESCAPE character must be a single character, either SBCS or DBCS as appropriate. For 'column-name LIKE pattern', the ESCAPE character can only appear in the character string if it is followed by itself, %, or _ (underscore). The Escape Clause cannot be specified if the column name at the left of the LIKE or NOT LIKE has the MIXED subtype.

System Action: The statement cannot be executed.

Programmer Response: Correct the string pattern, or choose a different ESCAPE character and change the pattern accordingly, or eliminate the use of the Escape Clause on the LIKE or NOT LIKE predicate where the column name to the left has the MIXED subtype.

SQLSTATE: 22019 if other than invalid ESCAPE pattern. 22025 if invalid ESCAPE pattern.

-131 STATEMENT WITH LIKE PREDICATE HAS INCOMPATIBLE DATA TYPES

Explanation: If the column name at the left of LIKE or NOT LIKE is of type character, the expression at the right and the ESCAPE character must be of type character. If the column name is of type graphic, the expression at the right and the ESCAPE character must be of type graphic.

System Action: The statement cannot be executed.

Programmer Response: Check the data type of every operand.

SQLSTATE: 42818

-132 AN OPERAND OF *value* IS NOT VALID

Explanation: The operation *value* can be the LIKE predicate, the ESCAPE clause, the LOCATE scalar function, or the POSSTR scalar function.

The operand appearing to the left of a LIKE or NOT LIKE predicate, the operand appearing in the ESCAPE clause, the second parameter of LOCATE or the first operand of POSSTR must be a string expression. The value appearing to the right of the LIKE predicate, the first operand of LOCATE or the second operand of POSSTR can be one of:

- a constant
- a special register
- a host variable
- a scalar function whose operands are any of the above. However, nested function invocatoins may not be used
- an expression concatenating any of the above

The actual length of a *pattern-expression* or *search-expression* cannot be more than 4000 bytes. The actual length of an escape clause cannot exceed one character.

A LIKE predicate, ESCAPE clause, LOCATE scalar function or POSSTR scalar function cannot be used with DATE, TIME, or TIMESTAMP.

System Action: The statement cannot be processed.

Programmer Response: Check and correct the syntax of LIKE, LOCATE, or POSSTR

SQLSTATE: 42824

-133 **A COLUMN FUNCTION IN A SUBQUERY OF A HAVING CLAUSE IS INVALID BECAUSE ALL COLUMN REFERENCES IN ITS ARGUMENT ARE NOT CORRELATED TO THE GROUP BY RESULT THAT THE HAVING CLAUSE IS APPLIED TO**

Explanation: If a column function has a correlated column reference, it must be correlated from within a HAVING clause to the GROUP BY result that the HAVING clause is applied to. All column references in the argument must satisfy this condition.

System Action: The statement cannot be executed.

Programmer Response: Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on the syntax of the HAVING clause.

SQLSTATE: 42906

-134 **IMPROPER USE OF LONG STRING COLUMN** *column-name* **OR AN EXPRESSION THAT RESOLVES TO A LONG STRING**

Explanation: The SQL statement references a long string, but DB2 does not allow the use of long strings in the specified context. For an exhaustive list of such contexts, refer to "Varying Length Character Strings" in Chapter 3 of *DB2 SQL Reference*.

System Action: DB2 cannot process the statement.

Note: The *column-name* might not be returned in the SQLCA, depending on the nature of the error and the syntax in which it occurred.

Programmer Response: DB2 does not support the requested operation on a long string value. Refer to Chapter 3 of *DB2 SQL Reference* for information about restrictions on the specification and manipulation of long string values.

SQLSTATE: 42907

-136 **SORT CANNOT BE EXECUTED BECAUSE THE SORT KEY LENGTH IS GREATER THAN 4000 BYTES**

Explanation: A sort key is derived from the list of columns specified following a DISTINCT qualifier, or in an ORDER BY or GROUP BY clause. If both a DISTINCT qualifier and an ORDER BY or GROUP BY clause are present, the sort key is derived from the combination of both lists of columns.

The *internal* length of the sort key cannot exceed 4000 bytes. In attempting to process the SQL statement, the internal length of the sort key derived from the DISTINCT and/or ORDER BY or GROUP BY specifications was found to exceed that 4000-byte maximum.

System Action: The statement cannot be executed.

Programmer Response: The statement must be modified such that the internal length of the sort key will not exceed 4000 bytes. In general, this means that one or more column names must be deleted from the ORDER BY or GROUP BY clause, or the list following the DISTINCT qualifier.

SQLSTATE: 54005

-137 **THE LENGTH RESULTING FROM** *operation* **IS GREATER THAN** *maximum-length*

Explanation: The length of the result of concatenation or a function exceeds the defined maximum. The operation that resulted in the error is *operation*.

- For concatenation, the length cannot exceed 32,764 (if character operands) or 16,382 (if graphic operands).
- For other functions, see *DB2 SQL Reference* for the maximum result length.

System Action: The statement cannot be executed.

Programmer Response: Ensure that the length of the result does not exceed the defined maximum.

SQLSTATE: 54006

-138 **THE SECOND OR THIRD ARGUMENT OF THE SUBSTR FUNCTION IS OUT OF RANGE**

Explanation: One of the following conditions exists:

- The second argument of the SUBSTR function is less than 1 or greater than M.
- The third argument of the SUBSTR function is an integer constant 0 or an expression whose value is less than 0 or greater than M−N+1.

M is the length of the first argument, if it is of fixed-length, or M is the maximum length of the first argument, if it is of varying-length. N is the value of the second argument.

System Action: The statement cannot be executed.

Programmer Response: Ensure that the second and third arguments of the SUBSTR function have legal values according the above rules.

SQLSTATE: 22011

-142 **THE SQL STATEMENT IS NOT SUPPORTED**

Explanation: An SQL statement was detected that is not supported by the database. The statement might be valid for other IBM relational database products or it might be valid in another context. For example, statements such as VALUES and SIGNAL SQLSTATE can appear only inside a trigger.

System Action: The statement cannot be executed.

Programmer Response: Change the syntax of the SQL statement or remove the statement from the program.

SQLSTATE: 42612

-144 INVALID SECTION NUMBER *number*

Explanation: One of the following:

1. The user attempted to execute an invalid section.
2. This release of DB2 does not support the SQL statement.
3. The section number in the call parameter list is one of these:
   - Negative
   - An invalid duplicate
   - Greater than the maximum section number of the DBRM or package.

System Action: The statement is not executed.

Programmer Response: For case 1: If you are executing a package that was bound with SQLERROR(CONTINUE), determine whether the statement in question was bound as a valid section. You can use the following statements to query the DB2 catalog:

```
  SELECT SQLERROR
FROM   SYSIBM.SYSPACKAGE
WHERE  COLLID    = collection-id AND
NAME      = package-id    AND
VERSION   = version-name;
```

If that query returns 'C', the package was bound with SQLERROR(CONTINUE).

```
  SELECT STMTNO, TEXT
FROM   SYSIBM.SYSPACKSTMT
WHERE  COLLID    = collection-id AND
NAME      = package-id    AND
VERSION   = version-name  AND
SECTNO    = number        AND
BINDERROR = 'Y';
```

If that query returns any rows, the section is invalid. Refer to the error messages issued during the bind to determine the cause. Correct any errors and bind the package again, using the REPLACE option.

For case 2: If the DB2 system has fallen back to a previous release, determine whether there are any SQL statements with a section number of zero that are not supported by that release. You can use the following statements to query the DB2 catalog.

When executing from a DBRM, use:

```
  SELECT *
FROM   SYSIBM.SYSSTMT
WHERE  SECTNO = 0
ORDER BY NAME, PLNAME, STMTNO, SEQNO;
```

When executing from a package, use:

```
  SELECT *
FROM   SYSIBM.SYSPACKSTMT
WHERE  SECTNO = 0
ORDER BY COLLID, NAME, VERSION, STMTNO, SEQNO;
```

For case 3: Examine the application to determine whether the call parameter list was changed in some way. In general, you should not attempt to change the output of the precompiler.

SQLSTATE: 58003

-147 ALTER FUNCTION *function-name* **FAILED BECAUSE SOURCE FUNCTIONS CANNOT BE ALTERED**

Explanation: The function cannot be altered because it is a source function. Only external scalar functions, or external table functions can be altered.

To change an existing source function, you must DROP the function and recreate it.

System Action: The statement cannot be executed.

Programmer Response: Change the statement to refer to a function that can be altered, or recreate the function by dropping it and then creating a new version of it.

SQLSTATE: 42809

-148 THE SOURCE TABLE *source-name* **CANNOT BE RENAMED OR ALTERED**

Explanation: Possible cases:

1 The RENAME statement cannot be used to rename a view, an active RLST table, or a table for which a synonym is defined.

2 The ALTER statement cannot be used to alter the length of the column because the column is referenced in a referential integrity relation, a user exit (field procedure, edit procedure, valid procedure, stored procedure or user defined function), a global temporary table, or a table defined with data capture changes. If the table name specified in the alter is a view or if there exists a row in SYSVIEWDEP that has *source-name* as a base table name, then this ALTER statement will fail.

System Action: The statement cannot be executed.

Programmer Response: For case:

1 Drop all views, inactivate the RLST table, or drop the synonym.

2 Avoid referential integrity relations, user exits, or global temporary tables.

3 Run REORG INDEX, REORG TABLESPACE,

or REBUILD INDEX. If the index is partitioned, then run the utility on all the partitions. Reissue the statement.

SQLSTATE: 42809

-150 THE OBJECT OF THE INSERT, DELETE, OR UPDATE STATEMENT IS A VIEW OR TRANSITION TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED

Explanation: One of the following occurred:

- A transition table was named in an INSERT, UPDATE, or DELETE statement in a triggered action. Transition tables are read-only.
- The view named in the INSERT, UPDATE, or DELETE statement is defined in such a way that the requested insert, update, or delete operation cannot be performed upon it.

  Inserts into a view are prohibited if:
  - The view definition contains a join, a GROUP BY, or a HAVING clause.
  - The SELECT clause in the view definition contains the DISTINCT qualifier, an arithmetic expression, a string expression, a built-in function, or a constant.
  - Two or more columns of the view are derived from the same column.
  - A base table of the view contains a column that does not have a default value and is not included in the view.

  Updates to a view are prohibited if:
  - The view definition contains a join, a GROUP BY, or a HAVING clause.
  - The SELECT clause in the view definition contains the DISTINCT qualifier or a function.

  Also, a given column in a view cannot be updated (that is, the values in that column cannot be updated) if the column is derived from an arithmetic expression, a constant, a column that is part of the key of a partitioned index, or a column of a catalog table that cannot be updated.

  Deletes against a view are prohibited if:
  - The view definition contains a join, a GROUP BY, or a HAVING clause.
  - The SELECT clause in the view definition contains the DISTINCT qualifier or a built-in function.

System Action: The statement cannot be executed. No data was inserted, updated, or deleted.

Programmer Response: The requested function cannot be performed on the view. Refer to Chapter 6 of *DB2 SQL Reference* for further information regarding inserting, deleting, and updating views.

If the error occurred on a CREATE TRIGGER statement, remove the INSERT, UPDATE, or DELETE reference to the transition table.

SQLSTATE: 42807

-151 THE UPDATE STATEMENT IS INVALID BECAUSE THE CATALOG DESCRIPTION OF COLUMN *column-name* INDICATES THAT IT CANNOT BE UPDATED

Explanation: The specified column cannot be updated for one of the following reasons:

- The values for columns occurring in the partitioning key of a partitioned table cannot be updated.
- The object table is a view and the specified column is defined (in the definition of the view) in such a way that it cannot be updated.
- The object table is a catalog table with no columns that can be updated.
- The object column is a ROWID column.
- The object column is defined with the AS IDENTITY and GENERATED ALWAYS attributes.
- The specified column of catalog tables cannot be updated because the column itself is not updatable.

Individual columns in a view cannot be updated for one of the following reasons:

- The column is derived from an SQL function, an arithmetic expression, or a constant.
- The column is defined for a column of an underlying view that cannot be updated.
- The column is defined for a read-only view.
- The column is defined for a column that is in the partitioning key of a partitioned table.

System Action: The statement cannot be executed. No data was updated in the object table or view.

Programmer Response: The requested function is not supported by DB2. Refer to the description of the UPDATE statement in Chapter 6 of *DB2 SQL Reference* for information about restrictions on the ability to update ROWID columns, identity columns, and columns in partitioned tables and views.

SQLSTATE: 42808

-152 THE DROP *clause* CLAUSE IN THE ALTER STATEMENT IS INVALID BECAUSE *constraint-name* IS A *constraint-type*

Explanation: The DROP *clause* of an ALTER TABLE statement tried to drop a constraint that does not match the *constraint-type* in the DROP clause. *clause* must identify an appropriate *constraint-type* as follows:

REFERENTIAL CONSTRAINT
The identified constraint must be a referential constraint.

CHECK CONSTRAINT
The identified constraint must be a check constraint.

PRIMARY KEY CONSTRAINT
The identified constraint must be a primary key constraint.

UNIQUE KEY CONSTRAINT
The identified constraint must be a unique key constraint.

System Action: The ALTER TABLE DROP statement cannot be executed. No object was dropped.

Programmer Response: Drop the existing object with the correct DROP clause of the ALTER TABLE statement.

SQLSTATE: 42809

-153 THE STATEMENT IS INVALID BECAUSE THE VIEW OR TABLE DEFINITION DOES NOT INCLUDE A UNIQUE NAME FOR EACH COLUMN

Explanation: You must specify a list of column names if the result table of the subselect that is specified in the CREATE VIEW or DECLARE GLOBAL TEMPORARY TABLE statement has duplicate column names or an unnamed column (a column from a constant, function, or expression).

System Action: The statement cannot be executed. The specified view was not created, or the declared temporary table was not declared.

Programmer Response: Correct the statement by providing a list of names for the columns of the view. Refer to Chapter 6 of *DB2 SQL Reference* for information about the syntax of the CREATE VIEW statement or the DECLARE GLOBAL TEMPORARY TABLE statement.

SQLSTATE: 42908

-154 THE STATEMENT FAILED BECAUSE VIEW OR TABLE DEFINITION IS NOT VALID

Explanation: The view defined in the CREATE VIEW statement or the table declared in the DECLARE GLOBAL TEMPORARY TABLE statement is not valid because the view or table definition contains one of the following:

- UNION or UNION ALL
- a reference to a remote object

System Action: The statement cannot be executed. The specified object is not defined.

Programmer Response: Refer to Chapter 6 of *DB2 SQL Reference* for information about restrictions on the definitions for views or declared temporary tables.

SQLSTATE: 42909

-156 THE STATEMENT DOES NOT IDENTIFY A TABLE

Explanation: The statements ALTER TABLE, DROP TABLE, LOCK TABLE, CREATE INDEX, and CREATE TRIGGER apply only to tables. Indexes and triggers can be defined only on tables.

System Action: The statement cannot be executed. The specified view or remote object was not altered, dropped, or locked, or the index or trigger was not created.

Programmer Response: Verify that the proper name was specified in the statement.

SQLSTATE: 42809

-157 ONLY A TABLE NAME CAN BE SPECIFIED IN A FOREIGN KEY CLAUSE. *object-name* **IS NOT THE NAME OF A TABLE.**

Explanation: The indicated object was identified in a FOREIGN KEY clause of a CREATE or ALTER TABLE statement. A FOREIGN KEY clause must identify a table.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement to specify a table name in the foreign key clause.

SQLSTATE: 42810

-158 THE NUMBER OF COLUMNS SPECIFIED FOR THE VIEW IS NOT THE SAME AS THE NUMBER OF COLUMNS SPECIFIED BY THE SELECT CLAUSE, OR THE NUMBER OF COLUMNS SPECIFIED IN THE CORRELATION CLAUSE IN A FROM CLAUSE IS NOT THE SAME AS THE NUMBER OF COLUMNS IN THE CORRESPONDING TABLE, VIEW, TABLE EXPRESSION, OR TABLE FUNCTION

Explanation: There are two cases:

- The number of column names specified for a view in a CREATE VIEW statement must equal the number of elements (column names, SQL functions, expressions, etc.) specified in the following AS SELECT clause.
- The number of column names specified in a correlation clause must equal the number of columns in the corresponding table, view, table expression or table function.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the statement to specify a column name for each column in the corresponding object (table, view, etc.). Refer to Chapter 6 of *DB2 SQL Reference* for information about the syntax of the statement.

SQLSTATE: 42811

-159 **DROP OR COMMENT ON** *object* **IDENTIFIES A(N)** *object-type1* **RATHER THAN A(N)** *object-type2*

Explanation: The object specified in the DROP VIEW statement, DROP ALIAS statement, or COMMENT ON ALIAS statement identifies a table instead of a view or an alias.

The DROP VIEW statement can have only a view as its object. The DROP ALIAS or COMMENT ON ALIAS statement can have only an alias as its object. You must use the DROP TABLE statement to drop a table that is neither a view nor an alias. You must use the COMMENT ON TABLE statement to comment on a table or view.

System Action: The statement cannot be executed.

Programmer Response: Correct the DROP VIEW, DROP ALIAS, or COMMENT ON ALIAS statement so that the view name or the alias name is specified correctly (with the proper qualifier). If you intended to drop or comment on the specified table, use the DROP TABLE or COMMENT ON TABLE statement.

SQLSTATE: 42809

-160 **THE WITH CHECK OPTION CANNOT BE USED FOR THE SPECIFIED VIEW**

Explanation: The WITH CHECK OPTION does not apply to a view definition under either of the following circumstances:

- The view is read-only (for example, the view definition includes DISTINCT GROUP BY, or JOIN).
- The view definition includes a subquery.

System Action: The statement cannot be executed. The specified view was not created.

Programmer Response: Refer to Chapter 6 of *DB2 SQL Reference* for rules regarding use of the WITH CHECK OPTION in view definitions.

SQLSTATE: 42813

-161 **THE INSERT OR UPDATE IS NOT ALLOWED BECAUSE A RESULTING ROW DOES NOT SATISFY THE VIEW DEFINITION**

Explanation: The WITH CHECK OPTION applies to the view that is the object of the INSERT or UPDATE statement. Consequently, all attempts to insert or update rows in that view are checked to ensure that the results will conform to the view definition.

System Action: The statement cannot be executed. No inserts or updates were performed, and the contents of the object view (and underlying base table) remain unchanged.

Programmer Response: Examine the view definition to determine why the requested INSERT or UPDATE was rejected. Note that this may be a data-dependent condition.

SQLSTATE: 44000

-164 *auth-id1* **DOES NOT HAVE THE PRIVILEGE TO CREATE A VIEW WITH QUALIFICATION** *authorization-ID*

Explanation: The authorization ID *auth-id1* does not have the authority necessary to create views with qualifiers other than its own authorization ID. Specifically, the attempt to create a view with qualifier *authorization-ID* is rejected.

System Action: The statement cannot be executed. The specified view was not created.

Programmer Response: Do not attempt to create views with other than your own ID as a qualifier. Only an authorization ID that holds 'SYSADM' or 'DBADM' authority can create views for other authorization IDs. The DBADM privilege should be granted on any of the databases that contain at least one of the tables on which this CREATE VIEW is based.

SQLSTATE: 42502

-170 **THE NUMBER OF ARGUMENTS SPECIFIED FOR** *function-name* **IS INVALID**

Explanation: An SQL statement includes the scalar function 'function-name' with either too many or too few arguments.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement. Refer to Chapter 4 of *DB2 SQL Reference* for information about the number of arguments required by the scalar function 'function-name'.

SQLSTATE: 42605

-171 **THE DATA TYPE, LENGTH, OR VALUE OF ARGUMENT** *nn* **OF** *function-name* **IS INVALID**

Explanation: Either the data type, the length or the value of argument *nn* of scalar function *function-name* is incorrect.

If the encoding scheme is EBCDIC or ASCII, a possible

reason for this error is that a character argument was specified for a built-in function that expects a graphic argument, or a graphic argument was specified for a built-in function that expects a character argument. The UNICODE encoding scheme does support the mixing of character and graphic arguments.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement. Refer to Chapter 4 of *DB2 SQL Reference* for rules for each argument of the scalar function *function-name*.

SQLSTATE: 42815

-173 UR IS SPECIFIED ON THE WITH CLAUSE BUT THE CURSOR IS NOT READ-ONLY

Explanation: The cursor is not a read-only cursor. WITH UR can be specified only if DB2 can determine that the cursor is read-only.

System Action: Statement execution fails.

Programmer Response: If the cursor is intended to be read-only but is ambiguous, add the FOR FETCH ONLY clause. If the cursor is updateable, change the isolation level specified on the WITH clause.

SQLSTATE: 42801

-180 THE DATE, TIME, OR TIMESTAMP VALUE *value* IS INVALID

Explanation: The length or string representation of a DATE, TIME, or TIMESTAMP `value` does not conform to any valid format.

The `value` can contain one of the following:

- For a host variable, the position number of the input host variable. If the position number cannot be determined, a blank is displayed.
- For a character string constant, the character string constant. The maximum length that is displayed is the length of SQLERRM.
- For a character column, the column name. If the column is a VIEW column and it has a corresponding base column, the VIEW column name is displayed. If the column is a VIEW column but it does not have a corresponding base column, a string of '*N' is displayed.

Otherwise, `value` is a string of '*N'.

System Action: The statement cannot be executed.

Programmer Response: Correct the program to ensure the specified value conforms to the syntax of DATE, TIME, and TIMESTAMP. Refer to Chapter 3 of *DB2 SQL Reference* for a list of valid DATE and TIME formats.

SQLSTATE: 22007

-181 THE STRING REPRESENTATION OF A DATETIME VALUE IS NOT A VALID DATETIME VALUE

Explanation: The string representation of a datetime is not in the acceptable range or is not in the correct format. The proper ranges for datetime values are as follows:

Table 2.

Datetime		Numeric Range
Years		0001 to 9999
Months		1 to 12
Days	April, June, September, November (months 4, 6, 9, 11)	1 to 30
	February (month 2)	1 to 28 (Leap year 1 to 29)
	January, March, May, July, August, October, December (months 1, 3, 5, 7, 8, 10, 12)	1 to 31
Hours		0 to 24 (If hour is 24, other parts of time values are zeroes. If hour is USA, maximum hour is 12.)
Minutes		0 to 59
Seconds		0 to 59
Microseconds		0 to 999999

System Action: The statement cannot be executed.

Programmer Response: Check whether the value is within the valid range and is in the proper format. Refer to Chapter 3 of *DB2 SQL Reference* for information on string data formats.

SQLSTATE: 22007

-182 AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE IS INVALID

Explanation: The specified arithmetic expression contains an improperly used datetime value or labeled duration.

System Action: The statement cannot be executed.

Programmer Response: Correct the indicated arithmetic expression.

SQLSTATE: 42816

-183 AN ARITHMETIC OPERATION ON A DATE OR TIMESTAMP HAS A RESULT THAT IS NOT WITHIN THE VALID RANGE OF DATES

Explanation: The result of an arithmetic operation is a date or timestamp that is not within the valid range of dates which are between 0001-01-01 and 9999-12-31.

System Action: The statement cannot be executed.

Programmer Response: Examine the SQL statement to see if the cause of the problem can be determined. The problem may be data-dependent, in which case it will be necessary to examine the data that was processed at the time the error occurred.

SQLSTATE: 22008

-184 AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE CONTAINS A PARAMETER MARKER

Explanation: The specified arithmetic expression contains a parameter marker improperly used with a datetime value.

System Action: The statement cannot be executed.

Programmer Response: Correct the indicated arithmetic expression.

SQLSTATE: 42610

-185 THE LOCAL FORMAT OPTION HAS BEEN USED WITH A DATE OR TIME AND NO LOCAL EXIT HAS BEEN INSTALLED

Explanation: The local format option has been used with a datetime value and no datetime exit has been installed. This may occur if the LOCAL DATE LENGTH or LOCAL TIME LENGTH on the Installation Application Programming Defaults Panel indicated that an exit for datetime was supplied, but in fact the exit supplied by DB2 was not replaced. This may also occur if the datetime exit was replaced and the corresponding LOCAL DATE LENGTH or LOCAL TIME LENGTH on the Installation Application Programming Defaults Panel was not set to a nonzero value.

System Action: The statement cannot be executed.

Programmer Response: Contact the system programmer about installation of the date or time exit.

SQLSTATE: 57008

-186 THE LOCAL DATE LENGTH OR LOCAL TIME LENGTH HAS BEEN INCREASED AND EXECUTING PROGRAM RELIES ON THE OLD LENGTH

Explanation: The local format option has been used with a datetime value and DB2 has discovered that the datetime exit routine has been changed to produce a longer local format.

System Action: The statement cannot be executed.

Programmer Response: If the statement receiving this error is embedded in the application program, then a REBIND command must be issued for the application plan. If the statement was dynamic SQL, then the statement can be reentered.

SQLSTATE: 22505

-187 A REFERENCE TO A CURRENT DATE/TIME SPECIAL REGISTER IS INVALID BECAUSE THE MVS TOD CLOCK IS BAD OR THE MVS PARMTZ IS OUT OF RANGE

Explanation: DB2 has encountered an invalid time-of-day (TOD) clock. The user referenced one of the special registers: CURRENT DATE, CURRENT TIME, CURRENT TIMESTAMP, or CURRENT TIMEZONE. If the user referenced CURRENT TIMEZONE, the MVS parameter PARMTZ was out of range.

System Action: The statement cannot be executed.

Programmer Response: For CURRENT TIMEZONE, check that the MVS parameter PARMTZ is between −24 and +24 hours. For the other CURRENT special registers, check that the MVS TOD clock has been set correctly.

SQLSTATE: 22506

-188 THE STRING REPRESENTATION OF A NAME IS INVALID

Explanation: The host variable referenced in the DESCRIBE statement does not contain a valid string representation of a name. One of the following error conditions has occurred.

- The first byte of the variable is a period or a blank.
- The number of identifiers is greater than 3.
- An identifier is too long.
- A period not contained in a delimited identifier is followed by a period or a blank.
- A delimited identifier is followed by a character other than a period or a blank.
- A delimited identifier is not terminated by a quotation mark.

System Action: The statement cannot be executed.

Programmer Response: Correct the value of the host variable so that it is a valid string representation of a name.

SQLSTATE: 22503

-189 **CCSID *ccsid* IS UNKNOWN OR INVALID FOR THE DATA TYPE OR SUBTYPE**

Explanation: To determine the subtype of an input host variable or result column, the SYSSTRINGS catalog table was accessed with the specified CCSID and:

- The CCSID is not a value of either INCCSID or OUTCCSID, or
- The TRANSTYPE column classifies the CCSID as GRAPHIC rather than CHARACTER, or
- A graphic CCSID has not been specified on your system.

This error can occur when SYSSTRINGS is accessed with a pair of CCSIDs to determine if a translation is defined for the pair. In this case, the error is the inconsistency between the data type of a string and the TRANSTYPE classification of its CCSID (one is GRAPHIC and the other is CHARACTER).

This error can also occur when a CCSID specified in DECP does not exist as a value in the INCCSID or OUTCCSID columns of SYSSTRINGS.

Another reason this error can occur is that you may be using one of the graphic built-in functions but a graphic CCSID was not specified during system installation.

System Action: The statement cannot be bound or executed.

Programmer Response: Ensure that the CCSSID is valid and consistent with the data type of the string. If a valid CCSID is not listed in a built-in row of SYSSTRINGS, it can be defined by inserting a user-provided row. If a valid CCSID is misclassified in a user-provided row, that row can be updated to correct the mistake. Refer to the appendices of *DB2 Installation Guide* for more information on CCSIDs and to *DB2 SQL Reference* for more information on the SYSSTRINGS catalog table.

If a graphic CCSID had not been specified at system installation, update your DECP to include a graphic CCSID and recycle your DB2.

SQLSTATE: 22522

-190 **ATTRIBUTES OF COLUMN *column-name* IN TABLE *table-name* ARE NOT COMPATIBLE WITH THE EXISTING COLUMN**

Explanation: The attributes specified for the column of the specified table in an ALTER statement are not compatible with the attributes of the existing column. Either the data type or length is not valid:

- The new altered length for the column is of different length than the current length of the column.
- The existing column is not a VARCHAR data type.

System Action: The ALTER statement cannot be executed.

Programmer Response: Specify attributes that are compatible with the existing column.

SQLSTATE: 42837

-191 **A STRING CANNOT BE USED BECAUSE IT IS INVALID MIXED DATA**

Explanation: The operation required the translation of a mixed data character string to a different coded character set. The string could not be translated because it does not conform to the rules for well-formed mixed data. For example, the string contains EBCDIC shift codes that are not properly paired.

System Action: The statement cannot be executed.

Programmer Response: If the string contains the intended information, the description of the column or host variable should be changed from MIXED DATA to BIT or SBCS DATA. If the description of the column or host variable is correct, the string is the problem and it must be changed to conform to the rules for well-formed mixed data. For more information about well-formed MIXED DATA refer to Chapter 3 of *DB2 SQL Reference*.

SQLSTATE: 22504

-197 **QUALIFIED COLUMN NAMES IN ORDER BY CLAUSE NOT PERMITTED WHEN UNION OR UNION ALL SPECIFIED**

Explanation: A SELECT statement that specifies both the union of two or more tables and the ORDER BY clause cannot use qualified column names in the ORDER BY clause.

Programmer Response: Change the statement so that qualified names are not necessary in the ORDER BY clause.

System Action: The statement is not executed.

SQLSTATE: 42877

-198 **THE OPERAND OF THE PREPARE OR EXECUTE IMMEDIATE STATEMENT IS BLANK OR EMPTY**

Explanation: The operand (host variable or literal string) that was the object of the PREPARE or EXECUTE IMMEDIATE statement either contained all blanks or was an empty string. A DBRM built in Version 2 Release 3 cannot be used on a Version 2 Release 2 system if the distributive functions were used. If this error appears on Version 2 Release 2 and the DBRM was built on Version 2 Release 3,the program needs to be precompiled again to correct the problem.

System Action: The statement cannot be executed.

Programmer Response: Correct the logic of the application program to ensure that a valid SQL statement is provided in the operand of the PREPARE or EXECUTE IMMEDIATE statement before that statement is executed.

SQLSTATE: 42617

-199 **ILLEGAL USE OF KEYWORD** *keyword*. **TOKEN** *token-list* **WAS EXPECTED**

Explanation: A syntax error was detected in the statement at the point where the keyword 'keyword' appears.

As an aid to the programmer, a partial list of valid tokens is provided in SQLERRM as 'token-list'. Only those tokens that will fit are listed. Some tokens in the list might not be valid in statements to be executed by DB2; those tokens are valid for sending to other database management systems.

System Action: The statement cannot be executed.

Programmer Response: Examine the statement in the area of the indicated keyword. A colon or SQL delimiter might be missing.

SQLSTATE: 42601

-203 **A REFERENCE TO COLUMN** *column-name* **IS AMBIGUOUS**

Explanation: An unqualified column name is ambiguous if more than one table or view identified in the FROM clause has a column with that name, or if more than one column of a nested table expression has that name.

A qualified column name is ambiguous only if the qualifier is the correlation name for a nested table expression and the column name is not unique.

A reference to a column of the triggering table in a CREATE TRIGGER statement is ambiguous if it does not use the correlation name to indicate if it refers to the old or new transition variable.

System Action: The statement cannot be executed.

Programmer Response: If the problem is caused by a nonunique column name in a nested table expression, change the nested table expression so that the column name is unique. If the problem is caused by the use of an unqualified name, qualify it with a table, view, or correlation name.

SQLSTATE: 42702

-204 *name* **IS AN UNDEFINED NAME**

Explanation: A routine was invoked. The routine invocation was not accepted because of DB2 reason code *rc*.

name The name of the routine that was invoked.

rc The DB2 reason code describing the cause of the failure. Possible values are: 00E79000, 00E79001, 00E79002, 00E79003, 00E79004, 00E79005, 00E79006, 00E79007, 00E7900B, 00E7900C.

System Action: The statement cannot be executed.

Programmer Response: Verify that the object name was correctly specified in the SQL statement, including any required qualifiers. If it is correct, ensure that the object exists in the system before resubmitting the statement.

For missing a data type or function in the SOURCE clause, it may be that the object does not exist, OR it may be that the object does exist in some schema, but the schema is not present in your current path.

SQLSTATE: 42704

-205 *column-name* **IS NOT A COLUMN OF TABLE** *table-name*

Explanation: No column with the specified 'column-name' occurs in the table or view 'table-name'.

System Action: The statement cannot be executed.

Programmer Response: Verify that the column and table names are specified correctly (including any required qualifiers) in the SQL statement.

SQLSTATE: 42703

-206 *column-name* **IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE, OR IS NOT A COLUMN OF THE TRIGGERING TABLE OF A TRIGGER**

Explanation: This return code is used to report one of the following errors:

- In the case of an INSERT or UPDATE statement, the specified column is not a column of the table or view that was specified as the object of the insert or update.
- In the case of an INSERT with VALUES clause, there is a column referenced and columns are not allowed in the VALUES clause.
- In the case of a SELECT or DELETE statement, the specified column is not a column of any of the tables or views identified in a FROM clause in the statement.
- There is a correlated reference in the GROUP BY clause in the select list of a subselect, or a correlated reference is not used in a search condition.
- There is an unresolved qualified reference in HAVING.
- For a CREATE TRIGGER statement:

- A reference is made to a column using an OLD or NEW correlation name. The column name is not defined in the triggering table.
- The left side of an assignment in the SET transition-variable statement in the triggered action specifies an old transition variable where only a new transition variable is supported or trigger was not created.

System Action: The statement cannot be executed. No data was retrieved, inserted, or updated or the trigger was not created.

Programmer Response: Verify that the column and table names are specified correctly in the SQL statement. In the case of a SELECT statement, check to be sure that all of the required tables were named in the FROM clause.

In the case of a CREATE TRIGGER statement, ensure that only new transition variables are specified on the left side of assignments in the SET transition-variable statement and that any reference to columns of the triggering table are qualified with a transition variable correlation name.

SQLSTATE: 42703

-208 **THE ORDER BY CLAUSE IS INVALID BECAUSE COLUMN** *name* **IS NOT PART OF THE RESULT TABLE**

Explanation: The statement is invalid because a column ('name') specified in the ORDER BY list does not appear in the result table (that is, it is not specified in the SELECT-list). Only columns in the result table can be used to order that result when the *fullselect* of the *select-statement* is not a *subselect*.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the statement, either by adding the specified column to the result table, or deleting it from the ORDER BY clause. Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on the use of the ORDER BY clause to order the result of an SQL SELECT.

SQLSTATE: 42707

-212 *name* **IS SPECIFIED MORE THAN ONCE IN THE REFERENCING CLAUSE OF A TRIGGER DEFINITION**

Explanation: The REFERENCING clause of a CREATE TRIGGER statement specified the same name for more than one of the OLD or NEW correlation names or the OLD_TABLE or NEW_TABLE identifiers. *name* is the name that was specified multiple times.

System Action: The statement cannot be executed. The trigger was not created.

Programmer Response: Change the statement to specify unique names for all transition variables and tables in the REFERENCING clause and resubmit the CREATE TRIGGER request.

SQLSTATE: 42712

-214 **AN EXPRESSION IN THE FOLLOWING POSITION, OR STARTING WITH** *position-or-expression-start* **IN THE** *clause-type* **CLAUSE IS NOT VALID. REASON CODE =** *reason-code*

Explanation: The expression identified by the first part of the expression *expression-start* in the *clause-type* clause is not valid for the reason specified by the *reason-code* as follows:

- 1- The fullselect of the select-statement is not a subselect. Expressions are not allowed in the ORDER BY clause for this type of select-statement. This reason code occurs only when *clause-type* is ORDER BY.
- 2 - DISTINCT is specified in the select clause, and either a column name in the ORDER BY clause cannot be matched exactly with a column name in the SELECT list, or a *sort-key-expression* is specified in the ORDER BY clause. This reason code occurs only when *clause-type* is ORDER BY.

System Action: The statement cannot be executed.

Programmer Response: Modify the select-statement based on the reason specified by the *reason-code* as follows:

- 1 - Remove the expression from the ORDER BY clause. If attempting to reference a column of the result, change the sort key to the *simple-integer* or *simple-column-name* form. See the ORDER BY syntax diagram in the DB2 SQL Reference for more information.
- 2 - Remove DISTINCT from the select clause.

SQLSTATE: 42822

-216 **THE NUMBER OF ELEMENTS ON EACH SIDE OF A PREDICATE OPERATOR DOES NOT MATCH. PREDICATE OPERATOR IS** *operator***.**

Explanation: The number of expressions specified on the left-hand side of OPERATOR *operator* is unequal to either the number of values returned by the fullselect or to the number of expressions specified on the right-hand side of the operator. The number of expressions and the number of values/expressions on either side of the operator must be equal.

System Action: The statement was not executed.

Programmer Response: Change the number of expressions to match the number of values returned by the fullselect or vice versa.

SQLSTATE: 428C4

-219 **THE REQUIRED EXPLANATION TABLE *table-name* DOES NOT EXIST**

Explanation: The EXPLAIN statement assumes the existence of the explanation table and it is not defined in the DB2 subsystem as a base table. Refer to Chapter 6 of *DB2 SQL Reference* for more information.

System Action: The statement cannot be executed.

Programmer Response: Determine whether the required explanation table does exist. If not, create the required table.

SQLSTATE: 42704

-220 **THE COLUMN *column-name* IN EXPLANATION TABLE *table-name* IS NOT DEFINED PROPERLY**

Explanation: An error occurred during the insertion of a row into the explanation table. The table is improperly defined for the following reasons:

- A column is missing.
- Columns are defined in the wrong order.
- The table contains an extra column.
- A column description is invalid because of its name, data type, length, or null attributes.

System Action: The statement cannot be executed. The explanation information is not generated.

Programmer Response: Correct the definition of the required explanation table. Refer to Chapter 6 of *DB2 SQL Reference* for information on defining an explanation table.

SQLSTATE: 55002

-221 **"SET OF OPTIONAL COLUMNS" IN EXPLANATION TABLE *table-name* IS INCOMPLETE. OPTIONAL COLUMN *column-name* IS MISSING**

Explanation: The EXPLAIN statement assumes the required explanation table is defined properly. The optional column indicated is not defined in the indicated explanation table. PLAN_TABLEs must have one of several specific formats. The format chosen must be complete, and each column in the PLAN_TABLE definition must be correct for the chosen format. The allowed formats for the PLAN_TABLE are described in Chapter 6 of *DB2 SQL Reference*.

System Action: The explanation information is not generated.

Programmer Response: Correct the definition of the required explanation table to include all of the optional columns in the chosen format, just the Version 2 Release 2 optional columns, or no optional columns. Refer to Chapter 6 of *DB2 SQL Reference* for information on defining an explanation table.

SQLSTATE: 55002

-222 **AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE USING *cursor-name***

Explanation: DB2 could not process a positioned update or delete with cursor *cursor-name* that is defined as SENSITIVE STATIC. The selected row is either a delete hole or an update hole. DB2 detects these holes when DB2 tries to delete or update the current row of the result table for cursor *cursor-name*, and cannot locate the corresponding row of the underlying table.

A *delete hole* occurs when the corresponding row of the underlying table has been deleted.

An *update hole* occurs when the corresponding row of the underlying table has been updated, and the updated row no longer satisfies the search condition that is specified in the SELECT statement of the cursor.

System Action: The statement cannot be processed. The cursor is positioned on the hole.

Programmer Response: Issue a FETCH statement to position the cursor on a row.

SQLSTATE: 24510

-223 **AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST AN UPDATE HOLE USING *cursor-name***

Explanation: DB2 detected an *update hole* when DB2 attempted a positioned UPDATE or DELETE on a row that no longer satisfies its previous search condition. An *update hole* is created when a row exists in the result table and the resulting row has been updated in the base table such that the row no longer satisfies the search condition in the SELECT statement.

cursor-name
Name of the cursor used for the positioned update or delete.

System Action: The statement cannot be processed. The cursor is positioned on the table.

Programmer Response: Correct the application program to handle this error condition or change isolation levels so the base row cannot be updated during the cursor operation.

SQLSTATE: 24511

-224 **THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING *cursor-name***

Explanation: DB2 attempted a positioned UPDATE or DELETE was attempted on a row that no longer matches its previous condition. The column values in the result table row do not match the current values in the base table row because the row was updated

between the time it was inserted into the result table and the positioned update or delete was executed.

cursor-name
Name of the cursor used for the positioned update or delete.

System Action: The statement cannot be processed. The cursor is positioned on the same row.

Programmer Response: Correct the application program to handle this error condition or change isolation levels so the base row cannot be updated during the cursor operation.

SQLSTATE: 24512

-225 FETCH STATEMENT FOR *cursor-name* IS NOT VALID BECAUSE THE CURSOR IS NOT DEFINED AS SCROLL

Explanation: DB2 could not process a FETCH statement for cursor *cursor-name* because it contained a disallowed keyword. You may only specify the keyword NEXT for non-scrollable cursors. The keywords PRIOR, FIRST, LAST, ABSOLUTE, RELATIVE, CURRENT, BEFORE, and AFTER are disallowed for a cursor that was not declared with the SCROLL attribute.

cursor-name
Name of the cursor used for the FETCH statement.

System Action: The statement cannot be processed.

Programmer Response: Correct the FETCH statement to excluse the disallowed keyword, or corect the DECLARE CURSOR statement to include the appropriate SCROLL option.

SQLSTATE: 42872

-228 FOR UPDATE CLAUSE SPECIFIED FOR READ-ONLY CURSOR *cursor-name*

Explanation: A cursor was declared read-only with the INSENSITIVE SROLL option, but the SELECT statement contained a FOR UPDATE clause.

cursor-name
Name of the cursor used for the FETCH.

System Action: The statement cannot be processed.

Programmer Response: To define a scrollable cursor that is read-only, specify INSENSITIVE SCROLL, but do not specify FOR UPDATE clause. To define a scrollable cursor that can be updated, specify SENSITIVE SCROLL. Corect the application program to DECLARE CURSOR appropriately.

SQLSTATE: 42620

-229 THE LOCALE *locale* SPECIFIED IN A SET LOCALE OR OTHER STATEMENT THAT IS LOCALE SENSITIVE WAS NOT FOUND

Explanation: The statement attempted to reference a Locale that is not known or not available to DB2. *locale* is the locale that was either specified on the SET CURRENT LOCALE statement or the locale that was in effect at the time the Locale access was attempted.

System Action: The statement cannot be executed.

Programmer Response: If the statement was a SET LOCALE statement, re-specify a locale that is correct (known and available to DB2). Refer to Chapter 3 of SQL Reference for more information on Locales. If the statement was something other than SET LOCALE, then the statement contained a locale sensitive interface (the UPPER function is an example of a locale sensitive interface). Issue ″SELECT CURRENT LOCALE FROM SYSIBM.SYSDUMMY1″ to determine the value of the LOCALE in use by your program. Possible reasons for this message include an incorrect LOCALE bind option, or an incorrect LOCALE default value specified at installation time (The value of a Locale is not validated until it is needed in a Locale sensitive interface). Because Locales are dynamic in nature, they can be added, created, or deleted at anytime, they are not validated until they are used. Therefore, it is possible to specify a locale that is not valid at installation or bind time.

SQLSTATE: 42708

-240 THE PART CLAUSE OF A LOCK TABLE STATEMENT IS INVALID

Explanation: The LOCK TABLE statement is invalid for one of the following reasons:

- The table space in which the table resides is not partitioned or does not have the LOCKPART YES attribute, and the PART clause is specified.
- An integer specified in the PART clause does not identify a partition of the table space.

System Action: The LOCK TABLE statement cannot be executed.

Programmer Response: Determine whether the specified table resides in a partitioned table space defined with LOCKPART YES.

- If it is partitioned and defined with LOCKPART YES, specify a PART clause that identifies the partition you want to lock.
- If it is partitioned but does not have the LOCKPART YES attribute and you want to lock a single partition, use ALTER TABLESPACE to change the LOCKPART attribute to YES.
- If it is not partitioned, do not specify the PART clause.

SQLSTATE: 428B4

-243 **SENSITIVE CURSOR** *cursor-name* **CANNOT BE DEFINED FOR THE SPECIFIED SELECT STATEMENT**

Explanation: The cursor *cursor-name* is defined as SENSITIVE, but the content of of the SELECT statement requires DB2 to build a temporary table with the result table of the cursor, and DB2 cannot guarantee that changes made outside the cursor will be visible. This could result from the content of the query making the result table read-only. In this case the cursor must be defined INSENSITIVE.

System Action: The statement cannot be processed.

Programmer Response: Either change the content of the query to not be read-only, or change the type of cursor to be INSENSITIVE.

SQLSTATE: 36001

-244 **SENSITIVITY** *sensitivity* **SPECIFIED ON THE FETCH IS NOT VALID FOR CURSOR** *cursor-name*

Explanation: The sensitivity option specified on FETCH conflicts with the sensitivity option in effect for cursor *cursor-name*. If a cursor is declared INSENSITIVE, the FETCH statement can only specify INSENSITIVE or nothing. If a cursor is declared SENSITIVE, the FETCH statement can specify INSENSITIVE, SENSITIVE, or nothing.

sensitivity
Specified sensitivity for the FETCH statement.

cursor-name
Name of the cursor used for the FETCH statement.

In case of a non-scrollable cursor, sensitivity option cannot be specified.

System Action: The statement cannot be processed.

Programmer Response: Change the host variable to be an exact numeric with a scale of zero.

SQLSTATE: 428F4

-245 **THE INVOCATION OF FUNCTION ROUTINE-NAME IS AMBIGUOUS**

Explanation: DB2 issues this error when an invocation of a function is ambiguous. This occurs when an untyped parameter marker is passed to a function and there are two or more possible candidate functions to resolve to during function resolution.

System Action: The statement cannot be processed.

Programmer Response: Fix the problem and retry. This could involve a change to the SQL statement, changing the definition of a function or a change to the user's current path. See the DB2 Application Programming and SQL Guide for details on function resolution.

SQLSTATE: 428F5

-250 **THE LOCAL LOCATION NAME IS NOT DEFINED WHEN PROCESSING A THREE-PART OBJECT NAME**

Explanation: A three-part object name (table, view, or alias) cannot be used until the local location name is defined.

System Action: Install or reinstall the DB2 distributed data facility (DDF) with a registered location name for local DB2.

Programmer Response: Define the local location name and then retry the function.

SQLSTATE: 42718

-251 **TOKEN** *name* **IS NOT VALID**

Explanation: A *location name* cannot contain alphabetic extenders. (The standard alphabetic extenders in the United States are #, @, $.)

System Action: The statement cannot be executed

Programmer Response: Correct the name and reissue the statement.

SQLSTATE: 42602

-300 **THE STRING CONTAINED IN HOST VARIABLE OR PARAMETER** *position-number* **IS NOT NUL-TERMINATED**

Explanation: A host variable or parameter is invalid. Its entry in the SQLDA is indicated by *position-number*. The host variable or parameter is a C string variable that is one of the following:

- Used as an input parameter to a stored procedure or function.
- Returned as an output parameter from a stored procedure or function.
- Referenced as an input variable in an embedded SQL statement.
- Used to provide a value for a parameter marker of a dynamic SQL statement.

If the data type of the variable is character string, it is invalid because it does not include X'00'. If the data type of the variable is graphic string, it is invalid because it does not include X'0000'.

System Action: The statement cannot be executed.

Programmer Response: Append a NUL-terminator to the end of the string.

SQLSTATE: 22024

-301 **THE VALUE OF INPUT HOST VARIABLE OR PARAMETER NUMBER** *position-number* **CANNOT BE USED AS SPECIFIED BECAUSE OF ITS DATA TYPE**

Explanation: DB2 received data that could not be used as specified in the statement because its data type is incompatible with the requested operation.

The *position-number* identifies either the host variable number (if the message is issued as a result of an INSERT, UPDATE, DELETE, SELECT, VALUE INTO, or SET assignment statement), or the parameter number (if the message is issued as the result of a CALL statement, or the invocation of a function).

System Action: The statement cannot be executed.

Programmer Response: Correct the application program, function or stored procedure. Ensure that the data type of the indicated input host variable or parameter in the statement is compatible with the way it is used.

SQLSTATE: 42895

-302 **THE VALUE OF INPUT VARIABLE OR PARAMETER NUMBER** *position-number* **IS INVALID OR TOO LARGE FOR THE TARGET COLUMN OR THE TARGET VALUE**

Explanation: DB2 received data that was invalid or too large to fit in the corresponding column of the table or the corresponding target value. The *position-number* identifies either the host variable number (if the message is issued as a result of an INSERT, UPDATE, DELETE, SELECT, VALUES INTO, or SET assignment statement), or the parameter number (if the message is issued as the result of a CALL statement or the invocation of a function).

One of the following occurred:

- The column is defined as a string and the host variable or parameter contains a string that is too long for the column.
- The column is defined as numeric and the host variable or parameter contains a numeric value too large for the definition of the column.
- The host variable is defined as decimal, but contains invalid decimal data.
- The target value is a string constant and the host variable or parameter contains a string that is too long for the target value.
- The target value is a numeric constant and the host variable or parameter contains a numeric value that is too large for the target value.

System Action: The statement cannot be executed.

Programmer Response: Correct the application program, function or stored procedure. Check the column type and length of the value or the data type and contents of the input host variable or parameter *position-number*. Ensure that the value of the host variable or parameter will fit in the column or contains valid decimal data. Valid decimal data is a System/370 packed decimal number.

SQLSTATE: 22003 if *number too large for target*; 22001 otherwise.

-303 **A VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER** *position-number* **BECAUSE THE DATA TYPES ARE NOT COMPARABLE**

Explanation: A CALL, FETCH, SELECT, VALUES INTO, or SET *host-variable* statement with an output host variable, whose entry in the output SQLDA is indicated by *position-number*, could not be performed. The data type of the variable was not compatible with the data type of the corresponding SELECT, VALUES INTO, or SET *host-variable* statement list element. The values of the output host variable and the corresponding list element must be in one of the following categories:

- Both must be numbers.
- Both must be character strings if not using Unicode.
- Both must be graphic strings if not using Unicode.
- Both must be row IDs.

In addition, for datetime, timestamp values, the host variable must be a character string variable with a correct length.

System Action: The CALL, FETCH, SELECT, VALUES INTO, or SET *host-variable* statement cannot be executed. No data was retrieved.

Programmer Response: Verify that table definitions are current and that the host variable has the correct data type.

SQLSTATE: 42806

-304 **A VALUE WITH DATA TYPE** *data-type1* **CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION** *position-number* **WITH DATA TYPE** *data-type2*

Explanation: A CALL, FETCH, SELECT, VALUES INTO, or SET assignment statement with a host variable list or structure in position number *position-number* failed because the host variable with data type *data-type2* was not large enough to hold the retrieved value with data type *data-type1*.

System Action: The statement cannot be executed. No data was retrieved. If the statement was a FETCH, the cursor remains open.

Programmer Response: Verify that table definitions

are current, and that the host variable has the correct data type. See the explanation for SQLCODE -405 for ranges of SQL data types.

SQLSTATE: 22003

-305 **THE NULL VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER** *position-number* **BECAUSE NO INDICATOR VARIABLE IS SPECIFIED**

Explanation: A FETCH, SELECT, VALUES INTO, or SET assignment statement resulted in the retrieval of a null value to be inserted into the output host variable, designated by entry number 'position-number' of the output SQLDA, for which no indicator variable was provided. An indicator variable must be supplied if a column returns a null value.

System Action: The statement cannot be executed. No data was retrieved.

Programmer Response: Examine the definition of the table that is the object of the statement and correct the application program to provide indicator variables for all host variables into which null values can be retrieved. This includes host variables for columns which can contain null values and host variables which receive the results of column functions whose result table could be empty.

SQLSTATE: 22002

-309 **A PREDICATE IS INVALID BECAUSE A REFERENCED HOST VARIABLE HAS THE NULL VALUE**

Explanation: The statement could not be processed because a host variable appearing in a predicate such as

```
column-name = host-variable
```

had the NULL value. Such a predicate is not permitted when the host variable contains the NULL value even though the object column might contain nulls.

System Action: The statement cannot be executed.

Programmer Response: Rebind the plan or package containing the statement. The condition described is not an error in DB2 Version 2 Release 3 and later releases.

SQLSTATE: 22512

-310 **DECIMAL HOST VARIABLE OR PARAMETER** *number* **CONTAINS NON-DECIMAL DATA**

Explanation: DB2 received nondecimal data from either an application (in the form of a host variable), function or a stored procedure (in the form of a parameter that was passed to or from function or a stored procedure).

number Identifies either the host variable number (if the message is issued as a result of a FETCH, INSERT, UPDATE, DELETE, SELECT, VALUES INTO, or SET assignment statement statement), or the parameter number (if the message is issued as the result of the invocation of a function, or a CALL statement).

System Action: The statement cannot be processed.

Programmer Response: Correct the application program or stored procedure. Ensure that all decimal variables or parameters contain valid System/370 packed decimal numbers.

SQLSTATE: 22023

-311 **THE LENGTH OF INPUT HOST VARIABLE NUMBER** *position-number* **IS NEGATIVE OR GREATER THAN THE MAXIMUM**

Explanation: When evaluated, the length specification for input host string variable, whose entry in the SQLDA is indicated by position-number, was negative or greater than the maximum.

System Action: The statement cannot be executed.

Programmer Response: Correct the program to ensure that the lengths of all host string variables are not negative or that they are not greater than the maximum allowed length.

SQLSTATE: 22501

-312 *variable-name* **IS AN UNDEFINED OR UNUSABLE HOST VARIABLE OR IS USED IN A DYNAMIC SQL STATEMENT OR A TRIGGER DEFINITION**

Explanation: The host variable *variable-name* appears in the SQL statement, but:

- The SQL statement is a prepared statement, or
- The attributes of the variable are inconsistent with its usage in the static SQL statement, or
- The variable is not declared in the application program or
- The variable appeared in one of the triggered SQL statements in a CREATE TRIGGER statement.

System Action: The statement cannot be executed.

Programmer Response: Verify that

- The variable name is spelled properly in the SQL statement.
- The variable is allowed in the SQL statement.
- The application program contains a declaration for that variable.
- The attributes of the variable are compatible with its use in the statement.

SQLSTATE: 42618

-313 THE NUMBER OF HOST VARIABLES SPECIFIED IS NOT EQUAL TO THE NUMBER OF PARAMETER MARKERS

Explanation: The number of host variables specified in the EXECUTE or OPEN statement is not the same as the number of parameter markers (question marks) appearing in the prepared SQL statement.

System Action: The statement cannot be executed.

Programmer Response: Correct the application program so that the number of host variables specified in the EXECUTE or OPEN statement is the same as the number of parameter markers appearing in the prepared SQL statement. The DESCRIBE INPUT SQL statement can be used to determine the expected number of input parameter markers.

SQLSTATE: 07001

-314 THE STATEMENT CONTAINS AN AMBIGUOUS HOST VARIABLE REFERENCE

Explanation: A host variable used in the statement has been defined more than once in this application program causing confusion as to which host variable defined should be used.

System Action: The statement cannot be executed.

Programmer Response: Make the host variable unique or use qualifications to indicate which host variable definition is to be used.

SQLSTATE: 42714

-327 THE ROW CANNOT BE INSERTED BECAUSE IT IS OUTSIDE THE BOUND OF THE PARTITION RANGE FOR THE LAST PARTITION

Explanation: When a row is inserted, the calculated partition key value for the new row must be within the bounds of a partition (as specified in the VALUES clause of the CREATE INDEX statement).

System Action: The statement cannot be executed.

Programmer Response: Correct the statement to specify a value for the partition key that is within the bounds of the last partition of the partitioned table space.

SQLSTATE: 22525

-330 A STRING CANNOT BE USED BECAUSE IT CANNOT BE TRANSLATED. REASON *reason-code, CHARACTER code-point*, **HOST VARIABLE** *position-number*

Explanation: A translation error occurred during the translation of a string to a different coded character set. The type of error is indicated by the *reason-code*.

4 Substitution exception. The string cannot be converted with substitution characters being placed in the target string.

8 :dd.Length exception (for example, expansion required for PC MIXED data exceeds the maximum length of the string).

12 Invalid code point (for example, use of the ERRORBYTE option of SYSSTRINGS).

16 Form exception (for example, invalid MIXED data).

20 Translate procedure error (for example, an exit set the length control field of the string to an invalid value).

24 SBCS character found in string contained in a wchar_t host variable.

If the *reason-code* is 12, *code-point* is the invalid code point. Otherwise, *code-point* is either blank or an additional *reason-code* returned by an exit. If the string is the value of an input host variable, the *position-number* is the ordinality of the variable in the SQLDA. If the string is not the value of a host variable, the *position-number* is blank.

System Action: The statement cannot be executed.

Programmer Response: Take one of the following actions based on the *reason-code*:

- If the *reason-code* is 4, then the operation that is being performed is one that cannot continue when a substitution occurs in a conversion. Change the data to eliminate the *code-point*.
- If the *reason-code* is 8, extend the maximum length of the host variable to allow for the expansion that occurs when the string is translated.
- If the *reason-code* is 12, either change the translate table to accept the *code-point* or change the data to eliminate the *code-point*.
- If the *reason-code* is 16 and the string is described as MIXED data, either change its description or the string to conform to the rules for well-formed mixed data.
- If the *reason-code* is 20, correct the translate procedure.
- If the *reason-code* is 24, delete the SBCS character from the graphic string.

SQLSTATE: 22021

-331 **A STRING CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE IT CANNOT BE TRANSLATED. REASON** *reason-code*, **CHARACTER** *code-point*, **POSITION** *position-number*

Explanation: The operation required the translation of a string to the coded character set of the host variable and a translation error occurred. The type of error is indicated by the 'reason-code':

- 8 for length exception (e.g., expansion required for PC MIXED data exceeds the maximum length of the string).
- 12 for invalid 'code point' (e.g., use of the ERRORBYTE option of SYSSTRINGS).
- 16 for form exception (e.g., invalid MIXED data).
- 20 for translate procedure error (e.g., an exit set the length control field of the string to an invalid value).

If the 'reason-code' is 12, 'code-point' is the invalid 'code point'. Otherwise, 'code-point' is blank. The 'position-number' is the ordinality of the output variable in the SQLDA.

System Action: The statement cannot be executed.

Programmer Response: If the 'reason-code' is 8, the maximum length of the result column must be extended to allow for the expansion that occurs when the string is translated. If the 'reason-code' is 12, either the translate table must be changed to accept the 'code-point' or the data must be changed to eliminate the 'code point'. If the 'reason-code' is 16, and the string is described as MIXED data, either its description must be changed or the string must be changed to conform to the rules for well-formed MIXED data. If the 'reason-code' is 20, the translate procedure must be corrected. An alternative to these corrective actions is to provide an indicator variable so that a null value and a warning can be returned rather than an error. Refer to Chapter 3 of *DB2 SQL Reference* for more information on coded character set.

SQLSTATE: 22021

-332 **CHARACTER CONVERSION BETWEEN CCSID** *from-ccsid* **TO** *to-ccsid* **REQUESTED BY** *reason-code* **IS NOT SUPPORTED**

Explanation: The operation required a conversion between two differend CCSIDs, but no conversion support was found.

from-ccsid identifies the coded character set of the string to be converted.

to-ccsid identifies the coded character set to which it must be translated.

reason code describes the reason codes returned from DB2. Reason codes returned from DB2 begin with 'DSN' and identify the context in which the conversion was requested. Values other than those that start with 'DSN' are returned from other DB2 platforms and are described in the documentation for the platform.

System Action: The statement cannot be processed.

Programmer Response: If the conversion request is correct, refer to Appendix B of DB2 Installation Guide for information on how to add conversion support.

SQLSTATE: 57017

-333 **THE SUBTYPE OF A STRING VARIABLE IS NOT THE SAME AS THE SUBTYPE KNOWN AT BIND TIME AND THE DIFFERENCE CANNOT BE RESOLVED BY TRANSLATION**

Explanation: The CCSID in the run time SQLDA is inconsistent with the bind time subtype of the host variable or parameter marker. Either the run time description is BIT and the bind time description was not BIT, or the run time description is not BIT and the bind time description was BIT.

System Action: The statement cannot be executed.

Programmer Response: Change the CCSID in the SQLDA so that the subtype of the host variable is consistent with the bind time subtype of the host variable or parameter marker. If the input data in error is a parameter marker, you can use the DESCRIBE INPUT SQL statement to determine the expected SQLTYPE, SQLLEN and CCSID expected. Refer to Chapter 3 of *DB2 SQL Reference* for more information on coded character set.

SQLSTATE: 56010

-338 **AN ON CLAUSE IS INVALID**

Explanation: This return code reports a violation of one of the following:

- One expression of the predicate must only reference columns of one of the operand tables of the associated join operator, full join, and the other expression of the predicate must only reference columns of the other operand table.
- A VALUE or COALESCE function is allowed in the ON clause only when the join operator is a FULL OUTER JOIN or FULL JOIN.
- An operator other than '=' is not allowed in a FULL OUTER JOIN or FULL JOIN.
- A subquery is not allowed in the ON clause.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax so that it doesn't violate any of the above items within the ON clause

SQLSTATE: 42972

-339 THE SQL STATEMENT CANNOT BE EXECUTED FROM AN ASCII BASED DRDA APPLICATION REQUESTOR TO A V2R2 DB2 SUBSYSTEM

Explanation: The application is connected to a DB2 Version 2 Release 3 database server. The SQL statement is using an alias or three-part name, which refers to another DB2 subsystem that is at the Version 2 Release 2 level. DB2 Version 2 Release 2 does not support character conversion. Since the execution of SQL statements from an ASCII DRDA requester to an EBCDIC Version 2 Release 2 DB2 server could require character conversion, access to the Version 2 Release 2 DB2 is denied for data integrity reasons.

System Action: The statement cannot be executed.

Programmer Response: Remove statements from the application that resolve to a DB2 Version 2 Release 2 subsystem.

System Programmer Response: If the application must refer to the Version 2 Release 2 subsystem data, the Version 2 Release 2 DB2 subsystem must be migrated to Version 2 Release 3 where character conversion is supported.

SQLSTATE: 56082

-350 INVALID SPECIFICATION OF A LARGE OBJECT COLUMN

Explanation: The ALTER TABLE, CREATE TABLE, or CREATE INDEX statement is invalid for one of the following reasons:

- A LOB column cannot be added to a temporary table.
- A LOB column cannot be added to a table defined with an EDITPROC.
- The *PRIMARY KEY* clause cannot specify a LOB column as a column of the primary key.
- The *UNIQUE* clause cannot specify a LOB column as a column of the unique key.
- The *referential-constraint* clause cannot specify a LOB column as a column of a foreign key.
- The CREATE INDEX statement cannot name a LOB column as a column of the index key.
- A LOB column cannot be specified in a *references* clause.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax and resubmit the statement.

SQLSTATE: 42962

-351 AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION *position-number* OF THE SELECT-LIST

Explanation: *position-number* is the position of the first element in the SQLDA with an unsupported data type. Either the application requestor or the application server does not have support for this type. This error can only occur in a client/server environment.

System Action: The statement cannot be executed.

Programmer Response: Change the statement to exclude the unsupported data type. For a select statement, remove the names of any columns in the select-list with the unsupported data types.

SQLSTATE: 56084

-352 AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION *position-number* OF THE INPUT-LIST

Explanation: The input SQLDA for an OPEN, EXECUTE, FETCH, or CALL statement contains an unsupported SQLTYPE for the parameter in position *position-number*. *position-number* is the position of the first element in the SQLDA with an unsupported data type. Either the application requestor or the application server does not have support for this data type. This error can only occur in a client/server environment.

System Action: The statement cannot be executed.

Programmer Response: Change the SQLDA to exclude the unsupported data type.

SQLSTATE: 56084

-355 A LOB COLUMN IS TOO LARGE TO BE LOGGED

Explanation: One of the following has occurred:

- a CREATE TABLE statement for an auxiliary table stores a BLOB, CLOB or DBCLOB column whose length exceeds 1 gigabyte but whose associated LOB table space was defined with the LOG YES attribute
- an ALTER TABLESPACE statement of a LOB table space specifies the *LOG YES* clause but the auxiliary table in the LOB table space stores a BLOB or CLOB column whose length exceeds 1 gigabyte or a DBCLOB column whose length exceeds 500 megabyte characters

System Action: The statement cannot be executed.

Programmer Response: Either change the attribute of the LOB table space to LOG NO or drop the base table, and recreate it with columns of an acceptable length for logging.

SQLSTATE: 42993

-359 THE RANGE OF VALUES FOR THE IDENTITY COLUMN IS EXHAUSTED

Explanation: An INSERT statement was issued against a table with an identity column; however all allowable values for the identity column data type have already been assigned, assuming NO CYCLE is in effect.

System Action: The statement cannot be processed.

Programmer Response: Redefine the table to use a data type on the identity column that supports a larger range of values. For example, if SMALLINT is currently specified, change the data type for the identity column to INTEGER.

SQLSTATE: 23522

-372 ONLY ONE ROWID OR IDENTITY COLUMN IS ALLOWED IN A TABLE

Explanation: An attempt was made to do one of the following:

- Create a table with more than one ROWID column.
- Add a ROWID column to a table that already has one.
- Create a table with more than one identity column.
- Add an identity column to a table that already has one.

System Action: The statement was not executed.

Programmer Response: For a CREATE TABLE statement, select only one column to have the row ID data type or the AS IDENTITY attribute. For an ALTER TABLE statement, a ROWID column or identity column already exists for the table. Do not attempt to add another column with the data type row ID or with the AS IDENTITY attribute to the table.

SQLSTATE: 428C1

-373 DEFAULT CANNOT BE SPECIFIED FOR IDENTITY COLUMN *column-name*

Explanation: A DEFAULT clause may not be specified for a column that has been identified as an IDENTITY column.

System Action: The statement cannot be executed.

Programmer Response: Remove the DEFAULT clause and resubmit the statement.

SQLSTATE: 42623

-374 THE *clause* **CLAUSE HAS NOT BEEN SPECIFIED IN THE CREATE FUNCTION STATEMENT FOR LANGUAGE SQL FUNCTION** *function-name* **BUT AN EXAMINATION OF THE FUNCTION BODY REVEALS THAT IT SHOULD BE SPECIFIED**

Explanation: The following situations may be the cause of this error:

NOT DETERMINISTIC
must be specified if either of the following conditions apply within the body of the function:

- a function that has the NOT DETERMINISTIC attribute is invoked
- a special register is accessed

READS SQL DATA
must be specified if the body of the function defined with LANGUAGE SQL contains a subselect or if it invokes a function that can read SQL data.

EXTERNAL ACTION
must be specified if the body of the function defined with LANGUAGE SQL invokes a function that has the EXTERNAL ACTION attribute.

System Action: The statement cannot be processed.

Programmer Response: Either specify the clause or change the function body.

SQLSTATE: 428C2

-390 THE FUNCTION *function-name*, **SPECIFIC NAME** *specific-name*, **IS NOT VALID IN THE CONTEXT IN WHICH IT OCCURS**

Explanation: A function resolved to a specific function that is not valid in the context where it is used. If *specific-name* is an empty string, then the function resolved to the built-in function identified by *function-name*. Some of the possible situations include:

- A scalar or column function is referenced where only a table function is allowed (such as in the FROM clause of a query).
- A table function is referenced where only a scalar or column function is allowed (such as in an expression, or in a SOURCE clause of a CREATE FUNCTION statement).

System Action: The statement cannot be executed.

Programmer Response: Ensure that the correct function name and arguments are specified and that the SQL path includes the schema where the correct function is defined. You may need to change the function name, arguments, SQL path (using SET CURRENT PATH or the PATH bind option), or change

the context in which the function is used. Refer to Chapter 6 of *DB2 SQL Reference* for information on the use of functions.

SQLSTATE: 42887

-392 **SQLDA PROVIDED FOR CURSOR** *cursor* **HAS BEEN CHANGED FROM THE PREVIOUS FETCH**

Explanation: The application is running with *DB2 rules*, and has requested that LOB data be returned as a LOB in one FETCH statement, and as a locator in another FETCH statement. This is not permitted.

System Action: The statement cannot be executed.

Programmer Response: Either do not use *DB2 rules*, or change to application to not change the data type code from LOB to locator (or the reverse) in the SQLDA between successive fetches.

SQLSTATE: 42855

-396 *object-type object-name* **ATTEMPTED TO EXECUTE AN SQL STATEMENT DURING FINAL CALL PROCESSING**

Explanation: A user-defined function named *object-name* was invoked and attempted to execute an SQL statement (other than CLOSE CURSOR) during final call processing, but the statement is not allowed.

System Action: The SQL statement cannot be executed.

Programmer Response: Change the definition of the function to not issue SQL statements during final call processing.

SQLSTATE: 38505

-397 **THE OPTION GENERATED IS SPECIFIED WITH A COLUMN THAT IS NOT A ROW ID OR DISTINCT TYPE BASED ON A ROW ID**

Explanation: GENERATED was specified in a CREATE or ALTER TABLE statement for a column with a data type that is not a row ID, and is not a distinct type that is based on a row ID. GENERATED can only be specified for a column with a data type of row ID, or a distinct type that is based on a row ID.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement. Either eliminate the GENERATED clause or ensure that the data type of the object is row ID.

SQLSTATE: 428D3

–398 **A LOCATOR WAS REQUESTED FOR HOST VARIABLE NUMBER** *position-number* **BUT THE VARIABLE IS NOT A LOB**

Explanation: The application requested that a locator be returned from host variable number *position-number*. A locator can only be used with LOB data, and the requested data is not a LOB.

System Action: The statement cannot be executed.

Programmer Response: Change the statement to either return LOB data, or change the target host variable to not be a locator.

SQLSTATE: 428D2

-399 **ATTEMPTED TO INSERT AN INVALID VALUE INTO A ROWID COLUMN**

Explanation: When inserting into a table, a value specified for a ROWID column was invalid. Only row ID values previously generated by DB2 are valid.

System Action: The INSERT is not performed.

Programmer Response: Do not attempt to generate any value for insertion into a ROWID column. Insertion into ROWID columns is supported for purposes of Data Propagation, where DB2 has previously generated the row ID values. Only row ID values previously generated by DB2 can be used as values for insertion into a row ID column. Alternatively, insert the row specifying DEFAULT for the ROWID column or remove the ROWID column from the insert column-list.

You may also use the OVERRIDING clause as a possible solution for this situation. See INSERT in DB2 SQL Reference for more information about the OVERRIDING USER VALUE clause.

SQLSTATE: 22511

-400 **THE CATALOG HAS THE MAXIMUM NUMBER OF USER DEFINED INDEXES**

Explanation: Only one hundred user-defined indexes can be created in the catalog database.

System Action: The statement cannot be executed.

Programmer Response: If this index must be created, another user-defined index on the catalog must be dropped. After that index is dropped, this statement can be executed.

SQLSTATE: 54027

-401 **THE OPERANDS OF AN ARITHMETIC OR COMPARISON OPERATION ARE NOT COMPARABLE**

Explanation: An arithmetic operation appearing within the SQL statement contains a mixture of numeric and

non-numeric operands, or the operands of a comparison operation are not compatible.

One reason for this error is that the comparison involves both character and graphic operands. This combination of operands is not allowed.

System Action: The statement cannot be executed.

Programmer Response: Check the data types of all operands to ensure that their data types are comparable and compatible with their usage in the statement.

If all the operands of the SQL statement are correct, then, if a view is being accessed, check the data types of all the operands in the view definition.

SQLSTATE: 42818

-402 **AN ARITHMETIC FUNCTION OR OPERATOR *arith-fop* IS APPLIED TO CHARACTER OR DATETIME DATA**

Explanation: A nonnumeric operand has been specified for the arithmetic function or operator *arith-fop*.

System Action: The statement cannot be executed.

Programmer Response: Examine and correct the syntax of the SQL statement such that all operands of the specified function or operator are numeric.

SQLSTATE: 42819

-404 **THE SQL STATEMENT SPECIFIES A STRING THAT IS TOO LONG**

Explanation: An INSERT, UPDATE, CALL, VALUES INTO, SET, parameter, host variable, or transition variable statement specifies a value that is longer than the maximum length string that can be stored in the target column.

System Action: The statement cannot be executed.

Programmer Response: Check the length of the target column, parameter, host variable or transition variable and correct the program or SQL statement so that the length of the string does not exceed that maximum. For example, you could use the SUBSTR function to shorten the string.

SQLSTATE: 22001

-405 **THE NUMERIC LITERAL literal CANNOT BE USED AS SPECIFIED BECAUSE IT IS OUT OF RANGE**

Explanation: The specified numeric literal is not in the proper range.

The proper ranges for SQL values are as follows:
- 5.4E−79 to 7.2E+75 for FLOAT values
- $-(10^{31} -1)$ to $+(10^{31} -1)$ for DECIMAL values
- −2147483648 to 2147483647 for INTEGER values
- −32768 to +32767 for small integer (SMALLINT) values.

System Action: The statement cannot be executed.

Programmer Response: The value of the literal should be reduced to the appropriate size for this data type.

SQLSTATE: 42820

-406 **A CALCULATED OR DERIVED NUMERIC VALUE IS NOT WITHIN THE RANGE OF ITS OBJECT COLUMN**

Explanation: A value derived or calculated during processing of the SQL statement was outside the range of the data type of its object column. This problem might have arisen because either the values occurring in the object column were out of range, or the SQL operation performed was not appropriate for the values in the object column.

System Action: The statement cannot be executed.

Programmer Response: See the explanation of SQLCODE -405 for allowed ranges for numeric data types.

SQLSTATE: 22003

-407 **AN UPDATE, INSERT, OR SET VALUE IS NULL, BUT THE OBJECT COLUMN *column-name* CANNOT CONTAIN NULL VALUES**

Explanation: One of the following conditions occurred:
- A null insert or update value was specified for a column defined as NOT NULL.
- No insert value was provided for a column that does not have a default value.
- A SET transition variable statement specified a NULL value for column defined as NOT NULL.
- The insert value was DEFAULT, but the object column was declared as NOT NULL without WITH DEFAULT in the table definition. Consequently, a default value of NULL cannot be inserted into that column.
- A null insert value was specified for a ROWID column.

System Action: The statement cannot be executed. The 'column-name' might be returned in the SQLCA, depending on the syntax of the SQL statement in which the error was detected.

Programmer Response: Examine the definition of the object table to determine which columns of the table have the NOT NULL attribute or have a type of ROWID, and correct the SQL statement accordingly.

SQLSTATE: 23502

-408 THE VALUE IS NOT COMPATIBLE WITH THE DATA TYPE OF ITS TARGET

Explanation: The data type of the value to be assigned to the column, parameter, host variable or transition variable by the SQL statement is incompatible with the declared data type of the assignment target. Both must be:

- Numeric
- Character
- Graphic
- Dates or character
- Times or character
- Timestamps or character
- Row IDs
- The same distinct types

However, dates, times, or timestamps cannot be assigned to a character column that has a field procedure. Also note that character and graphic data types are compatible when using Unicode.

This SQLCODE is issued for any statement that fails required assignment rule checking.

System Action: The statement cannot be executed.

Programmer Response: Examine the current definition for the object table, procedure, user-defined function, or host variable and ensure that the host variable or literal value that is assigned to the object has the proper data type. In some cases, you can convert the value to the proper data type by using a function such as CHAR or DECIMAL.

SQLSTATE: 42821

-409 INVALID OPERAND OF A COUNT FUNCTION

Explanation: The operand of the COUNT or COUNT_BIG function in the statement violates SQL syntax. A common error is a column name or other expression without DISTINCT.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax and resubmit the statement. Refer to Chapter 4 of *DB2 SQL Reference* for information about the proper form for the operands of a COUNT or COUNT_BIG function.

SQLSTATE: 42607

-410 THE FLOATING POINT LITERAL *literal* CONTAINS MORE THAN 30 CHARACTERS

Explanation: The specified floating-point literal is more than 30 characters in length. A floating-point literal has a maximum length of 30 characters.

System Action: The statement cannot be executed.

Programmer Response: Correct the indicated literal.

SQLSTATE: 42820

-411 CURRENT SQLID CANNOT BE USED IN A STATEMENT THAT REFERENCES REMOTE OBJECTS

Explanation: A reference to the CURRENT SQLID special register is invalid in a statement that contains the three-part name or alias of an object that is remote to the remote server.

System Action: The statement cannot be executed.

Programmer Response: Either remove the reference to CURRENT SQLID or the reference to the remote object.

SQLSTATE: 56040

-412 THE SELECT CLAUSE OF A SUBQUERY SPECIFIES MULTIPLE COLUMNS

Explanation: In the context in which it was used in the SQL statement, the subquery can have only one column specified in its SELECT clause.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the SQL statement. Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on the syntax for subqueries.

SQLSTATE: 42823

-413 OVERFLOW OCCURRED DURING NUMERIC DATA TYPE CONVERSION

Explanation: During processing of the SQL statement, an overflow condition arose when converting from one numeric type to another. Numeric conversion is performed according to the standard rules of SQL.

System Action: The statement cannot be processed. No data was retrieved, updated, or deleted.

Programmer Response: Examine the syntax of the SQL statement to determine the cause of the error. If the problem is data-dependent, it might be necessary to examine the data processed at the time of the error.

SQLSTATE: 22003

-414 A LIKE PREDICATE IS INVALID BECAUSE THE FIRST OPERAND IS NOT A STRING

Explanation: The data type of the first operand of the LIKE predicate must be a character string or graphic string.

System Action: The statement cannot be executed.

Programmer Response: Respecify the predicate so that the data type of each operand is a character string or a graphic string.

SQLSTATE: 42824

-415 **THE CORRESPONDING COLUMNS, *column-number*, OF THE OPERANDS OF A UNION OR A UNION ALL DO NOT HAVE COMPARABLE COLUMN DESCRIPTIONS**

Explanation: The column descriptions of corresponding columns of the operands of a UNION or UNION ALL must be comparable. The columns of ordinality 'column-number' of the operands in this UNION or UNION ALL do not satisfy this requirement. For columns to be comparable, they must both be either numeric, character, graphic, date, time, or timestamp. They cannot be a mixture of these groups. If corresponding columns have field procedures, they must both have the same field procedure.

System Action: The statement cannot be executed.

Programmer Response: Check the data types of the specified columns and correct the UNION or UNION ALL statement so that all corresponding columns have comparable column descriptions.

SQLSTATE: 42825

-416 **AN OPERAND OF A UNION CONTAINS A LONG STRING COLUMN**

Explanation: The UNION specified in the SQL statement could not be performed because one of the tables participating in the union contains a long string column (for example, a VARCHAR column with length greater than 255). The operands of a UNION cannot contain long string columns.

System Action: The statement cannot be executed.

Programmer Response: The implied function is not supported by DB2. Refer to Chapter 3 of *DB2 SQL Reference* for information about restrictions on the manipulation of long string columns.

SQLSTATE: 42907

-417 **A STATEMENT STRING TO BE PREPARED INCLUDES PARAMETER MARKERS AS THE OPERANDS OF THE SAME OPERATOR**

Explanation: The statement string specified as the object of a PREPARE contains a predicate or expression where parameter markers have been used as operands of the same operator—for example:

```
? > ?
```

This syntax is not permitted.

System Action: The statement cannot be executed.

Programmer Response: Correct the logic of the application program so that this syntax error does not occur. Refer to Chapter 6 of *DB2 SQL Reference* for information about the proper usage of parameter markers within SQL statements to be prepared.

SQLSTATE: 42609

-418 **A STATEMENT STRING TO BE PREPARED CONTAINS AN INVALID USE OF PARAMETER MARKERS**

Explanation: Parameter markers cannot be used in the SELECT list, as the sole argument of a scalar function, or in a concatenation operation. Parameter markers cannot be used in the string expression of an EXECUTE IMMEDIATE SQL statement.

System Action: The statement cannot be executed.

Programmer Response: Correct the logic of the application program so that this error does not occur. Refer to Chapter 6 of *DB2 SQL Reference* for information about the proper usage of parameter markers within SQL statements and for EXECUTE IMMEDIATE SQL statement restrictions.

SQLSTATE: 42610

-419 **THE DECIMAL DIVIDE OPERATION IS INVALID BECAUSE THE RESULT WOULD HAVE A NEGATIVE SCALE**

Explanation: The decimal division is invalid because it will result in a negative scale.

The formula used internally to calculate the scale of the result for decimal division is explained in Chapter 3 of *DB2 SQL Reference*.

System Action: The statement cannot be executed. No data was retrieved, updated, or deleted.

Programmer Response: Examine the precision and scale of all columns that may have participated in a decimal division. Note that an integer or small integer value may have been converted to decimal for this calculation.

SQLSTATE: 42911

-420 **THE VALUE OF A STRING ARGUMENT WAS NOT ACCEPTABLE TO THE *function-name* FUNCTION**

Explanation: A string argument did not conform to the requirements of the function. For example, a character string passed to the DECIMAL function did not conform to the rules for forming an SQL integer or decimal constant.

System Action: The statement cannot be processed.

Programmer Response: Change the argument value

so that it conforms to the requirements of the function as specified in *DB2 SQL Reference*.

SQLSTATE: 22018

-421 **THE OPERANDS OF A UNION OR UNION ALL DO NOT HAVE THE SAME NUMBER OF COLUMNS**

Explanation: The operands of a UNION or UNION ALL must have the same number of columns.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement so that there are exactly the same number of columns in each operand.

SQLSTATE: 42826

-423 **INVALID VALUE FOR LOCATOR IN POSITION** *position-#*

Explanation: The value specified in a result set locator host variable or a LOB locator host variable specified at position *position-#* in the locator variable list of the SQL statement does not identify a valid result set locator or LOB locator variable, respectively.

System Action: The statement cannot be executed.

Programmer Response: For a result set locator there are two common causes for the error:
- The host variable used as a result set locator was never assigned a valid result set locator value. Result set locator values are returned by the DESCRIBE PROCEDURE and ASSOCIATE LOCATORS statements. Make sure the value in your host variable is obtained from one of these statements.
- Result set locator values are only valid as long as the underlying SQL cursor is open. If a commit or rollback operation closes an SQL cursor, the result set locator associated with the cursor is no longer valid.

For a LOB locator, some common causes for the error are:
- The host variable used as a LOB locator was never assigned a valid LOB value.
- A commit or rollback operation or an SQL FREE LOCATOR statement freed the locator.

SQLSTATE: 0F001

-426 **DYNAMIC COMMIT NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED**

Explanation: An application executing using DRDA protocols has attempted to issue a dynamic COMMIT statement, or a stored procedure has attempted to issue a COMMIT_ON_RETURN, while connected to a location at which updates are not allowed. A dynamic COMMIT or COMMIT_ON_RETURN can be issuedonly while connected to a location at which updates are allowed.

System Action: The statement cannot be executed. No COMMIT is performed.

Programmer Response: The IMS or CICS protocols should be used to commit work in these environments.

SQLSTATE: 2D528

-427 **DYNAMIC ROLLBACK NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED**

Explanation: An application executing using DRDA protocols has attempted to issue a dynamic ROLLBACK statement while connected to a location at which updates are not allowed. A dynamic ROLLBACK may be issued only while connected to a location at which updates are allowed.

System Action: The statement cannot be executed. No ROLLBACK is performed.

Programmer Response: The IMS or CICS protocols should be used to rollback work in these environments.

SQLSTATE: 2D529

-430 *routine-type routine-name* **(SPECIFIC NAME** *specific-name***) HAS ABNORMALLY TERMINATED**

Explanation: An abnormal termination has occurred while the routine *routine-name* (stored procedure or function) was in control.

System Action: The statement cannot be executed.

Programmer Response: The stored procedure or function needs to be fixed. Contact the author of the routine or your database administrator. Until it is fixed, the routine should not be used.

SQLSTATE: 38503

-433 **VALUE** *value* **IS TOO LONG**

Explanation: The value *value* required truncation by a system (built-in) cast or adjustment function, which was called to transform the value in some way. The truncation is not allowed where this value is used. The value being transformed is one of the following:
- an argument to a user defined function (UDF)
- an input to the SET clause of an UPDATE statement
- a value being INSERTed into a table
- an input to a cast or adjustment function in some other context.

If *value* has the 'for bit data' subtype, then the *value* is printed as a hexadecimal string in quotes followed by an X.

System Action: The statement cannot be executed.

Programmer Response: If *value* is a literal string in the SQL statement, it is too long for its intended use. If *value* is not a literal string, examine the SQL statement to determine where the transformation is taking place. Either the input to the transformation is too long, or the target is too short. Correct the problem and rerun the statement.

SQLSTATE: 38xxx

-435 **AN INVALID SQLSTATE** *sqlstate* **IS SPECIFIED IN THE FUNCTION RAISE_ERROR OR IN A SIGNAL SQLSTATE STATEMENT**

Explanation: The SQLSTATE specified in the RAISE_ERROR function or specified in a SIGNAL SQLSTATE statement of a trigger definition does not conform to the rules for an application defined SQLSTATE.

System Action: The statement cannot be processed.

Programmer Response: Correct the SQLSTATE specified in the RAISE_ERROR function or SIGNAL statement. The SQLSTATE must be a character string containing exactly 5 characters. It must be of type CHAR defined with a length of 5, or a type VARCHAR defined with a length of 5 or greater. The SQLSTATE value must follow the rules for application-defined SQLSTATEs as follows:

- Each character must be from the set of digits ('0' through '9') or non-accented upper case letters ('A' through 'Z').
- The SQLSTATE class (first two characters) cannot be '00', '01' or '02' because these are not error classes.
- If the SQLSTATE class (first two characters) starts with the character '0' through '6' or 'A' through 'H', then the subclass (last three characters) must start with a letter in the range 'I' through 'Z'.
- If the SQLSTATE class (first two characters) starts with the character '7', '8', '9' or 'I' though 'Z', then the subclass (last three characters) can be any of '0' through '9' or 'A' through 'Z'.

SQLSTATE: 428B3

-438 **APPLICATION RAISED ERROR WITH DIAGNOSTIC TEXT:** *text*

Explanation: This error occurred as a result of execution of the RAISE_ERROR function or as a result of the SIGNAL SQLSTATE statement.

text Diagnostic text provided by the invocation of the RAISE_ERROR function or the SIGNAL SQLSTATE statement.

System Action: The statement cannot be processed.

Programmer Response: Use application-provided diagnostic text, if provided, to determine the cause of the error.

SQLSTATE: application-defined

-440 **NO** *routine-type* **BY THE NAME** *routine-name* **HAVING COMPATIBLE ARGUMENTS WAS FOUND**

Explanation: This occurs in a reference to routine (stored procedure or function) *routine-name*, when DB2 cannot find a function or stored procedure it can use to implement the reference. There are several reasons why this could occur.

- *routine-name* was either incorrectly specified or does not exist in the database.
- A qualified reference was made, and the qualifier was incorrectly spelled.
- A user's current path does not contain the schema to which the desired function belongs, and an unqualified reference was used.
- The wrong number of arguments were included.
- For functions, the data types of one or more of the arguments is incorrect.
- The routine invoker is not authorized to execute the routine.

System Action: The statement cannot be executed.

Programmer Response: Fix the problem and retry. This could involve a change to the SQL statement, the addition of new routines or a change to the user's current path.

SQLSTATE: 42884

-441 **INVALID USE OF 'DISTINCT' OR 'ALL' WITH SCALAR FUNCTION** *function-name*

Explanation: The keyword 'DISTINCT' or 'ALL' was detected within parentheses in a reference to function *function-name* and the function has been resolved as a scalar function. Use of the keyword 'DISTINCT' or 'ALL' with a scalar function is invalid.

System Action: The statement cannot be executed.

Programmer Response: If a scalar function is being used, then remove the keyword 'DISTINCT' or 'ALL'.

If a column function is being used, then there is a problem with function resolution. Check your current path to see if the desired function is in one of the schemas, and also check the SYSIBM.SYSROUTINES catalog for the spelling of the function name and the number and types of parameters.

SQLSTATE: 42601

-443 **EXTERNAL FUNCTION** *function-name* **(SPECIFIC NAME** *specific-name***) HAS RETURNED AN ERROR SQLSTATE WITH DIAGNOSTIC TEXT** *msg-text*

Explanation: An SQLSTATE of the form 38xxx was returned by function *function-name*, along with message text *msg-text*.

If the third character is not 5 (i.e. '385xx') then the last 3 characters of the SQLSTATE value were chosen by the function, to indicate the reason of the failure. SQLSTATEs values of the form 385xx are issued by IBM with a different SQLCODE.

System Action: The actions in the external function should be rolled back.

Programmer Response: Contact the author of the function or your database administrator. Until the problem is resolved, the function should not be used.

SQLSTATE: 42601

-444 **USER PROGRAM** *name* **COULD NOT BE FOUND**

Explanation: DB2 received an SQL CALL statement for a stored procedure or an SQL statement containing an invocation of a user-defined function, and found the row in the SYSIBM.SYSROUTINES catalog table associated with the requested procedure name. However, the MVS load module identified in the LOADMOD column of the SYSIBM.SYSROUTINES row could not be found.

name The name of the MVS load module that could not be found

System Action: The statement cannot be executed.

Programmer Response: If the LOADMOD column value in the SYSIBM.SYSROUTINES table is incorrect, use the ALTER FUNCTION or ALTER PROCEDURE statement to correct the value.

If the LOADMOD column value is correct, use the MVS linkage editor to create the required MVS load module in one of the MVS load libraries used by your installation for stored procedures.

This error can also occur if you are invoking a WLM-managed stored procedure that is not APF authorized, and the DB2 load libraries are not in the STEPLIB concatenation because they are being loaded from LINKLIST. In this case, if you want the stored procedure program to run APF-authorized, link-edit it with AC=1 into an MVS APF authorized library. If you do not want the stored procedure program to run APF authorized, add the DB2 load library to the STEPLIB concatenation of the JCL used to start the WLM-managed address space.

SQLSTATE: 42724

-449 **CREATE OR ALTER STATEMENT FOR FUNCTION OR PROCEDURE** *routine-name* **CONTAINS AN INVALID FORMAT OF THE EXTERNAL NAME CLAUSE OR IS MISSING THE EXTERNAL NAME CLAUSE**

Explanation: An error was found in the EXTERNAL NAME clause of the CREATE FUNCTION, CREATE PROCEDURE, ALTER FUNCTION, or ALTER PROCEDURE statement for *routine-name*, or the clause is needed but was not specified.

- For a LANGUAGE JAVA or COMPJAVA stored procedure, or LANGUAGE JAVA user-defined function, the name must be specified and it must contain a valid *external-java-routine-name* of the following form:

  jar-name:*package-id*...*class-id*.*method-id*(*method-signature*)

  - Do not include blanks.
  - For LANGUAGE COMPJAVA, do not specify a *jar-name*.
  - The *method-name* consists of the list of *package-ids*, *class-id*, and *method-id*, and must not be longer than 254 bytes.
  - For LANGUAGE JAVA, zero or more *package-ids* may be specified preceding the *class-id*
  - For LANGUAGE COMPJAVA,
  - The *method-signature* is optional, and is a list of Java data types that are separated by commas. If specified, the *method-signature* must not be longer than 1024 bytes.
  - If multiple *string*s are specified, the total length of all the *string*s concatenated together for the *external-java-routine-name* must not be greater than 1305.
- For external routines with a language other than JAVA or COMPJAVA, the external name must be a short identifier with letters or digits. The first character must be a letter. (This is the MVS naming convention for load modules). A possible cause for this error is the inclusion of a blank in the name.

  If the clause is omitted, the external name defaults to *function-name*. However, for CREATE FUNCTION or CREATE PROCEDURE if the function or procedure name is longer than eight characters then the EXTERNAL NAME clause must be explicitly specified to specify a valid *short identifier* as the external name.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the SQL statement. Refer to the *DB2 SQL Reference* for information on the EXTERNAL NAME clause.

User Response: When LANGUAGE is JAVA or COMPJAVA, possible causes include:

- Omitting hte EXTERNAL NAME clause.

- Including a blank.
- Having the '!' at the beginning or end of the name.
- Specifying an invalid *external-java-routine-name*.

SQLSTATE: 42878

-450 **USER-DEFINED FUNCTION OR STORED PROCEDURE** *name*, **PARAMETER NUMBER** *parmnum*, **OVERLAYED STORAGE BEYOND ITS DECLARED LENGTH.**

Explanation: Upon return from a specific function *name* or a stored procedure *name*, DB2 has detected an overlay storage beyond a parameter's declared length. The parameter number is specified for a stored procedure or function. This is not permitted.

System Action: The statement cannot be executed.

Programmer Response: Contact the author of the function or your database administrator. Until it is fixed, the function should not be used.

SQLSTATE: 39501

-451 **THE** *data-item* **DEFINITION, IN THE CREATE FUNCTION FOR** *function-name* **CONTAINS DATA TYPE** *type* **WHICH IS NOT APPROPRIATE FOR AN EXTERNAL FUNCTION WRITTEN IN THE GIVEN LANGUAGE**

Explanation: An error was made in the *data-item* part of the CREATE FUNCTION statement for *function-name*. The CREATE FUNCTION statement contained an invalid *type*, or it contained a distinct type which is based on the invalid *type.*

System Action: The statement cannot be executed.

Programmer Response: Correct the statement.

SQLSTATE: 42815

-453 **THERE IS A PROBLEM WITH THE RETURNS CLAUSE IN THE CREATE FUNCTION STATEMENT FOR** *function-name*

Explanation: A problem casting the result of user-defined function *function-name* has been identified. The CAST FROM data type is not castable to the RETURNS data type, and it must be. See the *SQL Reference* for details on casting between data types.

System Action: The statement cannot be executed.

Programmer Response: Change the RETURNS or CAST FROM clause so that the CAST FROM data type is castable to the RETURNS data type.

SQLSTATE: 42880

-454 **THE SIGNATURE PROVIDED IN THE CREATE FUNCTION STATEMENT FOR** *function-name* **MATCHES THE SIGNATURE OF SOME OTHER FUNCTION ALREADY EXISTING IN THE SCHEMA**

Explanation: The signature consists of the function name (*function-name*), the number of parameters defined for the function, and an ordered list of the types of the parameters (without regard to any parameters of the types). In this case there is a function already in the schema and the existing function has the same signature as the function being created. See the SQL Reference for the details on the uniqueness of a function.

System Action: The statement cannot be executed.

Programmer Response: Determine if the existing function already provides the functionality desired. If not, then the new function's signature will have to be changed (e.g. change the function name).

SQLSTATE: 42723

-455 **IN CREATE FUNCTION FOR** *function-name*, **THE SCHEMA NAME** *schema-name1* **PROVIDED FOR THE SPECIFIC NAME DOES NOT MATCH THE SCHEMA NAME** *schema-name2* **OF THE FUNCTION**

Explanation: If the SPECIFIC name is specified as a two part name, the *schema-name1* portion must be the same as the *schema-name2* portion of the *function-name*. Note that the *schema-name2* portion of *function-name* may have been specified directly or it may have defaulted to the authorization ID of the statement.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement.

SQLSTATE: 42882

-456 **IN CREATE FUNCTION FOR** *function-name*, **THE SPECIFIC NAME** *specific-name* **ALREADY EXISTS IN THE SCHEMA**

Explanation: A SPECIFIC name has been explicitly specified as *specific-name* in the CREATE FUNCTION statement for *function-name*, but this name already exists as the SPECIFIC name for another function within the schema.

System Action: The statement cannot be executed.

Programmer Response: Choose a new SPECIFIC name.

SQLSTATE: 42710

-457 **A FUNCTION OR DISTINCT TYPE CANNOT BE CALLED *name* SINCE IT IS RESERVED FOR SYSTEM USE**

Explanation: The user-defined function or distinct type cannot be created or referenced because the name selected is reserved for use by the system.

A number of names used as keywords are reserved for system use. These names may not be used as user-defined functions or distinct-type-names, **even if they are delimited identifiers**. These names are:

```
=         <         >         >=        <=
       ¬=        ¬>        ¬>
ALL       AND       ANY       BETWEEN   DISTINCT
EXCEPT    EXISTS    FALSE     FOR       FROM
IN        IS        LIKE      MATCH     NOT
NULL      ONLY      OR        OVERLAPS  SIMILAR
SOME      TABLE     TRUE      TYPE      UNIQUE
UNKNOWN
```

The names of built-in data types cannot be used as the name of a distinct type (for example, CHAR).

System Action: The statement is not executed.

Programmer Response: Select a name for the function or distinct type that is not reserved for system use.

SQLSTATE: 42939

-458 **IN A REFERENCE TO FUNCTION *function-name* BY SIGNATURE, A MATCHING FUNCTION COULD NOT BE FOUND**

Explanation: In a reference to function *function-name* by signature, no matching function could be found. The problem could be with the data type or some other attributes of a parameter. For some data types there are attributes in addition to data type:

- Length, precision, or scale

  While it is not necessary to specify a length, precision, or scale attribute for a data type, if one is specified then there must be an exact match on the corresponding specification of the parameter for the existing function as defined in SYSPARMS.

  A type of FLOAT(n) does not need to match the defined value for n since 1<=n<=21 means REAL and 22<=n<=53 means DOUBLE. Matching occurs based on whether the type is REAL or DOUBLE.

  However, a match on data type is sufficient.

  To indicate this, an empty set of parentheses must be specified for the data types that allow a specification of length, precision, or scale. For example, assuming a function exists for which a parameter was defined as CHAR(12) on the CREATE FUNCTION statement, a reference to that function by a signature could specify this parameter as either CHAR(12), or CHAR(). The CHAR() syntax provides a way to say ″don't care about length, precision and scale attributes in finding a matching function″.

- FLOAT() cannot be used since the parameter value indicates different data types (REAL or DOUBLE).

  If, however, neither length, presision, scale, or empty parenthesis were specified, then normal default rules apply. For example, a specification of CHAR would result in CHAR(1) as on the CREATE TABLE statement. Furthermore, this implicit specification of length, precision, or scale must exactly match the corresponding specification of the parameter for the existing function as defined in SYSPARMS.

  – Subtype, or encoding scheme

    You do not need to specify the subtype or encoding scheme (CCSID clause) to identify an existing function in the database. However, if a subtype or encoding scheme is specified then there there must be an exact match on the corresponding specification of the parameter for the existing function as defined in SYSPARMS.

- Unqualified function names:
  – For ALTER FUNCTION, DROP FUNCTION, COMMENT ON FUNCTION, GRANT and REVOKE statements for EXECUTE on functions, an unqualified function name is implicitly qualified with the statement authorization ID, and this is the schema where the function with the problem can be found.
  – In the SOURCE clause of a CREATE FUNCTION statement, the qualification comes from the SQL path. In this case, the is no matching function in the entire path.

    Note: A function cannot be sourced on the COALESCE, NULLIF, RAISE_ERROR, or VALUE built-in functions. Additionally, there are restrictions on the way that you can source on the COUNT, COUNT_BIG, CHAR, and STRIP built-in functions because of some of the keywords that they accept.

System Action: The statement cannot be executed.

Programmer Response: Possible responses include:
- Changing the SQL path to include the correct schema.
- Changing the attibutes of the parameters.
- Using a SPECIFIC name to refer to the function instead of a signature.

SQLSTATE: 42883

-461 **A VALUE WITH DATA TYPE *source-data-type* CANNOT BE CAST TO TYPE *target-data-type***

Explanation: The statement contains a CAST with the first operand having a data type of *source-data-type* to

be cast to the data type *target-data-type*. This is not supported.

Change the data type of either the source or target so that the cast is supported. For predefined (built-in) data types or a cast involving a user-defined distinct type, see the SQL Reference.

System Action: The statement could not be processed.

Programmer Response: Correct the CAST specification to specify a supported combination of source and target types.

SQLSTATE: 42846

-463 **EXTERNAL ROUTINE** *routine-name* **(SPECIFIC NAME** *specific-name***) HAS RETURNED AN INVALID SQLSTATE** *sqlstate***, WITH DIAGNOSTIC TEXT** *text*

Explanation: The valid SQLSTATEs that a user-defined function or stored procedure can return are 38xxx (error), 38502 (error) and 01Hxx (warning). User-defined function or stored procedure *routine-name* returned an invalid SQLSTATE *sqlstate*, along with message text *text*. The user-defined function or stored procedure is in error.

System Action: The statement could not be executed.

Programmer Response: The user-defined function or stored procedure needs to be corrected. See your database administrator, or the author of the function to find out the meaning of the warning. The significance of the bad SQLSTATE to the invoking application can be learned from the author of the function.

SQLSTATE: 39001

-469 **SQL CALL STATEMENT MUST SPECIFY AN OUTPUT HOST VARIABLE FOR PARAMETER** *number*

Explanation: DB2 received an SQL CALL statement for a stored procedure. DB2 found the row in the SYSIBM.SYSROUTINES catalog table associated with the requested procedure name. However, parameter *number* was identified in the SYSIBM.SYSPARMS table as an OUT or INOUT parameter. A host variable must be supplied on the SQL CALL statement for parameters defined as OUT or INOUT.

number
The parameter number from the ORDINAL field in SYSIBM.SYSPARMS.

System Action: The statement cannot be executed.

Programmer Response: If the SQL CALL statement is coded incorrectly, modify the SQL application to provide an output host variable on the SQL CALL statement.

If the SYSIBM.SYSPARMS table contains incorrect information, the DROP PROCEDURE and CREATE PROCEDURE statements must be used to replace the catalog definition for the stored procedure.

SQLSTATE: 42886

-470 **SQL CALL STATEMENT SPECIFIED A NULL VALUE FOR INPUT PARAMETER** *number***, BUT THE STORED PROCEDURE DOES NOT SUPPORT NULL VALUES.**

Explanation: DB2 received an SQL CALL statement for a stored procedure and found a null value in the incoming parameter list. The stored procedure was defined in the SYSIBM.SYSROUTINES catalog table with PARAMETER_STYLE of GENERAL, which specifies that the routine does not accept null values.

A call to a stored procedure with a LANGUAGE value of COMPJAVA receives this SQLCODE if an input parameter in the compiled Java stored procedure has a Java base type that cannot be set to a null value.

number
The parameter number from the ORDINAL field in SYSIBM.SYSPARMS.

System Action: The statement cannot be executed.

Programmer Response: If the stored procedure should not accept null values, change the calling application to provide a nonnull value.

If the stored procedure should accept null values, use the ALTER PROCEDURE statement to change the PARAMETER STYLE of the stored procedure to be DB2SQL or GENERAL WITH NULLS.

SQLSTATE: 39004

-471 **INVOCATION OF FUNCTION OR PROCEDURE** *name* **FAILED DUE TO REASON** *rc*

Explanation: A routine was invoked. The routine invocation was not accepted because of DB2 reason code *rc*.

name The name of the routine that was invoked.

rc The DB2 reason code describing the cause of the failure.

System Action: The statement cannot be executed. A DSNX9xx message describing the error might be displayed on the MVS system console.

Programmer Response: Correct the condition described by the DB2 reason code.

SQLSTATE: 55023

-472 **CURSOR** *cursor-name* **WAS LEFT OPEN BY EXTERNAL FUNCTION** *function-name* **(SPECIFIC NAME** *specific-name***)**

Explanation: The function program did not close the specified cursor. Modify the function program so that it closes the cursor.

System Action: The statement cannot be executed.

Programmer Response: Reissue the statement when desired.

SQLSTATE: 24517

-473 **A USER DEFINED DATA TYPE CANNOT BE CALLED THE SAME NAME AS A SYSTEM PREDEFINED TYPE (BUILT-IN TYPE)**

Explanation: The name of a data type to be created has an unqualified name that is the same as a system-predefined data type or is BOOLEAN. This is not allowed. Adding delimiters does not make the name valid. The following names are restricted:

```
   BLOB        CHAR
CHARACTER  CLOB       DATE       DBCLOB
DEC        DECIMAL    DOUBLE     FLOAT       GRAPHIC
INT        INTEGER    NUMERIC    REAL
ROWID      SMALLINT   TIME       TIMESTAMP   VARCHAR
VARGRAPHIC
```

System Action: The statement could not be processed.

Programmer Response: Correct the statement to use another identifier for the name of the new user-defined type.

SQLSTATE: 42918

-475 **THE RESULT TYPE** *type-1* **OF THE SOURCE FUNCTION CANNOT BE CAST TO THE RETURNS TYPE** *type-2* **OF THE USER-DEFINED FUNCTION** *function-name*

Explanation: In order for the CREATE FUNCTION for a sourced user-defined function to be valid, the result type (*type-1*) of the source function must be castable to the RETURNS type (*type-2*) of the function being created. There is no supported cast between these data types. See the *DB2 SQL Reference* for details on casting between data types.

System Action: The statement cannot be executed.

Programmer Response: Change the RETURNS data type or the SOURCE function identified so that the result type of the SOURCE function is castable to the RETURNS data type.

SQLSTATE: 42866

-476 **REFERENCE TO FUNCTION** *function-name* **WAS NAMED WITHOUT A SIGNATURE, BUT THE FUNCTION IS NOT UNIQUE WITHIN ITS SCHEMA**

Explanation: References to a function without a signature are permitted, but the named function *function-name* must be unique in its schema and it is not.

Note also that in the ALTER FUNCTION, DROP FUNCTION, COMMENT ON FUNCTION, GRANT and REVOKE statements for EXECUTE on functions, an unqualified reference is qualified with the statement authorization ID, and this is the schema where the problem can be found. In the SOURCE clause of a CREATE FUNCTION statement, the qualification comes from the CURRENT PATH. In this case, the first schema in the path containing a function with this name had other functions by the same name.

System Action: The statement cannot be executed.

Programmer Response: Correct the reference by one of the following:
- completing the signature
- using the SPECIFIC name of the desired function
- changing the CURRENT PATH

SQLSTATE: 42725

-478 **DROP OR REVOKE ON OBJECT TYPE** *type1* **CANNOT BE PROCESSED BECAUSE OBJECT** *name* **OF TYPE** *type2* **IS DEPENDENT ON IT**

Explanation: The requested action cannot be processed because a dependency exists on this *type1*. *type2* is the type of object that has the dependency on the *type1* object involved in the DROP or REVOKE. *type2* can be one of the following:
- FUNCTION
- PROCEDURE
- TABLE
- VIEW
- TRIGGER (for the trigger package)
- CHECK CONSTRAINT (*object_name* contains the table name)
- DEFAULT (*object_name* contains the table name)

DROP If *type1* is *FUNCTION*, the dependencies for DROP might be:
- Another function is sourced on this function.
- A view uses this function.
- A trigger package uses this function.
- A table uses this function in a check constraint or user-defined default.

The dependency might be on one of the generated *cast functions* for a distinct type. If *type1* is *DISTINCT TYPE*, the dependencies for DROP might be:

- A parameter of a function is defined as this distinct type.
- A column of a table is defined as this distinct type.
- A parameter of a stored procedure is defined as this distinct type.

If *type1* is *PROCEDURE*, the dependencies for DROP might be:

- A trigger definition contains a CALL statement with the name of this stored procedure

REVOKE

If *type1* is *FUNCTION*, the dependencies for REVOKE might be:

- A function owned by the revokee is sourced on this function.
- A view owned by the revokee uses this function.
- A trigger package owned by the revokee uses this function.
- A table owned by the revokee uses this function in a check constraint or user-defined default.

If *type1* is *DISTINCT TYPE*, the dependencies for REVOKE might be:

- A parameter of a function owned by the revokee is defined as this distinct type.
- A column of a table owned by the revokee is defined as this distinct type.
- A parameter of a stored procedure owned by the revokee is defined as this distinct type.

If *type1* is *PROCEDURE*, the dependencies for REVOKE might be:

- A trigger definition owned by the revokee contains a CALL statement with the name of this stored procedure.

This SQLCODE may also be issued when SYSADM is being revoked. When SYSADM is revoked the cascading of the REVOKE statement may encounter dependencies that prevent the REVOKE from being successfully processed.

System Action: The statement cannot be executed.

Programmer Response: Remove the dependencies on this object and then reissue the request.

SQLSTATE: 42893

-480 THE PROCEDURE *procedure-name* **HAS NOT YET BEEN CALLED**

Explanation: The procedure identified in a DESCRIBE PROCEDURE or an ASSOCIATE LOCATORS statement has not yet been called within the application process or the procedure has been called, but an explicit or implicit commit occurred before the statement.

System Action: The statement cannot be executed.

Programmer Response: Correct the statements so that the exact syntax used to specify the procedure name on the CALL statement be the same as that on the ASSOCIATE LOCATOR and/or DESCRIBE PROCEDURE. If an unqualified name is used to CALL the procedure, the 1-part name must also be used on the other statements. If the CALL statement is made with a 3-part name, and the current server is the same as the location in the 3-part name, the ASSOCIATE LOCATOR or DESCRIBE procedure can omit the location. Rerun the statements.

SQLSTATE: 51030

-482 THE PROCEDURE *procedure-name* **RETURNED NO LOCATORS**

Explanation: The procedure identified in an ASSOCIATE LOCATORS statement returned no result set locators.

System Action: The statement cannot be executed.

Programmer Response: Determine if result set locators are returned from the identified procedure by using the DESCRIBE PROCEDURE statement.

SQLSTATE: 51030

-483 IN CREATE FUNCTION FOR *function-name* **STATEMENT, THE NUMBER OF PARAMETERS DOES NOT MATCH THE NUMBER OF PARAMETERS OF THE SOURCE FUNCTION**

Explanation: An attempt is being made to CREATE a user-defined function *function-name* which is sourced on another function. One of the following situations has been identified:

- The SOURCE clause uses a function-name (input parameter list) to identify the source function, and the number of types in the list is different from the number of parameters of the function being created.
- The SOURCE clause uses different syntax to identify the source function, and the number of types of that function is different from the number of parameters of the function being created.

System Action: The statement cannot be executed.

Programmer Response: The number of parameters for the SOURCE function and for the function being

created must be the same. The identification of the SOURCE function needs to be changed to:
- fix the input parameter list
- correct the function name or function specific name to identify the proper function.

It is also possible that the current path needs to be corrected in order for correct function resolution to occur.

SQLSTATE: 42885

-487 *object-type object-name* **ATTEMPTED TO EXECUTE AN SQL STATEMENT WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION**

Explanation: A user-defined function or stored procedure *object-name* was invoked and attempted to execute SQL statements, but the function or procedure was created with the NO SQL option.

In an environment of nested functions and procedures, the SQL option in effect is the most restrictive one that has been specified in the nested hierarchy of functions an procedures. The SQL data access option in effect does not allow for modifying data.

System Action: The SQL statement cannot be executed.

Programmer Response: Either use an ALTER statement to change the definition of the function or procedure to allow SQL statements, or remove the failing SQL statement from the external function or procedure.

SQLSTATE: 38001

-490 **NUMBER** *number* **DIRECTLY SPECIFIED IN AN SQL STATEMENT IS OUTSIDE THE RANGE OF ALLOWABLE VALUES IN THIS CONTEXT (***minval***,** *maxval***)**

Explanation: A number (*number*) was specified that is not valid in the context in which it was specified. The minimum allowed value in this context is *minval*. The maximum allowed value in this context is *maxval*. *n* must be within the range specified by *minval* and *maxval* (*minval* =< *n* =< *maxval*).

System Action: The statement was not executed.

Programmer Response: Change the value *n* to a valid value in the statement.

SQLSTATE: 428B7

-491 **CREATE STATEMENT FOR USER-DEFINED FUNCTION** *function-name* **MUST HAVE A RETURNS CLAUSE AND: THE EXTERNAL CLAUSE WITH OTHER REQUIRED KEYWORDS; THE RETURN STATEMENT AND PARAMETER NAMES; OR THE SOURCE CLAUSE**

Explanation: A required clause is missing in the CREATE for function *function-name*.
- For an EXTERNAL function, specify EXTERNAL and one of the following:
  - LANGUAGE
  - PARAMETER STYLE
- For an SQL FUNCTION, specify a RETURN statement, and a parameter name for each parameter.
- For a user-defined SOURCE FUNCTION, specify the SOURCE class.

System Action: The statement cannot be processed.

Programmer Response: Add the missing clauses or statement, and reissue the failing statement.

SQLSTATE: 42601

-492 **THE CREATE FUNCTION FOR** *function-name* **HAS A PROBLEM WITH PARAMETER NUMBER** *number***. IT MAY INVOLVE A MISMATCH WITH A SOURCE FUNCTION**

Explanation: The parameter in position *number* of function *function-name* is in error. The parameter in position *number* of the source function is not castable to the corresponding parameter of the function being created.

If the parameter of the function being created is a *table parameter* then the corresponding parameter of the source function must also be a *table parameter*. Furthermore, the column numbers for both of the table parameters must be the same.

If the parameter of the function being created is not a table parameter then the corresponding parameter of the source function must also not be a table parameter.

System Action: The statement cannot be executed.

Programmer Response: Possible corrections include:
- Identify a different source function.
- Change the data type of the parameter of the function being created so that the data type of the source function can be cast to this data type.

SQLSTATE: 42879

-495 **ESTIMATED PROCESSOR COST OF** *estimate-amount1* **PROCESSOR SECONDS (***estimate-amount2* **SERVICE UNITS) IN COST CATEGORY** *cost-category* **EXCEEDS A RESOURCE LIMIT ERROR THRESHOLD OF** *limit-amount* **SERVICE UNITS**

Explanation: The prepare of a dynamic INSERT, UPDATE, DELETE, or SELECT SQL statement resulted in a cost estimate that exceeded the error threshold value specified in the resource limit specification table (RLST). This error is also issued if DB2's cost category value was "B", and the default action specified in the RLF_CATEGORY_B column in the RLST is to issue an error.

estimate_amount1
The cost estimate (in processor seconds) if the prepared INSERT, UPDATE, DELETE or SELECT statement were to be executed.

estimate_amount2
The cost estimate (in service units) if the prepared INSERT, UPDATE, DELETE or SELECT statement were to be executed.

cost-category
DB2's cost-category for this SQL statement. The possible values are A or B.

limit-amount
The error threshold (in service units) specified in the RLFASUERR column of the RLST.

System Action: The prepare of the dynamic INSERT, UPDATE, DELETE, or SELECT statement was unsuccessful.

Programmer Response: If this SQLCODE was returned because the cost category value is "B", it might be that the statement is using parameter markers or that some statistics are not available for the referenced tables and columns. Make sure the administrator has run the utility RUNSTATS on the referenced tables. It might also be that UDFs will be invoked when the statement is executed, or for INSERT, UPDATE, or DELETE statements that triggers are defined on the changed table. Check the DSN_STATEMNT_TABLE or the IFCID 22 record for this statement to find the reasons this SQL statement has been put in cost category "B". If the program cannot be changed, or if statistics cannot be obtained, ask the administrator to change the value in the RLF_CATEGORY_B column in the RLST to "Y" which allows the statement to execute or "W" which returns a warning instead of an error.

User Response: If the warning is caused by an SQL statement that is consuming too much processor resource, attempt to rewrite the statement to perform more efficiently. Another option is to ask the administrator to increase the error threshold value in the RLST.

SQLSTATE: 57051

-496 **THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT REFERENCES A RESULT SET THAT WAS NOT CREATED BY THE CURRENT SERVER**

Explanation: The SQL statement cannot be executed because the current server is different from the server that called a stored procedure. The SQL statement can be any of the following:
- ALLOCATE CURSOR
- DESCRIBE CURSOR
- FETCH (using an allocated cursor)
- CLOSE (using an allocated cursor)

System Action: The statement cannot be executed.

Programmer Response: Connect to the server that called the stored procedure which created the result set before running the SQL statement that failed.

SQLSTATE: 51033

-497 **THE MAXIMUM LIMIT OF INTERNAL IDENTIFIERS HAS BEEN EXCEEDED FOR DATABASE** *database-name*

Explanation: The SQL statement cannot be executed because an internal identifier limit has been exceeded for the database. The cause of this error is due to one of the following: (1) On a CREATE DATABASE statement, the limit of 65279 DBIDs has been exceeded. (2) For all other statements, the limit of 65279 OBIDs has been exceeded for that database.

System Action: The SQL statement cannot be executed.

Programmer Response: (1) In the case of a DBID limit being exceeded, DROP all unused databases and issue a COMMIT. (2) In the case of an OBID limit being exceeded, DROP all unused objects in the database and issue a COMMIT, specify a different database or run the MODIFY utility to reclaim unused OBIDs.

SQLSTATE: 54041

-499 **CURSOR** *cursor-name* **HAS ALREADY BEEN ASSIGNED TO THIS OR ANOTHER RESULT SET FROM PROCEDURE** *procedure-name***.**

Explanation: An attempt was made to assign a cursor to a result set using the SQL statement ALLOCATE CURSOR and one of the following applies:
- The result set locator variable specified in the ALLOCATE CURSOR statement has been previously assigned to cursor *cursor-name*.
- Cursor *cursor-name* specified in the ALLOCATE CURSOR statement has been previously assigned to a result set from stored procedure *procedure-name*.

System Action: The statement cannot be executed.

Programmer Response: Determine if the target result set named in the ALLOCATE CURSOR statement has been previously assigned to a cursor.

If the result set has been previously assigned to cursor *cursor-name*, then either choose another target result set or call stored procedure *procedure-name* again and reissue the ASSOCIATE LOCATOR and ALLOCATE CURSOR statements.

If the result set has not been previously assigned to a cursor, the cursor *cursor-name* specified in the ALLOCATE CURSOR statement has been previously assigned to some result set from stored procedure *procedure-name*. You can not assign cursor *cursor-name* to another result set, so you must specify a different cursor name in the ALLOCATE CURSOR statement.

Correct the statements so that the exact syntax used to specify the procedure name on the CALL statement be the same as that on the ASSOCIATE LOCATOR and/or DESCRIBE PROCEDURE. If an unqualified name is used to CALL the procedure, the 1-part name must also be used on the other statements. If the CALL statement is made with a 3-part name, and the current server is the same as the location in the 3-part name, the ASSOCIATE LOCATOR or DESCRIBE procedure can omit the location.

SQLSTATE: 24516

-500 THE IDENTIFIED CURSOR WAS CLOSED WHEN THE CONNECTION WAS DESTROYED

Explanation: The FETCH, UPDATE, DELETE, or CLOSE statement identifies a closed cursor that was defined with the WITH HOLD option. The cursor was closed when the connection on which it was dependent was destroyed during a commit operation. The connection was destroyed because the application process placed it in the released state, or the application plan was bound with the DISCONNECT(AUTOMATIC) option.

System Action: The statement cannot be executed.

Programmer Response: The correction depends on the desired state of both the cursor and the connection, as follows:

- If you want the cursor closed, change the application program so that the cursor is not referenced in the closed state.
- If you want the cursor open and the connection was placed in the released state by the application program, change the program so that the connection is not placed in the released state until the cursor is explicitly closed.
- If you want the cursor open and the connection was placed in the released state as a result of the DISCONNECT(AUTOMATIC) option, rebind the plan using DISCONNECT(CONDITIONAL).

Correct the error in the application, rebind the plan, and resubmit the job.

SQLSTATE: 24501

-501 THE CURSOR IDENTIFIED IN A FETCH OR CLOSE STATEMENT IS NOT OPEN

Explanation: The application program attempted to either:

1. FETCH using a cursor, or
2. CLOSE a cursor

at a time when the specified cursor was not open.

System Action: The statement cannot be executed.

Programmer Response: Check for a previous SQL return code that may have closed the cursor. Commit and rollback operations close cursors. SQLCODES -404, -652, -679, -802, -901, -904, -909, -910, -911, and -913 will force the cursor to close. After the cursor is closed, any fetches or close cursor statements will receive this SQLCODE -501.

If no previous SQL return codes have been issued, correct the logic of the application program to ensure that the cursor is open at the time the FETCH or CLOSE statement is executed.

SQLSTATE: 24501

-502 THE CURSOR IDENTIFIED IN AN OPEN STATEMENT IS ALREADY OPEN

Explanation: The application program attempted to execute an OPEN statement for a cursor that was already open.

System Action: The statement cannot be executed. The cursor was unchanged (that is, it was not 'reopened').

Programmer Response: Correct the logic of the application program to ensure that it does not attempt to execute an OPEN statement for a cursor that is already open.

SQLSTATE: 24502

-503 A COLUMN CANNOT BE UPDATED BECAUSE IT IS NOT IDENTIFIED IN THE UPDATE CLAUSE OF THE SELECT STATEMENT OF THE CURSOR

Explanation: The application program attempted to update (using a cursor) a value in a column of the object table that was not identified in the FOR UPDATE clause in the cursor declaration.

Any column that is to be updated must be identified in the FOR UPDATE clause of the cursor declaration.

System Action: The statement cannot be executed. No data was updated in the object table.

Programmer Response: Correct the application program. If the column is to be updated, its name must be added to the FOR UPDATE clause of the cursor declaration.

SQLSTATE: 42912

-504 THE CURSOR NAME *cursor-name* IS NOT DEFINED

Explanation: Cursor *cursor-name* was referenced in an SQL statement, and one of the following is true:

- Cursor *cursor-name* was not declared (using the DECLARE CURSOR statement) or allocated (using the ALLOCATE CURSOR statement) in the application program before it was referenced.
- Cursor *cursor-name* was referenced in a positioned UPDATE or DELETE statement which is not a supported operation for an allocated cursor.
- Cursor *cursor-name* was allocated, but a CLOSE cursor statement naming *cursor-name* was issued and deallocated the cursor before this cursor reference.
- Cursor *cursor-name* was allocated, but a ROLLBACK operation occurred and deallocated the cursor before this cursor reference.
- Cursor *cursor-name* was allocated, but its associated cursor declared in a stored procedure was not declared WITH HOLD, and a COMMIT operation occurred and deallocated the cursor before this cursor reference. The COMMIT operation can be either explicit (the COMMIT statement) or implicit (that is, a stored procedure defined as COMMIT_ON_RETURN = 'Y' was called before this cursor reference).
- Cursor *cursor-name* was allocated, but its associated stored procedure was called again since the cursor was allocated, new result sets were returned, and cursor *cursor-name* was deallocated.

System Action: The statement cannot be executed.

Programmer Response: Check the application program for completeness and for a possible spelling error in the cursor declaration or allocation. The declaration for or allocation of a cursor must appear in an application program before SQL statements that reference the cursor.

If the *cursor-name* was <UNKNOWN>, then the cursor was not successfully declared or allocated. This can occur if SQL(DB2) was used, and a warning message was issued during precompilation. Check the precompile output for warning messages on the DECLARE CURSOR or ALLOCATE CURSOR statement, and correct the statement.

For an allocated cursor, if an implicit or explicit COMMIT, ROLLBACK, or CLOSE occurred since the cursor was successfully allocated, modify the application program logic to do one of the following:

- After the COMMIT, ROLLBACK, or CLOSE operation, call the associated stored procedure again, and reissue the ASSOCIATE LOCATORS and ALLOCATE CURSOR statements.
- For COMMIT, declare the associated cursor in the stored procedure WITH HOLD so the COMMIT operation will not deallocate the cursor.

For an allocated cursor, if the associated stored procedure was called again and new result sets were returned since the cursor was allocated, reissue the ASSOCIATE LOCATORS and ALLOCATE CURSOR statements.

SQLSTATE: 34000

-507 THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT OPEN

Explanation: The application program attempted to execute an UPDATE or DELETE WHERE CURRENT OF cursor statement at a time when the specified cursor was not open.

System Action: The statement cannot be executed. No update or delete was performed.

Programmer Response: Check for a previous SQL return code that might have closed the cursor. SQLCODES -404, -652, -679, -901, -904, -909, -910, -911, and -913 force the cursor to close. After the cursor is closed, any fetches or close cursor statements receive SQLCODE -501. Any updates or deletes receive this SQLCODE -507. Correct the logic of the application program to ensure that the specified cursor is open at the time the UPDATE or DELETE statement is executed.

SQLSTATE: 24501

-508 THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT POSITIONED ON A ROW

Explanation: The application program attempted to execute an UPDATE or DELETE WHERE CURRENT OF cursor statement at a time when the specified cursor was not positioned on a row of the object table. The cursor must be positioned on the row that is to be updated or deleted.

This SQL code can be issued if the cursor is no longer positioned on the row because another cursor in the same application program delete the row or updates an index column. This includes deletes and index column updates that are performed as a result of rolling back to a savepoint.

System Action: The statement cannot be executed. No data was updated or deleted.

Programmer Response: Correct the logic of the application program to ensure that the cursor is

correctly positioned on the intended row of the object table before the UPDATE or DELETE statement is executed. Note that the cursor is not positioned on a row if FETCH returned an SQLCODE = 100.

SQLSTATE: 24504

-509 THE TABLE IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT THE SAME TABLE DESIGNATED BY THE CURSOR

Explanation: The application program attempted to execute an UPDATE or DELETE WHERE CURRENT OF cursor statement where the table named in that statement did not match the name of the table specified in the declaration for that cursor.

System Action: The statement cannot be executed. The update or delete was not performed.

Programmer Response: Correct the application program to ensure that the table identified in the UPDATE or DELETE statement is the same table identified in the declaration for the cursor.

SQLSTATE: 42827

-510 THE TABLE DESIGNATED BY THE CURSOR OF THE UPDATE OR DELETE STATEMENT CANNOT BE MODIFIED

Explanation: The application program attempted to execute an UPDATE or DELETE WHERE CURRENT OF cursor statement against a table or view that cannot be updated or deleted. This can occur for a delete from a read-only view or for an update in which the cursor was not defined with the FOR UPDATE clause.

This error code is also returned when the table exists at a remote location and DB2 has employed block fetching because you explicitly declared the cursor FOR FETCH ONLY, or because the application is bound CURRENTDATA(NO) and the cursor is ambiguous.

This error code is also returned if DB2 has employed parallelism to execute the SELECT statement associated with the cursor named in a DELETE WHERE CURRENT OF cursor statement, or if a DELETE WHERE CURRENT OF is issued against a row which DB2 cannot guarantee to have not been modified by another application since the time the cursor was positioned upon it (in accordance with ISO(CS)) semantics for an ambiguous cursor in an application bound CURRENTDATA(NO)).

System Action: The statement cannot be processed. No data was updated or deleted in the object table.

Programmer Response: The requested UPDATE or DELETE cannot be performed. Refer to Chapter 6 of *DB2 SQL Reference* for information about restrictions on using UPDATE and DELETE operations against views.

For a remote table, modify the DECLARE CURSOR and then rebind the PLAN.

For a cursor that uses parallelism, disable parallelism for the query by using the DEGREE(1) BIND option for static SQL or by setting the CURRENT DEGREE special register to '1' for dynamic SQL.

For an ambiguous cursor in an application bound CURRENTDATA(NO), either make the cursor unambiguous (declare it FOR UPDATE OF), or rebind the application CURRENTDATA(YES).

SQLSTATE: 42828

-511 THE FOR UPDATE CLAUSE CANNOT BE SPECIFIED BECAUSE THE TABLE DESIGNATED BY THE CURSOR CANNOT BE MODIFIED

Explanation: The result table of the SELECT statement cannot be updated. This can occur if the SELECT specifies more than one table or view in the FROM clause, if the SELECT list contains a built-in function or DISTINCT, or if the statement contains an ORDER BY or GROUP BY or HAVING clause. This can also occur if a view is specified in the FROM clause and the view cannot be updated.

System Action: The statement cannot be executed. The specified cursor remains undefined in the application program.

Programmer Response: Updates cannot be performed on the result table as it is specified. Refer to Chapter 6 of *DB2 SQL Reference* for information about restrictions on the updating of views.

SQLSTATE: 42829

-512 STATEMENT REFERENCE TO REMOTE OBJECT IS INVALID

Explanation: One of the following conditions exists:

- The statement refers to multiple locations.
- A statement with a remote reference is being EXPLAINED either by a dynamic EXPLAIN statement or the EXPLAIN(YES) option.
- An alias is used incorrectly.
- A three-part name is implicitly or explicitly used in a statement that is not supported by the DB2 private protocols.
- A three-part name is implicitly or explicitly used in a triggered statement.
- A PREPARE statement contains an ATTRIBUTES clause. This is not supported by the BD2 private protocols.

System Action: The statement cannot be executed.

Programmer Response: If the object cannot be meaningfully eliminated from the statement, see your Database Administrator for other ways to obtain the

data required. Refer to Chapter 4 of *DB2 SQL Reference* for more information about using remote objects.

If the remote object reference is in a triggered SQL statement, you can instead invoke a user-defined function or a stored procedure from the trigger and access the remote object from the function or stored procedure.

SQLSTATE: 56023

-513 THE ALIAS *alias-name* MUST NOT BE DEFINED ON ANOTHER LOCAL OR REMOTE ALIAS

Explanation: The object indicated by 'alias-name' is a local or remote alias. An alias is not allowed to be defined on a local alias, and it should not be defined on a remote alias.

System Action: The statement cannot be executed.

Programmer Response: Modify the SQL statement to ensure that all object references are to base tables or views.

SQLSTATE: 42924

-514 THE CURSOR *cursor-name* IS NOT IN A PREPARED STATE

Explanation: The application program has tried to use a cursor, 'cursor-name,' that is not in a prepared state. The cursor is associated with a statement that either (1) has never been prepared, or (2) has been invalidated by a commit or rollback operation.

System Action: The statement cannot be executed.

Programmer Response: For case (1), ensure that you prepare the statement that is named in the DECLARE CURSOR statement for 'cursor-name' before you try to open the cursor. For case (2), do one of the following:

- Use the WITH HOLD option of DECLARE CURSOR.
- Do not execute a commit or rollback operation until you are finished using the cursor.
- Prepare the statement again after the commit or rollback.

SQLSTATE: 26501

-516 THE DESCRIBE FOR STATIC STATEMENT DOES NOT IDENTIFY A PREPARED STATEMENT

Explanation: An attempt was made to execute a DESCRIBE FOR STATIC for a statement that had not been successfully prepared beforehand. This error can occur when the DESCRIBE originates on an application requester that supports extended dynamic SQL. Because the target statement is static on the DB2 for MVS/ESA subsystem, the DESCRIBE statement fails.

System Action: The statement cannot be executed.

Programmer Response: Correct the logic of the application program to ensure that a statement is properly prepared before a DESCRIBE FOR STATIC of the statement is attempted. If the DESCRIBE FOR STATIC is a distributed request that originated on a system that supports extended dynamic SQL, contact your system administrator about changing the DB2 subsystem parameter DESCSTAT to YES to tolerate these DESCRIBE FOR STATIC requests against static SQL.

SQLSTATE: 26501

-517 CURSOR *cursor-name* CANNOT BE USED BECAUSE ITS STATEMENT NAME DOES NOT IDENTIFY A PREPARED SELECT STATEMENT

Explanation: The cursor 'cursor-name' could not be used as specified because the prepared statement named in the declaration for the cursor was not a SELECT statement.

System Action: The statement cannot be executed.

Programmer Response: Verify that the statement-name is specified correctly in the PREPARE statement and the DECLARE CURSOR statement for cursor 'cursor-name'. Alternatively, correct the application program logic to ensure that only prepared SELECT statements are used in association with cursor declarations.

SQLSTATE: 07005

-518 THE EXECUTE STATEMENT DOES NOT IDENTIFY A VALID PREPARED STATEMENT

Explanation: One of the following conditions exists:

- The statement named in the EXECUTE statement has not been prepared.
- The statement named in the EXECUTE statement identifies a SELECT, VALUE INTO, or statement.
- The statement named in the EXECUTE IMMEDIATE statement identifies a SELECT, VALUE INTO, or statement.

System Action: The statement cannot be executed.

Programmer Response: Ensure that you prepare the statement prior to EXECUTE. Also, ensure that the statement you prepare is not a SELECT or VALUES INTO statement.

SQLSTATE: 07003

-519 **THE PREPARE STATEMENT IDENTIFIES THE SELECT STATEMENT OF THE OPENED CURSOR** *cursor-name*

Explanation: The application program has attempted to PREPARE (actually, re-PREPARE) the SELECT statement for the specified cursor at a time when that cursor was already open.

System Action: The statement cannot be executed. The cursor was not affected.

Programmer Response: Correct the logic of the application program so that it does not attempt to re-PREPARE the SELECT statement for a cursor when that cursor is open.

SQLSTATE: 24506

-525 **THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT WAS IN ERROR AT BIND TIME FOR SECTION =** *sectno* **PACKAGE =** *pkgname* **CONSISTENCY TOKEN =** *contoken*

Explanation: One of the following:

- The statement was in error when the package was bound, but the error was ignored then because the option SQLERROR (CONTINUE) was used. Since the statement contains an error, it cannot be executed.
- The statement might not be an executable statement at this location, or might only be executable by a DB2 application requester (for example, DECLARE TABLE in an application running on OS/2 causes this message).

The variables are:

sectno Section number

pkgname
locid.collid.pkgid

contoken
Consistency token in hexadecimal

System Action: The statement cannot be executed.

Programmer Response: If the SQL statement is not supposed to execute at the indicated location, then correct the program so that the statement in error does not execute at that location. Precompile, compile, and bind replace the package. If the SQL statement is supposed to execute at the indicated location, correct the problem found when it was bound and bind the package over using BIND with ACTION(REPLACE). If multiple versions of the package have been bound, issue the following SELECT statement to determine which version has the error: SELECT VERSION FROM *locid*.SYSIBM.SYSPACKAGE WHERE LOCATION = *locid* AND COLLID = *collid* AND NAME = *pkgid* AND HEX(CONTOKEN) = *contoken*

Where:

locid Location name

collid Collection id

pkgid Program name

SQLSTATE: 51015

-526 **THE REQUESTED OPERATION OR USAGE DOES NOT APPLY TO** *table type* **TEMPORARY TABLE** *table name*

Explanation: DB2 assumes that the SQL statement being executed refers to a *created* or *declared* temporary table named *table name*, and the requested operation or usage in the statement is not allowed on the temporary table.

table type
CREATED or DECLARED

CREATED is for a temporary table defined by the CREATE GLOBAL TEMPORARY TABLE statement.

DECLARED is for a temporary table defined by the DECLARE GLOBAL TEMPORARY TABLE statement.

table name
Qualified name of the temporary table.

System Action: The statement cannot be processed.

Programmer Response: Modify the SQL statement to ensure that the object references are not to the indicated type of temporary table, or if *table type* is DECLARED and you intended *table name* to refer to an existing persistent base table, you must perform one of the following actions:

- Recreate the persistent base table *table name* with a different qualifier
- In the same application process, issue a DROP TABLE for *table name* followed by a COMMIT to drop the declared temporary table and afterwards be able to reference the persistent base table with the same *table name* in the same application process
- Remove the DECLARE GLOBAL TEMPORARY TABLE statement from the application process to use the persistent base table with the same *table name*

SQLSTATE: 42995

-530 **THE INSERT OR UPDATE VALUE OF FOREIGN KEY** *constraint-name* **IS INVALID**

Explanation: An UPDATE or INSERT operation attempted to place a value in a foreign key of the object table; however, this value was not equal to some value of the parent key of the parent table.

When a row is inserted into a dependent table, the insert value of a foreign key must be equal to the value

of the parent keyof some row of the parent table in the associated relationship.

When the value of the foreign key is updated, the update value of a foreign key must be equal to the value of the parent keyof some row of the parent table of the associated relationship.

System Action: The UPDATE or INSERT statement cannot be executed. The object table is unchanged.

Programmer Response: Examine the insert or update value of the foreign key first, and then compare it with each of the parent keyvalues of the parent table to determine the cause of the problem.

SQLSTATE: 23503

-531 PARENT KEY IN A PARENT ROW CANNOT BE UPDATED BECAUSE IT HAS ONE OR MORE DEPENDENT ROWS IN RELATIONSHIP *constraint-name*

Explanation: For plans and packages bound beginning with Version 5 or dynamic SQL, a multi-row update of a parent key attempted to remove a parent key value on which a foreign key was dependent.

For plans and packages bound prior to Version 5 an UPDATE operation attempted to update a primary key in the specified row of the object table; however, the primary key in the specified row had dependent rows associated with it. The value of a primary key in a parent row cannot be updated if the parent row has any dependent rows.

System Action: The UPDATE statement cannot be executed. The object table is unchanged.

Programmer Response: Examine the parent key of the object table and the foreign key of the dependent table to determine if the value of the specified row of the parent keyshould be changed. If this does not expose the problem, examine the contents of the object table and the dependent table to determine the cause of the problem.

SQLSTATE: 23504

-532 THE RELATIONSHIP *constraint-name* **RESTRICTS THE DELETION OF ROW WITH RID X'rid-number'**

Explanation: A DELETE operation attempted to delete a specified parent row in the object table and all related descendent rows in the descendent tables. However, a delete rule of RESTRICT or NO ACTION was specified for one or more descendent tables.

A row of the table cannot be deleted because it has a dependent in a relationship with a delete rule of RESTRICT or NO ACTION or the deletion cascades to a row which is a dependent in a relationship with a delete rule of RESTRICT or NO ACTION.

System Action: The DELETE statement cannot be executed. The contents of the object table are unchanged.

Programmer Response: Examine the delete rule for all descendent tables to determine the cause of the problem. The specific tables involved can be determined from the relationship 'constraint-name'. The specific descendent row is known by RID X'rid-number'.

SQLSTATE: 23504

-533 INVALID MULTIPLE-ROW INSERT

Explanation: An INSERT operation with a subselect attempted to insert multiple rows into a self-referencing table.

The subselect of the INSERT operation should return no more than one row of data.

System Action: The INSERT statement cannot be executed. The contents of the object table are unchanged.

Programmer Response: Examine the search condition of the subselect to make sure that no more than one row of data is selected.

SQLSTATE: 21501

-534 THE PRIMARY KEY CANNOT BE UPDATED BECAUSE OF MULTIPLE-ROW UPDATE

Explanation: An UPDATE operation attempted to update a primary key on multiple rows of the object table.

An UPDATE statement updating the primary key cannot be used to update more than one row of the object table.

Note: This SQLCODE will only be issued for plans and packages bound prior to Version 5. SQLCODE -534 will not be issued for dynamic SQL or plans and packages bound with Version 5 or later releases.

System Action: The UPDATE statement cannot be executed. The contents of the object table are unchanged.

Programmer Response: Examine the search condition of the UPDATE statement to make sure that no more than one row of the object table is selected to be updated.

SQLSTATE: 21502

-536 **THE DELETE STATEMENT IS INVALID BECAUSE TABLE *table-name* CAN BE AFFECTED BY THE OPERATION**

Explanation: A DELETE operation with the indicated table referenced in a subquery was attempted.

If 'T' is the object table of the DELETE, the indicated table is one of the following:

- A dependent of 'T' in a relationship with a delete rule of CASCADE or SET NULL
- A dependent of another table in a relationship with a delete rule of CASCADE or SET NULL in which deletions from 'T' can cascade to that table.

System Action: The DELETE statement cannot be processed. The contents of the object table are unchanged.

Programmer Response: Do not attempt to reference a table in a subquery of a DELETE statement when the table can be affected by the DELETE statement.

SQLSTATE: 42914

-537 **THE PRIMARY KEY CLAUSE, A FOREIGN KEY CLAUSE, OR A UNIQUE CLAUSE IDENTIFIES COLUMN *column-name* MORE THAN ONCE**

Explanation: PRIMARY KEY, FOREIGN KEY, or UNIQUE can each be followed by a list of column names. The statement violates the rule that no column name can appear more than once in any such list.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement to specify unique names for each column.

SQLSTATE: 42709

-538 **FOREIGN KEY *name* DOES NOT CONFORM TO THE DESCRIPTION OF A PARENT KEY OF TABLE *table-name***

Explanation: The definition of the indicated foreign key does not conform to the description of parent key of the indicated table due to one of the following reasons:

- The referenced parent kkey has not been defined as a primary key or a unique key.
- The keys do not have the same number of columns.
- The decription of the keys are not identical. The requirement for identical descriptions includes data type, length attribute, and field procedure.

name is the constraint-name specified in the foreign key clause or, if a constraint-name was not specified, the first column-name specified in the clause.

System Action: The statement cannot be processed.

Programmer Response: Correct the statement so that the description of the foreign key references a primary key or unique key, or so that the description of the foreign key conforms to that of a parent key of the indicated table.

SQLSTATE: 42830

-539 **TABLE *table-name* DOES NOT HAVE A PRIMARY KEY**

Explanation: DB2 cannot perform the CREATE or ALTER TABLE statement because the indicated table does not have a primary key. Thus, the primary key cannot be dropped, or the table cannot be defined as a parent in a referential constraint.

System Action: The statement cannot be processed.

Programmer Response: Correct the statement to reference a table with a primary key, or define a primary key with ALTER TABLE ADD PRIMARY KEY before referencing the table in a FOREIGN KEY clause.

SQLSTATE: 42888

-540 **THE DEFINITION OF TABLE *table-name* IS INCOMPLETE BECAUSE IT LACKS A PRIMARY INDEX OR A REQUIRED UNIQUE INDEX**

Explanation: The table named was defined with a PRIMARY KEY clause, a UNIQUE clause, or with a ROWID column with the GENERATED BY DEFAULT attribute. Its definition is incomplete, and it cannot be used until a unique index is defined for

- the primary key (the primary index)
- a ROWID column
- for each set of columns in any UNIQUE clause (the required unique indexes).

An attempt was made to use the table in a FOREIGN KEY clause or in an SQL manipulative statement.

System Action: The statement cannot be executed.

Programmer Response: Define a primary index or a required unique index on the table before referencing it.

SQLSTATE: 57001

-542 ***column-name* CANNOT BE A COLUMN OF A PRIMARY KEY, A UNIQUE CONSTRAINT, OR A PARENT KEY BECAUSE IT CAN CONTAIN NULL VALUES**

Explanation: The code is used to report that a column identified in a PRIMARY KEY, a UNIQUE constraint clause, or a parent key (via a REFERENCES clause) is defined to allow null values.

System Action: The statement cannot be executed.

Programmer Response: In the case of a column identified in a PRIMARY KEY or a UNIQUE constraint clause, correct the statement and rerun it.

In the case of a column identified in a REFERENCES clause, drop the parent table then recreate it with referenced columns defined as NOT NULL. Afterwards, rerun the statement.

SQLSTATE: 42831

-543 **A ROW IN A PARENT TABLE CANNOT BE DELETED BECAUSE THE CHECK CONSTRAINT** *check-constraint* **RESTRICTS THE DELETION**

Explanation: The delete operation cannot be executed because the target table is a parent table and is connected with a referential constraint to a dependent table with a delete rule of SET NULL. However, a check constraint defined on the dependent table restricts the column from containing a null value.

System Action: The DELETE statement was not executed. The contents of the tables are unchanged.

Programmer Response: Examine the foreign key and its delete rule in the dependent table and the conflicting check constraint. Change either the delete rule or the check constraint so that they do not conflict.

SQLSTATE: 23511

-544 **THE CHECK CONSTRAINT SPECIFIED IN THE ALTER TABLE STATEMENT CANNOT BE ADDED BECAUSE AN EXISTING ROW VIOLATES THE CHECK CONSTRAINT**

Explanation: An existing row violates the check constraint specified in the ALTER TABLE statement.

System Action: The statement cannot be executed. The check constraint definition is not added to the table. The table definition is unchanged.

Programmer Response: Examine the check constraint definition that was specified in the ALTER TABLE statement and the data in the table to determine why the ALTER TABLE statement was rejected.

You can determine which rows violated the check constraint by using the SELECT statement, negating the check constraint in the WHERE clause. For example:

```
SELECT * FROM table WHERE (NOT (check-condition));
```

SQLSTATE: 23512

-545 **THE REQUESTED OPERATION IS NOT ALLOWED BECAUSE A ROW DOES NOT SATISFY THE CHECK CONSTRAINT** *check-constraint*

Explanation: Table check constraint violations occurred on an INSERT or UPDATE statement. The resulting row violated the check constraint definition on the table.

System Action: The INSERT or UPDATE statement cannot be executed. The contents of the table are unchanged.

Programmer Response: Examine the data and the check constraint definition in the SYSIBM.SYSCHECKS catalog table to determine why the INSERT or UPDATE statement was rejected. The data must be changed to satisfy the check constraint.

SQLSTATE: 23513

-546 **THE CHECK CONSTRAINT** *constraint-name* **IS INVALID**

Explanation: A check constraint in the CREATE TABLE or ALTER TABLE statement is invalid for one or more of the following reasons:

- The constraint definition refers to a column that has a field procedure.
- The constraint definition refers to a column with a data type that is lower in the hierarchy of numeric data types than the data type of any other operand. The hierarchy is as follows:

  ```
  small integer < large integer < decimal
  < single precision float < double precision float
  ```
- The constraint definition refers to a column with a numeric data type that is not the same numeric data type as that of the other column operands.
- The constraint definition refers to a column with a length that is shorter than the other operands when the column and other operands are not character string data types.
- The constraint definition refers to a built-in or user-defined function.
- The constraint definition uses a cast function that requires conversion of the data. The only functions that are allowed in a check constraint are cast functions that do not require conversion of the data.

System Action: The statement is not executed.

For ALTER TABLE, the check constraint is not added to the object table. The definition of the table is unchanged.

For CREATE TABLE, the table is not created.

Programmer Response: Correct the check constraint definition and execute the statement again.

SQLSTATE: 42621

-548 **A CHECK CONSTRAINT THAT IS DEFINED WITH** *column-name* **IS INVALID**

Explanation: A check constraint in the CREATE TABLE or ALTER TABLE statement is invalid for one or more of the following reasons:

- The constraint definition refers to a column that has a field procedure.

- The constraint definition refers to a column with a data type that is lower in the hierarchy of numeric data types than the data type of any other operand. The hierarchy is as follows:

  ```
  small integer < large integer < decimal
  < single precision float < double precision float
  ```

- The constraint definition refers to a column with a numeric data type that is not the same numeric data type as that of the other column operands.
- The constraint definition refers to a column with a length that is shorter than the other operands when the column and other operands are not character string data types.
- The constraint definition refers to a ROWID column.
- The constraint definition refers to a LOB column.

System Action: The statement is not executed.

For ALTER TABLE, the check constraint is not added to the object table. The definition of the table is unchanged.

For CREATE TABLE, the table is not created.

Programmer Response: Correct the check constraint definition and execute the statement again.

SQLSTATE: 42621

-549 **THE** *statement* **STATEMENT IS NOT ALLOWED FOR** *object_type1* *object_name* **BECAUSE THE BIND OPTION DYNAMICRULES(RUN) IS NOT IN EFFECT FOR** *object_type2*

Explanation: A program attempted to issue the indicated SQL statement that is one of several SQL statements that cannot be issued from a plan or package for which the option DYNAMICRULES(RUN) is not in effect. Those SQL statements are:

- Dynamic GRANT statement
- Dynamic REVOKE statement
- Dynamic ALTER statement
- Dynamic CREATE statement
- Dynamic DROP statement

The indicated SQL statement is bound to one of the following:

- The named plan or package that was not bound with the option DYNAMICRULES(RUN)
- The named package that was not bound with the DYNAMICRULES option, but is appended to a plan that was not bound with DYNAMICRULES(RUN)

statement
The SQL statement in error

object_type1
PACKAGE or DBRM

object_name
If *object_type1* is PACKAGE, *object_name* is the name of the package in the format 'location-id.collection-id.package-id'.

If *object_type1* is DBRM, *object_name* is the name of the DBRM in the format 'plan-name DBRM-name'.

object_type2
PLAN or PACKAGE

If *object_type1* is PACKAGE, *object_type2* can be either PACKAGE or PLAN (whichever is bound with a DYNAMICRULES value other than RUN).

If *object_type1* is DBRM, *object_type2* is PLAN.

System Action: The SQL statement cannot be executed.

Programmer Response: Do one of the following to correct the error:

- If the SQL statement is embedded, remove it, precompile and compile the application program again, and reissue the BIND command with the desired DYNAMICRULES option.
- If appropriate, use the SQL statement with a package or plan that is bound with DYNAMICRULES(RUN).
- Issue the REBIND command with the DYNAMICRULES(RUN) option for the plan or package to which the SQL statement is bound

Refer to the BIND PACKAGE(DSN), BIND PLAN(DSN), REBIND PACKAGE(DSN), or REBIND PLAN(DSN) statement in *DB2 Command Reference* for the description of the DYNAMICRULES option and the expected results. Determine if either the SQL statement should be removed from the program or the plan or package should be rebound with the DYNAMICRULES(RUN) option.

SQLSTATE: 42509

-551 *auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation* **ON OBJECT** *object-name*

Explanation: Authorization ID *auth-id* attempted to perform *operation* on object *object-name* without having been granted the proper authority to do so. This error might also occur if the object is a read-only view (for INSERT, DELETE, or UPDATE), or if *auth-id* is trying to create a table or view with an authorization ID other than its own.

You can create a table from an *auth-id* other than your own only if your authorization ID is SYSADM, DBADM, or DBCTRL. You can create a view from an *auth-id* other than your own only if your authorization ID is SYSADM.

When *operation* is 'GRANT ***', the keyword ALL was used in the GRANT statement, but the grantor *auth-id* did not possess any privilege to grant.

If this error occurs while DB2 is creating or altering a table involving referential constraints, this code reports that the user does not have the necessary ALTER privilege to perform a FOREIGN KEY, DROP FOREIGN KEY, DROP PRIMARY KEY, or DROP UNIQUE operation. The *object-name* identifies the object table of the CREATE or ALTER TABLE statement, not the table for which the user lacks the ALTER privilege.

If this error occurs for a distributed SQL request, then:
1. If authorization ID translation is in effect for either the requesting DB2 site or the serving (responding) DB2 site, then *auth-id* is the translated authorization ID. Refer to Part 3 (Volume 1) of *DB2 Administration Guide* for information on authorization ID translation.
2. If an alias name was used in the SQL statement, then *object-name* is the resolved remote table or view name.

If the operation is a DROP PACKAGE, the object name consists of the collection ID, the package name and the consistency token. The consistency token uniquely identifies which version of the package the user does not have authorization to drop.

Note: Beginning with Version 5, SQLCODE -551 will be returned instead of SQLCODE -204 for the runtime error where an object does not exist and the CURRENT RULES special register is set to 'STD'.

System Action: The statement cannot be executed.

Installation Action: Check for an attempted authorization violation.

Programmer Response: Ensure that *auth-id* was granted the authority to perform the desired operation, the *object-name* exists, and *auth-id* is not trying to create a table with a different authorization ID.

SQLSTATE: 42501

-552 *auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation*

Explanation: Authorization ID 'auth-id' has attempted to perform the specified 'operation' without having been granted the authority to do so.

System Action: The statement cannot be executed.

Installation Action: Check for an attempted authorization violation.

Programmer Response: Ensure that the authorization ID has been granted the authority necessary to perform the desired operation.

SQLSTATE: 42502

-553 *auth-id* **SPECIFIED IS NOT ONE OF THE VALID AUTHORIZATION IDS**

Explanation: The authorization ID specified as the value of the 'authorization-id' or host variable in the SQL SET CURRENT SQLID statement is neither the user's primary authorization ID nor one of the associated secondary authorization IDs.

System Action: The SET CURRENT SQLID statement cannot be executed. The current SQL ID is not changed.

Programmer Response: Correct the error in the statement or contact the security administrator to have the authorization ID defined for your use.

SQLSTATE: 42503

-554 **AN AUTHORIZATION ID CANNOT GRANT A PRIVILEGE TO ITSELF**

Explanation: An authorization ID attempted to execute a GRANT statement in which that ID itself appears as one of the entries in the list of 'grantee' authorization IDs.

An authorization ID cannot GRANT a privilege to itself. However, if SQLRULES(STD) is in effect or CURRENT RULES contains STD, GRANT to self is allowed.

System Action: The statement cannot be executed. No privileges were granted.

Programmer Response: Refer to Chapter 5 of *DB2 SQL Reference* for information about restrictions on the use of the GRANT statement.

SQLSTATE: 42502

-555 **AN AUTHORIZATION ID CANNOT REVOKE A PRIVILEGE FROM ITSELF**

Explanation: An authorization ID attempted to execute a REVOKE statement in which that ID itself appears as one of the entries in the list of authorization IDs to be revoked.

An authorization ID cannot REVOKE its own privilege. However, if SQLRULES(STD) is in effect or CURRENT RULES contains STD, REVOKE to self is allowed.

System Action: The statement cannot be executed. No privileges were revoked.

Programmer Response: Refer to Chapter 6 of *DB2 SQL Reference* for information about restrictions on the use of the REVOKE statement.

SQLSTATE: 42502

-556 *authid2* **CANNOT HAVE THE** *privilege* **PRIVILEGE** *on_object* **REVOKED BY** *authid1* **BECAUSE THE REVOKEE DOES NOT POSSESS THE PRIVILEGE OR THE REVOKER DID NOT MAKE THE GRANT**

Explanation: The REVOKE statement was not successful for one of the following reasons:

- *Authid2* does not possess the *privilege*.
- The revoker, *authid1*, did not explicitly grant the privilege to *authid2*.
- *Authid2* is the owner of the specified object.
- When *privilege* is '***' the keyword ALL was used in the REVOKE statement, but *authid2* did not possess any privilege to revoke.
- When *authid1* is ALL, the BY ALL clause was used in the REVOKE statement, but *authid2* did not possess any privilege to revoke.

An authorization ID can revoke only the privileges that it has explicitly granted to other authorization IDs, unless the authorization ID has SYSADM or SYSCTRL authority and specifies the BY clause. No authorization ID, not even SYSADM, can revoke 'privileges' on an object from the object owner.

System Action: The statement cannot be executed. No privileges were revoked from any authorization ID.

Programmer Response: Check the appropriate authorization catalog tables to verify that *authid2* possesses the privilege to be revoked. Queries can be made with GRANTEE = *authid2* and the privilege column not = blanks. Correct and reissue the REVOKE statement.

If a user holding SYSADM or SYSCTRL authority receives this SQLCODE, the BY clause might have been omitted from the REVOKE statement.

SQLSTATE: 42504

-557 **INCONSISTENT GRANT/REVOKE KEYWORD** *keyword***. PERMITTED KEYWORDS ARE** *keyword-list*

Explanation: The GRANT or REVOKE statement contains a syntax or spelling error at or before the specified 'keyword'. As an aid to the programmer, 'keyword-list' provides a list of the keywords that would be permitted in this context.

Alternatively:

- The mixture of privileges specified on the GRANT or REVOKE statement is not permitted. The privileges must all be of one type, and consistent with the form of the GRANT or REVOKE statement.
- REVOKE UPDATE (column-list) is not permitted; only REVOKE UPDATE is valid.
- The keywords DELETE, INSERT, SELECT, TRIGGER, UPDATE, REFERENCES and ALTER cannot be specified for an auxiliary table.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the GRANT or REVOKE statement.

SQLSTATE: 42852

-558 **INVALID CLAUSE OR COMBINATION OF CLAUSES ON A GRANT OR REVOKE**

Explanation: The location qualifier specified for a GRANT or REVOKE statement is invalid.

System Action: The statement cannot be executed.

Programmer Response: Refer to Chapter 6 of *DB2 SQL Reference* for valid keywords for the GRANT statement.

SQLSTATE: 56025

-559 **ALL AUTHORIZATION FUNCTIONS HAVE BEEN DISABLED**

Explanation: The authorization mechanism has been disabled in the DB2 subsystem. Consequently, GRANT and REVOKE statements are ignored.

System Action: The statement cannot be executed. No privileges were granted or revoked.

Programmer Response: Do not attempt to execute GRANT or REVOKE statements unless and until the authorization mechanism is enabled in the DB2 subsystem.

SQLSTATE: 57002

-567 *bind-type* **AUTHORIZATION ERROR USING** *auth-id* **AUTHORITY PACKAGE =** *package-name* **PRIVILEGE =** *privilege*

Explanation: The authorization ID given does not have the privilege indicated, and cannot invoke the indicated subcommand against the indicated package.

bind-type
Type of bind subcommand (BIND | REBIND | FREE).

auth-id Authorization ID of the package owner.

package-name
Name of the package (location.collection.package.version)

privilege
Name of the privilege not held:

- BINDADD—The authority to create a new package using BIND with the ADD option.

- BIND—The authority to BIND (REPLACE) or REBIND a package.
- COPY—The authority to COPY from the indicated package
- CREATE IN—The authority to create a package in the indicated collection.

System Action: The indicated package is not bound, rebound, or freed.

System Programmer Response: The indicated privilege must be granted to the authorization ID that will become the package owner.

SQLSTATE: 42501

-571 THE STATEMENT WOULD RESULT IN A MULTIPLE SITE UPDATE

Explanation:

This SQLCODE is issued in the following situations:

- When an application program operating in an IMS or CICS environment attempts to modify data at a remote location where multi-site update capabilities are not supported.
- When an application program has explicit SQL statements within a commit scope that would result in updates at multiple sites where one of the sites at which data is being updated does not support multi-site update.

This SQLCODE can be issued when an application program explicitly modifies data at a single location within a commit scope. This can occur in the following situations:

- A package or plan associated with the application program was invalidated.
- A package or plan was bound at one release of DB2 and fallback occurs to a prior release.

In the situations described above, an implicit autobind is done on behalf of the user. An autobind results in the DB2 catalog being updated. The conditions that must exist for this SQLCODE to be issued when an autobind occurs are:

- One site where data has been modified does not support multi-site update.
- The autobind occurs at a separate and distinct site from where an application program explicitly modifies data.
- At the time of the autobind, locks are being held to process an SQL statement within the application program.

System Action: The statement cannot be executed.

Programmer Response:

- Ensure that all requests for modifications to the data are confined to a single location within any given commit scope for any application that references a location that does not support multi-site update.
- For programs operating in an IMS or CICS environment where the remote database systems do not support multi-site update, all SQL statements must be read-only access.
- If an autobind is causing this SQLCODE to be issued, REBIND the plan or package.

SQLSTATE: 25000

-574 THE SPECIFIED DEFAULT VALUE OR IDENTITY ATTRIBUTE VALUE CONFLICTS WITH THE DEFINITION OF COLUMN *column-name*

Explanation: The DEFAULT value specified for *column-name* is not valid for one of the following reasons:

- The value is not assignable to the column because the constant does not conform to the format for a constant of that data type, or the value has the incorrect length or precision.
- A floating-point constant is specified and the column is not a floating point data type.
- A decimal constant is specified and non-zero digits would be truncated when assigned to the column.
- The value is more than 255 bytes, including quotes for strings, introducer characters such as the X for a hex constant, fully qualified function names, and parentheses.
- Either the USER or CURRENT SQLID special register is specified and the length attribute of the character string data type is less than 8.
- A system-generated cast function was specified and the column is not defined with a user-defined distinct type
- A function was specified that is not supported. A function may only be specified when the data type is a distinct type, and in this case the specified function must be one of the system-generated cast functions associated with this distinct type.
- WITH DEFAULT is specified with a value other than NULL for a LOB column.
- A value with non-zero scale was specified for the START WITH or INCREMENT BY option of an identity column with the DECIMAL data type.

System Action: The SQL statement cannot be executed.

Programmer Response: Specify a default value or attribute that is valid for the definition of the column.

SQLSTATE: 42894

-577 *object-type object-name* **ATTEMPTED TO MODIFY DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION**

Explanation: The current environment does not allow SQL statements that modify data. One of the following situations has occurred:

- A user-defined function or stored procedure *object-name* was invoked and attempted to modify data, but the function or procedure was defined without the MODIFIES SQL option.
- A user-defined function or stored procedure *object-name* was invoked and attempted to execute a data definition statement, but the function or procedure was defined without the MODIFIES SQL option.
- A function or procedure defined with READS SQL DATA, CONTAINS SQL, or NO SQL has attempted to invoke another function or procedure defined with MODIFIES SQL DATA.

In an environment of nested functions and procedures, the SQL option in effect is the most restrictive one that has been specified in the nested hierarchy of functions and procedures. The SQL data access option in effect does not allow for modifying the data.

System Action: The SQL statement failed.

Programmer Response: Either use an ALTER statement to change the definition of the function or procedure to allow statements that modify data, or remove the failing SQL statement from the external function or procedure.

SQLSTATE: 38002

-579 *object-type object-name* **ATTEMPTED TO READ DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION**

Explanation: The current environment does not allow SQL statements that read data. One of the following situations had occurred:

- A user-defined function or stored procedure *object-name* was invoked and attempted to read data, but the function or procedure was defined without the READS SQL DATA or MODIFIES SQL DATA option.
- A function or procedure defined with CONTAINS SQL or NO SQL has attempted to invoke another function or procedure defined with READS SQL DATA.

In an environment of nested functions and procedures, the SQL option in effect is the most restrictive one that has been specified in the nested hierarchy of functions and procedures. The SQL data access option in effect does not allow for reading data.

System Action: The SQL statement failed.

Programmer Response: Either use an ALTER statement to change the definition of the function or procedure to allow statements that read data, or remove the failing SQL statement from the external function or procedure.

SQLSTATE: 38004

-580 **THE RESULT-EXPRESSIONS OF A CASE EXPRESSION CANNOT ALL BE NULL**

Explanation: There is a CASE expression in the statement that has all the *result-expressions* (expressions following the THEN and ELSE keywords) coded with the keyword NULL.

System Action: The statement cannot be processed.

Programmer Response: Change the CASE expression to include at least one *result-expression* with a keyword other than NULL.

SQLSTATE: 42625

-581 **THE DATA TYPES OF THE RESULT-EXPRESSIONS OF A CASE EXPRESSION ARE NOT COMPATIBLE**

Explanation: There is a CASE expression in the statement that has *result-expressions* (expressions following THEN and ELSE keywords) that are not compatible. The data type of the *result-expressions* might be incompatible because the CASE condition result data types are not all:

- character
- graphic
- numeric
- date
- time
- timestamp

If encoded in Unicode, character and graphic data types are compatible, howver. Refer to DB2 SQL Reference for more information about Unicode.

System Action: The statement cannot be processed.

Programmer Response: Correct the *result-expressions* so that they are compatible.

SQLSTATE: 42804

-582 **THE SEARCH-CONDITION IN A SEARCHED-WHEN-CLAUSE CANNOT BE A QUANTIFIED PREDICATE, IN PREDICATE, OR AN EXISTS PREDICATE.**

Explanation: The *search-condition* in a *searched-when-clause* specifies a quantified predicate,

an IN predicate, or an EXISTS predicate, but is not allowed.

System Action: The statement cannot be processed.

Programmer Response: Correct the *search-condition*.

SQLSTATE: 42625

-583 **THE USE OF FUNCTION *function-name* IS INVALID BECAUSE IT IS NOT DETERMINISTIC OR HAS AN EXTERNAL ACTION**

Explanation: The function *function-name* is defined as a not deterministic function or a function with an external action. This type of function is not supported in the context in which it is used. The contexts in which these are not valid are:

- in the expression prior to the first WHEN keyword in a simple-case-expression.
- in the WHERE clause of the subselect in a CREATE VIEW statement if the WITH CHECK OPTION is specified.
- in an expression of an ORDER BY clause

System Action: The statement cannot be executed.

Programmer Response: If the use of a not deterministic or external action function was not intended, substitute a function without these characteristics. If the behavior associated with the not deterministic or external action function is intentional, use the alternate form of the statements that make that intent explicit.

- Instead of a simple-when-clause, use the corresponding searched-when-clause where the function would get specified in each search-condition.
- Remove the WITH CHECK OPTION from the CREATE VIEW statement.
- Remove the function from the ORDER BY clause. If the column is part of the result set of the query, change the expression in the ORDER BY clause to the *simple-integer* or *simple-column-name* form of the sort key. See the ORDER BY syntax diagram in the DB2 SQL Reference for more information.

SQLSTATE: 42845

-585 **THE SCHEMA NAME *schema-name* CANNOT APPEAR MORE THAN ONCE IN THE CURRENT PATH**

Explanation: The current path includes *schema-name* more than once. The current path can only include one occurrence of each schema name.

System Action: The statement cannot be executed.

Programmer Response: Remove duplicate occurrences of *schema-name* from the current path.

SQLSTATE: 42732

-586 **THE TOTAL LENGTH OF THE CURRENT PATH SPECIAL REGISTER CANNOT EXCEED 254 CHARACTERS**

Explanation: The CURRENT PATH special register is defined as a VARCHAR(254). The content of the string includes each schema name delimited with double quotes and separated from the next schema name by a comma. The total length of the string of all schema names in the CURRENT PATH cannot exceed 254 characters. A SET CURRENT PATH statement causing this message would exceed this limit.

System Action: The statement is not executed.

Programmer Response: Remove schema names to reduce the total length to fit the 254 character maximum length. If all the schema names are required, it may be necessary to consolidate some user-defined functions so that fewer schema names are required for the CURRENT PATH.

SQLSTATE: 42907

-587 **A LIST OF *item-references* ARE NOT IN THE SAME FAMILY**

Explanation: Each *item-reference* in the SET statement is either a *host-variable* or a *transition-variable*. The list of *item-references* must be of the same family, meaning if one of the *item-references* is a *transition-variable*, then all of the *item-references* in the list must be a *transition-variable*. If the statement is used in the triggered action of a CREATE TRIGGER statement, each *item-reference* must identify a *transition-variable*. If the statement is used in any other context, each *item-reference* must identify a *host-variable*.

System Action: The statement can not be processed.

Programmer Response: Correct the statement and execute it again.

SQLSTATE: 428C6

-590 **PARAMETER NAME *parameter-name* IS NOT UNIQUE IN THE CREATE FOR ROUTINE *routine-name***

Explanation: The parameter name *parameter-name* specified on a CREATE FUNCTION or CREATE PROCEDURE statement for *routine-name* is not unique.

System Action: The statement cannot be executed.

Programmer Response: Change the name of the parameter to make all of the parameter names unique within the CREATE statement.

SQLSTATE: 42734

-592 **NOT AUTHORIZED TO CREATE FUNCTIONS OR PROCEDURES IN WLM ENVIRONMENT** *env-name*

Explanation: This mesage is issued when:

- The value of the *env-name* token is 'NO WLM ENVIRONMENT' and the check for authorization to the DB2-managed stored procedures address space failed because the NO WLM ENVIRONMENT clause was specified on the CREATE PROCEDURE or ALTER PROCEDURE statement.
- There is no DB2-managed stored procedures address space.

System Action: The statement cannot be executed.

Programmer Response: To correct the error, perform one of the following actions:

- If the value of the *env-name* token is 'NO WLM ENVIRONMENT', choose a different value for the WLM ENVIRONMENT keyword or request authorization to create objects in the specified WLM ENVIRONMENT from the system administrator.
- If there is no DB2-managed stored procedures address space, request that one be created. Also request that a RACF PERMIT be completed to allow access to this resource.

After doing one of the above, reissue the SQL statement.

SQLSTATE: 42510

-593 **NOT NULL MUST BE SPECIFIED FOR ROWID OR DISTINCT TYPE COLUMN** *column-name*

Explanation: ROWID columns and distinct type columns for which the source type is a ROWID do not support null values. When a ROWID column (or distinct type for which the source type is a ROWID) is defined on a CREATE TABLE, ALTER TABLE, or DECLARE TABLE statement, the NOT NULL clause must be specified for the column.

System Action: The statement cannot be executed.

Programmer Response: Change the statement to specify NOT NULL for ROWID column *column-name*.

SQLSTATE: 42831

-601 **THE NAME OF THE OBJECT TO BE CREATED OR THE TARGET OF A RENAME STATEMENT IS IDENTICAL TO THE EXISTING NAME** *name* **OF THE OBJECT TYPE** *obj-type*

Explanation: One of the following situations has been detected:

- A CREATE statement tried to create an object *name* of type *obj-type*, but an object of that type with the same name is already defined in the DB2 subsystem.
  - If *obj-type* is CONSTRAINT, the *name* was specified in the FOREIGN KEY clause, CHECK clause, PRIMARY KEY clause, or UNIQUE clause of either a CREATE or ALTER TABLE statement. All referential integrity, check constraint, primary key, and unique key constraint names defined on a table must be unique.
  - If *obj-type* is TABLE or VIEW, and a CREATE ALIAS statement failed, then the alias-name specified in the CREATE ALIAS statement is identical to the table name or view name specified. The TABLE or VIEW might not exist in the DB2 subsystem.
  - If *obj-type* is DISTINCT TYPE, and a CREATE DISTINCT TYPE statement failed, then there is already a user-defined type existing with the same name as the distinct type name specified in the CREATE DISTINCT TYPE statement.
  - If *obj-type* is FUNCTION or PROCEDURE, and a CREATE FUNCTION or CREATE PROCEDURE statement failed, then there is already a routine existing with the same name as the name specified in the CREATE FUNCTION or CREATE PROCEDURE statement.
  - If *obj-type* is FUNCTION, and a CREATE DISTINCT TYPE statement failed, then there is already a cast function existing with the same name as the distinct type and an input parameter of the source type or there is already a cast function existing with the same name as the source type and an input parameter of the distinct type name.
- A RENAME statement specified a target name *name*, but an object with the same name is already defined in the DB2 subsystem.

System Action: The CREATE, ALTER or RENAME statement cannot be executed. No new object was created, no existing object was altered, and no existing object was renamed.

Programmer Response: Either drop the existing object or choose another name. If *obj-type* is data set, do an IDCAMS DELETE of the data set before retrying the CREATE. Refer to Chapter 3 of *DB2 SQL Reference* for information about the scope of object names in DB2.

SQLSTATE: 42710

-602 **TOO MANY COLUMNS SPECIFIED IN A CREATE INDEX**

Explanation: The number of columns specified in the CREATE INDEX statement exceeds 64, the maximum permitted by DB2.

System Action: The statement cannot be executed.

The specified index was not created.

Programmer Response: The index definition must be modified to conform to the system-imposed column limit of 64.

SQLSTATE: 54008

-603 **A UNIQUE INDEX CANNOT BE CREATED BECAUSE THE TABLE CONTAINS ROWS WHICH ARE DUPLICATES WITH RESPECT TO THE VALUES OF THE IDENTIFIED COLUMNS**

Explanation: The index defined in the CREATE INDEX statement could not be created as unique because the specified table already contains rows that are duplicates with respect to the values of the identified columns.

System Action: The statement cannot be executed.

Programmer Response: Examine the data to ascertain whether or not the duplicate data is valid. Alternatively, consider creating a nonunique index.

SQLSTATE: 23515

-604 **A DATA TYPE DEFINITION SPECIFIES AN INVALID LENGTH, PRECISION, OR SCALE ATTRIBUTE**

Explanation: A data type definition in a CREATE or ALTER statement contains an invalid length, precision, or scale attribute specification. In addition, the specification of data type might be incorrect or invalid. Or, the column definition in a view referenced in a CREATE TABLE LIKE *view* has an invalid length.

System Action: The statement cannot be executed. The specified object was not created or altered.

Programmer Response: Correct the syntax, and resubmit the statement. Refer to Chapter 3 of *DB2 SQL Reference* for information about valid length, precision, and scale attributes for the data type of an object.

SQLSTATE: 42611

-607 **OPERATION OR OPTION** *operation* **IS NOT DEFINED FOR THIS OBJECT**

Explanation: The operation or option cannot be performed on the object specified in the SQL statement.

System Action: The statement cannot be executed.

Programmer Response: If an option of the SQL statement is not allowed for this object, modify the SQL statement and resubmit the statement. If an operation is not defined for the object, the statement cannot be executed.

SQLSTATE: 42832

-611 **ONLY LOCKMAX 0 CAN BE SPECIFIED WHEN THE LOCK SIZE OF THE TABLESPACE IS TABLESPACE OR TABLE**

Explanation: This message is issued when:

- The LOCKSIZE of the table space is TABLESPACE or TABLE, and LOCKMAX is being altered to or created as a nonzero value.
- The LOCKSIZE of the table space is being altered to TABLESPACE or TABLE, and LOCKMAX is being altered to or created as a nonzero value.

If LOCKSIZE is TABLESPACE or TABLE, LOCKMAX must be 0 because lock escalation is not supported from these levels.

System Action: The statement cannot be executed.

Programmer Response: Do one of the following:

- Reissue the statement with LOCKMAX 0.
- Alter the LOCKSIZE of the table space to a value other than TABLESPACE or TABLE.

SQLSTATE: 53088

-612 *column-name* **IS A DUPLICATE COLUMN NAME**

Explanation: The CREATE INDEX, CREATE TABLE, CREATE VIEW or ALTER TABLE statement specifies the same *column-name* for two (or more) columns of the index, table, view, or the UPDATE OF clause of a trigger definition specifies the same column name more than once. Column names must be unique within an index, a table, a view, or in the UPDATE OF clause of a trigger definition. A column cannot be specified in more than one ALTER TABLE clause except if it is specified in an ALTER COLUMN clause and ADD CHECK CONSTRAINT clause.

System Action: The statement cannot be executed. The specified index, table, view, or trigger was not created.

Programmer Response: Correct the CREATE statement to specify unique names for each of the columns of the index, table, view, or the columns in the UPDATE OF clause of a trigger definition. Correct the ALTER statement to specify unique names for each of the ALTER COLUMN clauses.

This error can also occur on CREATE TABLE when a column list of a PRIMARY KEY, FOREIGN KEY, or UNIQUE clause contains two or more occurrences of the same column name.

SQLSTATE: 42711

-613 **THE PRIMARY KEY OR A UNIQUE CONSTRAINT IS TOO LONG OR HAS TOO MANY COLUMNS**

Explanation: In a list of columns following either PRIMARY KEY or UNIQUE, the number of columns is greater than 64 or the sum of the column length attributes is greater than the number allowed for the type of index.

System Action: The CREATE or ALTER statement cannot be executed. The specified table cannot be created or altered.

Programmer Response: Change the table definition to keep within the prescribed limits.

SQLSTATE: 54008

-614 **THE INDEX CANNOT BE CREATED OR THE LENGTH OF A COLUMN CANNOT BE CHANGED BECAUSE THE SUM OF THE INTERNAL LENGTHS OF THE IDENTIFIED COLUMNS IS GREATER THAN THE ALLOWABLE MAXIMUM**

Explanation: The index could not be created or the length of a column cannot be changed because the sum of the *internal* lengths of the key columns would exceed the allowable maximum. The maximum permitted key length is 255.

System Action: The statement cannot be executed. The specified index was not created or the length of the column was not changed.

Programmer Response: The definition for the index must be modified (possibly by eliminating one or more key columns) to reduce the length of the key to the permitted maximum. Refer to Chapter 6 of *DB2 SQL Reference* if you require a complete explanation of other possible maximum key lengths and how they are computed.

SQLSTATE: 54008

-615 *operation-type* **IS NOT ALLOWED ON A PACKAGE IN USE**

Explanation: The operation 'operation-type' cannot be performed because the package is in use by the same application process.

operation-type
Type of bind operation (BIND, REBIND or DROP).

System Action: The BIND, REBIND, or DROP operation on the package is not performed.

Programmer Response: Change the application to invoke the BIND, REBIND or DROP operation when the package is not use by the same application process.

SQLSTATE: 55006

-616 *obj-type1 obj-name1* **CANNOT BE DROPPED BECAUSE IT IS REFERENCED BY** *obj-type2 obj-name2*

Explanation: Some types of objects cannot be dropped if there are other objects which are dependent upon them. For example, a storage group cannot be dropped if there are one or more existing table spaces that use that storage group.

Execution of the specified DROP statement would drop object *obj-name1* of type *obj-type1* on which object *obj-name2* of type *obj-type2* is dependent.

System Action: The statement cannot be processed. The specified object was not dropped.

Programmer Response: Verify that the object specified in the DROP statement was, indeed, the object to be dropped. If so, all the existing objects that have a dependency on that object must first be dropped.

A LOB table space cannot be dropped when an association exists between it and another table space. The associated base table must be dropped first.

A populated auxiliary table and its index can only be dropped by dropping the associated base table.

A trigger package cannot be explicitly dropped. It can only be dropped by dropping the associated trigger with a DROP TRIGGER statement or by dropping the triggering table.

SQLSTATE: 42893

-617 **A TYPE 1 INDEX IS NOT VALID FOR TABLE** *table-name*

Explanation: A TYPE 1 index cannot be created on the following tables:

- A table within a large table space
- A table within a table space with LOCKSIZE ROW
- An auxiliary table

In addition, a type 1 index cannot be created on a ROWID column of a table.

System Action: The statement cannot be executed. The index was not created

Programmer Response: Either create the index as TYPE 2 or, if LOCKSIZE ROW was used,alter the LOCKSIZE of the table space containing the table to a value other than LOCKSIZE ROW.

SQLSTATE: 56089

-618 **OPERATION** *operation* **IS NOT ALLOWED ON SYSTEM DATABASES**

Explanation: System databases cannot be the object of certain types of operations. The attempted 'operation' cannot be performed on system databases. One

possible reason for this error is that CCSID ASCII or CCSID UNICODE was specified when a system database was being created.

System Action: The statement cannot be executed. No changes were made to the specifie system database.

Programmer Response: Do not attempt to perform the requested operation on system databases.

SQLSTATE: 42832

-619 **OPERATION DISALLOWED BECAUSE THE DATABASE IS NOT STOPPED**

Explanation: The statements CREATE, ALTER or DROP for a table space, table, or index in the database cannot be processed unless the database is stopped (using the STOP command).

System Action: The statement cannot be processed.

Programmer Response: Issue the -DISPLAY DATABASE command to verify that the work file database is stopped before resubmitting the statement.

SQLSTATE: 55011

-620 **KEYWORD** *keyword* **IN** *stmt type* **STATEMENT IS NOT PERMITTED FOR A** *space type* **SPACE IN THE** *database type* **DATABASE**

Explanation: The specified *keyword* in the SQL statement *stmt type* indicates an attribute that is not allowed for a *space type* space in the *database type* database.

keyword
Specifies the keyword that is not allowed.

stmt type
CREATE or ALTER

CREATE is for CREATE TABLESPACE or CREATE INDEX.

ALTER is for ALTER TABLESPACE or ALTER INDEX.

space type
TABLE or INDEX

TABLE is for table space, and INDEX is for index space.

database type
WORK FILE or TEMP

System Action: The statement cannot be executed.

Programmer Response: Refer to *DB2 SQL Reference* for information about attributes that are allowed or not allowed for a *space type* space in a *database type* database. Correct and resubmit the *stmt type* statement.

SQLSTATE: 53001

-621 **DUPLICATE DBID** *dbid* **WAS DETECTED AND PREVIOUSLY ASSIGNED TO** *database-name*

Explanation: The current database being created was assigned a DBID of 'dbid', which is identical to the DBID assigned to database 'database-name'. An inconsistency exists between the DB2 catalog and directory.

System Action: The statement cannot be executed. No new object was created, and the existing object was not altered or modified.

Programmer Response: Notify the system programmer. The inconsistency must be corrected before CREATE DATABASE will be successful.

System Programmer Response: If you suspect an error in DB2, refer to Part 2 of *DB2 Diagnosis Guide and Reference* for information on identifying and reporting the problem.

SQLSTATE: 58001

-622 *FOR MIXED DATA* **IS INVALID BECAUSE THE MIXED DATA INSTALL OPTION IS NO**

Explanation: FOR MIXED DATA is specified in a column description of a CREATE or ALTER TABLE, a CREATE FUNCTION, or a CREATE PROCEDURE statement, but the MIXED DATA install option is set to NO. FOR MIXED DATA is valid only when the MIXED DATA install option is set to YES.

This message will only be issued for encoding scheme ASCII or EBCDIC. Mixed data is always allowed with the UNICODE encoding scheme

System Action: The statement is not executed.

Programmer Response: Either change the install option or the FOR clause. If the install option is correctly set to NO, the allowable FOR clause options are BIT and SBCS.

SQLSTATE: 56031

-623 **A CLUSTERING INDEX ALREADY EXISTS ON TABLE** *table-name*

Explanation: The CREATE INDEX statement would create a second cluster index on the specified table. A given table can have only one cluster index.

System Action: The statement cannot be executed.

Programmer Response: Check to determine the identity and validity of the existing cluster index on the object table. Alternatively, consider creating the index without the CLUSTER attribute.

SQLSTATE: 55012

-624 **TABLE *table-name* ALREADY HAS A PRIMARY KEY OR UNIQUE KEY CONSTRAINT WITH SPECIFIED COLUMNS**

Explanation: The code is used to report that a primary key or unique key cannot be defined in an ALTER TABLE statement because the table either:

- Already has a primary key, or
- Has an existing unique constraint with the same definition (same set of columns specified) as the new key.

System Action: The statement cannot be run.

Programmer Response: Do not attempt to define a table with more than one primary key, or a unique constraint that duplicates the definition of an existing unique constraint.

SQLSTATE: 42889

-625 **TABLE *table-name* DOES NOT HAVE AN INDEX TO ENFORCE THE UNIQUENESS OF THE PRIMARY OR UNIQUE KEY**

Explanation: The ALTER TABLE statement is invalid because the table does not have a unique index with a key that is identical to the nominated primary or unique key.

System Action: The statement cannot be executed.

Programmer Response: Make sure the key list specified on the ALTER TABLE statement identifies an existing unique index of the table.

SQLSTATE: 55014

-626 **THE ALTER STATEMENT IS NOT EXECUTABLE BECAUSE THE PAGE SET IS NOT STOPPED**

Explanation: An ALTER statement specifies a BUFFERPOOL, USING, PRIQTY, SECQTY, ERASE, or GBPCACHE clause, but the page set is not stopped.

System Action: The SQL statement cannot be executed.

Programmer Response: Stop the page set before resubmitting the statement.

SQLSTATE: 55015

-627 **THE ALTER STATEMENT IS INVALID BECAUSE THE PAGESET HAS USER-MANAGED DATA SETS**

Explanation: This code is used if a PRIQTY, SECQTY, or ERASE clause is specified, USING STOGROUP is not specified, and the page set has user-managed data sets.

System Action: The SQL statement cannot be executed.

Programmer Response: Verify that the correct table or partition is specified. The primary and secondary space allocation of a user-managed data set cannot be altered by means of an ALTER statement.

SQLSTATE: 55016

-628 **THE CLAUSES ARE MUTUALLY EXCLUSIVE**

Explanation: Mutually exclusive clauses were specified in one or more of the following ways:

- A CREATE TABLESPACE statement contains both the SEGSIZE and NUMPARTS clauses.
- A CREATE TABLESPACE statement contains both the SEGSIZE and LARGE clauses.
- A CREATE TABLESPACE statement contains both the SEGSIZE and MEMBER CLUSTER clauses.
- A CREATE or ALTER TABLESPACE contains both the LOCKPART YES and LOCKSIZE TABLESPACE
- A 'column-definition' contains both NOT NULL and DEFAULT NULL clauses.
- A 'column-definition' contains both FIELDPROC and DEFAULT clauses.
- A select-statement contains both the update-clause and the FOR FETCH ONLY clause.
- An ALTER TABLE statement contains both a DROP CONSTRAINT clause and either a DROP FOREIGN KEY, DROP CHECK, DROP PRIMARY KEY, or DROP UNIQUE clause.
- A CREATE or ALTER TABLESPACE statement contains both LOCKPART YES and LOCKSIZE TABLESPACE.
- A CREATE TRIGGER statement specifies more than one correlation name for OLD, NEW, OLD_TABLE, or NEW_TABLE. Each of these correlation specifications can appear no more than once in the CREATE TRIGGER statement.
- A CREATE FUNCTION statement contains both a CAST FROM clause and a SOURCE clause.
- A CREATE FUNCTION statement contains both a SOURCE clause and a RETURNS TABLE clause.
- A CREATE FUNCTION statement contains both a SOURCE clause and a clause used to define an external function (For example, EXTERNAL, LANGUAGE, NO SQL).
- A CREATE or ALTER PROCEDURE statement attempts to use the NO WLM ENVIRONMENT and PROGRAM TYPE SUB options. When NO WLM ENVIRONMENT is used, then SECURITY must also be used.
- A CREATE or ALTER PROCEDURE statement attempts to use both NO WLM ENVIRONMENT and

either USER or DEFINER for SECURITY. When NO WLM ENVIRONMENT is used, then SECURITY DB2 must also be used.

- A CREATE or ALTER PROCEDURE statement contains both a LANGUAGE REXX clause, and a PARAMETER STYLE DB2SQL or PARAMETER STYLE JAVA clause.
- An ALTER TABLE statement contains both an ALTER COLUMN clause and a VALIDPROC clause.
- An ALTER TABLE statement contains both an ALTER COLUMN clause and a clause other than the check constraint clause.
- The AS (*subselect*) clause of a DECLARE GLOBAL TEMPORARY TABLE statement contains both an INCLUDING COLUMN DEFAULTS clause and a USING TYPE DEFAULTS clause.
- A CREATE DATABASE statement contains both the AS WORKFILE clause and the AS TEMP clause.
- If INSENSITIVE or SENSITIVE is specified, then SCROLL must also be specified, either on DECLARE CURSOR or with the ATTRIBUTES clause of the PREPARE statement.
- If SCROLL is specified, then either INSENSITIVE or SENSITIVE STATIC must also be specified, either on DECLARE CURSOR or with the ATTRIBUTES clause of the PREPARE statement.
- The *attribute-string* specified in the ATTRIBUTES clause of the PREPARE statement cannot specify conflicting options.

Programmer Response: Change the options specified in the statement, and reissue the statement.

SQLSTATE: 42613

-629 **SET NULL CANNOT BE SPECIFIED BECAUSE FOREIGN KEY *name* CANNOT CONTAIN NULL VALUES**

Explanation: The code SET NULL option of the indicated FOREIGN KEY clause is invalid because no column of the key allows null values. The *name* is the constraint-name specified in the FOREIGN KEY clause or, if a constraint-name was not specified, the first column-name specified in the clause.

System Action: The statement cannot be processed.

Programmer Response: Change a column of the key to allow null values or change the delete rule.

SQLSTATE: 42834

-630 **THE WHERE NOT NULL SPECIFICATION IS INVALID FOR TYPE 1 INDEXES**

Explanation: Type 1 indexes cannot be created with the WHERE NOT NULL specification.

System Action: The statement cannot be executed.

Programmer Response: Either the index must be created as a type 2 index, or the WHERE NOT NULL specification must not be used.

SQLSTATE: 56089

-631 **FOREIGN KEY *name* IS TOO LONG OR HAS TOO MANY COLUMNS**

Explanation: This code is used to report that the sum of the length attributes of the columns identified in the indicated FOREIGN KEY clause is greater than 255 bytes or the number of columns identified is greater than 64. The 'name' is the constraint-name specified in the FOREIGN KEY clause or, if a constraint-name was not specified, the first column-name specified in the clause.

System Action: The statement cannot be executed.

Programmer Response: The table definition must be modified to conform to the system-imposed limit of the sum of the length attributes of the columns identified in the PRIMARY KEY clause.

SQLSTATE: 54008

-632 **THE TABLE CANNOT BE DEFINED AS A DEPENDENT OF *table-name* BECAUSE OF DELETE RULE RESTRICTIONS**

Explanation: This code is used to report that the object of an ALTER TABLE statement cannot be defined as a dependent of the indicated table because either:

- The relationship would form a cycle that would cause the table to be delete-connected to itself.
- The relationship would cause the table to be delete-connected to the indicated table through multiple paths and the delete rule of the existing relationship is SET NULL.

The error is due to the delete rules of existing relationships, not the delete rule specified in the FOREIGN KEY clause of the ALTER TABLE statement.

System Action: The statement cannot be executed.

Programmer Response: Eliminate the particular FOREIGN KEY clause from the ALTER or CREATE TABLE statement.

SQLSTATE: 42915

-633 **THE DELETE RULE MUST BE *delete-rule***

Explanation: The code is used to report that the 'delete-rule' specified in a FOREIGN KEY clause of the ALTER TABLE statement is invalid. The indicated 'delete-rule' is required because:

- A self-referencing constraint must have a 'delete-rule' of CASCADE or NO ACTION.

- The relationship would cause the table to be delete-connected to the same table through multiple paths and such relationships must have the same 'delete-rule'.

System Action: The statement cannot be executed.

Programmer Response: Change the 'delete rule' in the FOREIGN KEY clause.

SQLSTATE: 42915

-634 THE DELETE RULE MUST NOT BE CASCADE

Explanation: The code is used to report that the CASCADE delete rule specified in the FOREIGN KEY clause of an ALTER TABLE statement is invalid because:

- The relationship would form a cycle that would cause a table to be delete-connected to itself.
- The relationship would cause another table to be delete-connected to the same table through multiple paths with different delete rules or with a delete rule equal to SET NULL.

System Action: The statement cannot be executed.

Programmer Response: Change the delete rule.

SQLSTATE: 42915

-635 THE DELETE RULES CANNOT BE DIFFERENT OR CANNOT BE SET NULL

Explanation: The code is used to report that the delete rules specified in two FOREIGN KEY clauses of the CREATE TABLE statement are invalid because the table would be delete-connected to the same table through multiple paths involving relationships with different delete rules or with delete rules of SET NULL.

System Action: The statement cannot be executed.

Programmer Response: Change the delete rule.

SQLSTATE: 42915

-636 THE PARTITIONING KEYS FOR PARTITION *part-num* ARE NOT SPECIFIED IN ASCENDING OR DESCENDING ORDER

Explanation: In the CREATE INDEX or ALTER INDEX statement for the CLUSTER index for a partitioned table (that is, a table residing in a partitioned table space), the partitioning key values specified in the limit key value specifications were not in either ascending or descending order.

System Action: The statement cannot be executed. The specified cluster index was not created.

Programmer Response: Correct the limit key value specifications in the CREATE INDEX or ALTER INDEX statement for the identified partitionso that the limit key values for successive partitions are in strictly ascending or descending order.

SQLSTATE: 56016

-637 DUPLICATE *keyword* KEYWORD

Explanation: The SQL statement contains a duplicate specification for the keyword *keyword*. For example:

- DEFAULT, UNIQUE, and PRIMARY can only be specified once in a column definition.
- UNIQUE and PRIMARY cannot both be specified for the same column definition.
- PRIMARY can only be specified once in a CREATE TABLE statement.
- The *attribute-string* specified in the ATTRIBUTES clause of the PREPARE statement cannot specify an option more than once.

System Action: The statement cannot be processed.

Programmer Response: Correct the statement by removing duplicate clauses.

SQLSTATE: 42614

-638 TABLE *table-name* CANNOT BE CREATED BECAUSE COLUMN DEFINITION IS MISSING

Explanation: The CREATE TABLE statement does not contain any column definition.

System Action: The SQL statement cannot be executed.

Programmer Response: Add column definition to the statement.

SQLSTATE: 42601

-639 A NULLABLE COLUMN OF A FOREIGN KEY WITH A DELETE RULE OF SET NULL CANNOT BE A COLUMN OF THE KEY OF A PARTITIONED INDEX

Explanation: A partition key of the clustering index cannot be updated. Therefore, a foreign key column with a delete rule of SET NULL cannot be a column of a partition key if that column is nullable. If this error occurs for an ALTER TABLE operation, the foreign key cannot be created. If this error occurs for a CREATE INDEX operation, the index cannot be created.

System Action: The statement cannot be executed.

Programmer Response: Review the delete rule of the referential constraint and the partition keys for the index. Do one of the following:

- If the operation in error was CREATE INDEX, either change the index partition key definition or drop and redefine the referential constraint with a different delete rule.
- If the operation in error was ALTER TABLE, change the referential delete rule so that all nullable index keys are not part of the foreign keys.

SQLSTATE: 56027

-640 **LOCKSIZE ROW CANNOT BE SPECIFIED BECAUSE TABLE IN THIS TABLESPACE HAS TYPE 1 INDEX**

Explanation: If LOCKSIZE ROW is specified for a table space, all indexes on tables in the table space must be type 2 indexes. The following SQL statement identifies all the type 1 indexes:

```
SELECT I.CREATOR, I.NAME
FROM SYSIBM.SYSINDEXES I,
SYSIBM.SYSTABLES T
WHERE INDEXTYPE = ' '
AND T.TSNAME = 'table_space_name'
AND T.DBNAMe = 'database_name'
AND T.CREATOR = I.TBCREATOR
AND T.NAME = I.TBNAME;
```

where '*table_space_name*' is the name of the table space that is to be altered; '*database_name*' is the name of the database that contains the table space.

System Action: The statement cannot be executed.

Programmer Response: Since the LOCKSIZE ROW on the table space and the type 1 indexes conflict, either use the ALTER INDEX statement to convert all type 1 indexes to type 2 indexes or use another LOCKSIZE option.

SQLSTATE: 56089

-643 **CHECK CONSTRAINT EXCEEDS MAXIMUM ALLOWABLE LENGTH**

Explanation: The check constraint definition exceeds the maximum allowable limit of 3800 characters. The redundant blank spaces are excluded from this limit.

System Action: The CREATE TABLE or ALTER TABLE statement failed.

Programmer Response: Rewrite the check constraint definition so that it is less than 3800 characters. You might need to divide the check constraint into two or more smaller check constraints.

SQLSTATE: 54024

-644 **INVALID VALUE SPECIFIED FOR KEYWORD** *keyword* **IN** *stmt-type* **STATEMENT**

Explanation: The value specified for the 'keyword' parameter in the 'stmt-type' SQL statement is not a permitted value.

System Action: The SQL statement cannot be executed.

Programmer Response: Correct the statement. Refer to Chapter 6 of *DB2 SQL Reference* for information about the permissible values for the 'keyword' keyword in 'stmt-type' statements.

SQLSTATE: 42615

-646 **TABLE** *table-name* **CANNOT BE CREATED IN SPECIFIED TABLE SPACE** *table-space-name* **BECAUSE IT ALREADY CONTAINS A TABLE**

Explanation: The table space specified in a CREATE TABLE statement is a partitioned, default, or LOB table space in which an existing table already resides. Only one table may reside in a partitioned, default, or LOB table space.

System Action: The statement cannot be executed. The specified table was not created.

Programmer Response: Verify that the correct table space was specified in the CREATE statement. Do not attempt to create more than one table in a partitioned, default, or LOB table space.

SQLSTATE: 55017

-647 **BUFFERPOOL** *bp-name* **CANNOT BE SPECIFIED BECAUSE IT HAS NOT BEEN ACTIVATED**

Explanation: The buffer pool specified in a CREATE or ALTER statement for a table space or index (index space) is not activated.

Table spaces and indexes (index spaces) can only be assigned or reassigned to buffer pools that are currently activated.

System Action: The statement cannot be executed. The specified table space or index space was not created or altered.

Programmer Response: Verify that the proper buffer pool was specified in the CREATE or ALTER statement. Use the -DISPLAY BUFFERPOOL command to display the attributes of the buffer pool and determine if the buffer pool is activated. If the buffer pool is not activated, use the -ALTER BUFFERPOOL command to change the VPSIZE from 0 to the desired size.

SQLSTATE: 57003

-650 **THE ALTER INDEX CANNOT BE EXECUTED, REASON** *reason*

Explanation: The ALTER INDEX statement cannot be executed for one of the following reasons:

1 Alter to type 1 index is not allowed for the index whose associated table space has a LOCKSIZE specification of ROW.

2 Alter to type 1 index is not allowed for the index defined with UNIQUE WHERE NOT NULL.

3 Alter to type 1 index is not allowed for the index whose associated table space has been defined as a LARGE table space.

4 Alter to type 1 index is not allowed for an index on an ASCII table.

5 Alter PIECESIZE is not allowed for a partitioning index.

6 Alter PIECESIZE 4G is not allowed for non-partitioned indexes on a non-LARGE table.

11 Alter VALUES is not allowed for an index on a partitioned base table with LOB columns.

System Action: The ALTER INDEX statement is not executed.

Programmer Response: Correct the error according to the given reason and execute the statement again.

SQLSTATE: 56090

-651 **TABLE DESCRIPTION EXCEEDS MAXIMUM SIZE OF OBJECT DESCRIPTOR.**

Explanation: The CREATE TABLE or ALTER TABLE statement causes the table descriptor (record OBD) to exceed the object descriptor size limit of 32KB.

Programmer Response: Change the statement by reducing either the number or length (or a combination of both) of the user-defined default string constants or check constraints and execute the statement again.

System Action: The statement is not executed. For an ALTER TABLE statement, the definition of the table is unchanged. For a CREATE TABLE statement, the table is not created.

SQLSTATE: 54025

-652 **VIOLATION OF INSTALLATION DEFINED EDIT OR VALIDATION PROCEDURE** *proc-name*

Explanation: The result of the SQL statement has been rejected by the installation defined edit or validation procedure 'proc-name' for the object table.

System Action: The statement cannot be executed. The contents of the object table were not modified.

Programmer Response: Determine the requirements imposed by the edit or validation procedure for inserts and updates of the object table.

SQLSTATE: 23506

-653 **TABLE** *table-name* **IN PARTITIONED TABLE SPACE** *tspace-name* **IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT BEEN CREATED**

Explanation: An attempt has been made to insert or manipulate data in or create a view on a partitioned table (that is, a table residing in a partitioned table space) before the partitioned index for that table has been created.

A table residing in a partitioned table space cannot be referenced in any SQL manipulative statement or a CREATE VIEW statement before the partitioned index for that table has been created.

System Action: The statement cannot be executed.

Programmer Response: Verify that the correct table was specified in the statement. If so, ensure that the partitioned index for the table has been created successfully before attempting to execute any SQL manipulative statements that reference that table.

SQLSTATE: 57004

-655 **THE CREATE OR ALTER STOGROUP IS INVALID BECAUSE THE STORAGE GROUP WOULD HAVE BOTH SPECIFIC AND NON-SPECIFIC VOLUME IDS**

Explanation: One of the following error conditions occurred:

- Both a specific and a non-specific ('*') volume ID are specified in the VOLUMES clause of a CREATE STOGROUP statement.
- Both a specific and a non-specific ('*') volume ID are specified in an ADD VOLUMES clause of an ALTER STOGROUP statement.
- A specific volume ID is specified in an ADD VOLUMES clause of an ALTER of a storage group that has non-specific volume IDs or mixed volume IDs.
- A non-specific volume ID ('*') is specified in an ADD VOLUMES clause of an ALTER of a storage group that has specific volume IDs or mixed volume IDs.

System Action: The statement is not executed.

Programmer Response: Specify either specific or non-specific volume IDs in the VOLUMES clause of CREATE STOGROUP statement and the ADD VOLUMES clause of the ALTER STOGROUP statement. To add specific volume IDs to a storage

group with non-specific volume IDs, use the REMOVE VOLUMES clause to remove the non-specific volume IDs. To add non-specific volume IDs to a storage group with specific volume IDs, use the REMOVE VOLUMES clause to remove the specific volume IDs.

SQLSTATE: 56036

-658 **A *object-type* CANNOT BE DROPPED USING THE *statement* STATEMENT**

Explanation: A DROP statement was issued, but the object cannot be explicitly dropped. The object must be dropped by dropping an associated object:

TRIGGER PACKAGE
A *trigger package* cannot be dropped with the DROP PACKAGE statement. A trigger package can only be dropped implicitly when the associated trigger is dropped using the DROP TRIGGER statement.

CAST FUNCTION
A *cast function* cannot be explicitly dropped with the DROP FUNCTION statement. A cast function can only be dropped implicitly when the associated distinct type is dropped using the DROP DISTINCT TYPE statement.

System Action: The SQL statement cannot be executed.

Programmer Response: Issue the appropriate DROP statement to drop the intended objects.

SQLSTATE: 42917

-660 **INDEX *index-name* CANNOT BE CREATED OR ALTERED ON PARTITIONED TABLE SPACE *tspace-name* BECAUSE KEY LIMITS ARE NOT SPECIFIED**

Explanation: The CREATE INDEX or ALTER INDEX statement did not specify limit key values for the partitions of the table space. To create a clustering index for a table in a partitioned table space, or to modify those values using ALTER INDEX, you must include those values.

System Action: The statement cannot be executed. The specified cluster index was not created or altered.

Programmer Response: Verify that the correct table was specified in the CREATE INDEX or ALTER INDEX statement. If so, the definition for the partitioned table space must be examined so that a proper definition for the cluster index for the table may be constructed. Refer to Chapter 6 of *DB2 SQL Reference* for information about the requirements that must be satisfied by the definitions for the cluster indexes for partitioned tables.

SQLSTATE: 53035

-661 **INDEX *index-name* CANNOT BE CREATED ON PARTITIONED TABLE SPACE *tspace-name* BECAUSE THE NUMBER OF PART SPECIFICATIONS IS NOT EQUAL TO THE NUMBER OF PARTITIONS OF THE TABLE SPACE**

Explanation: The CREATE INDEX statement for the cluster index on a partitioned table (that is, a table residing in a partitioned table space) does not contain the same number of PART specifications as there are partitions in the table space. The definition for the cluster index for a partitioned table must contain exactly as many PART specifications as there are partitions in the table space in which that table resides. Also, the part numbers must be valid and unique.

System Action: The statement cannot be executed. The specified cluster index was not created.

Programmer Response: Examine the definition of the partitioned table space to determine how many partitions have been specified, and then correct the syntax of the CREATE INDEX statement to provide the proper number of PART specifications. Refer to Chapter 6 of *DB2 SQL Reference* for information about the definitions for cluster indexes on partitioned tables.

SQLSTATE: 53036

-662 **A PARTITIONED INDEX CANNOT BE CREATED ON A NON-PARTITIONED TABLE SPACE *tspace-name***

Explanation: The CREATE INDEX statement contains PART specifications, but the specified object table is not partitioned (that is, does not reside in a partitioned table space).

System Action: The statement cannot be executed. The specified index was not created.

Programmer Response: Verify that the proper object table was specified in the statement. Refer to Chapter 6 of *DB2 SQL Reference* for information about the proper usage of PART specifications in CREATE INDEX statements.

SQLSTATE: 53037

-663 **THE NUMBER OF KEY LIMIT VALUES IS EITHER ZERO, OR GREATER THAN THE NUMBER OF COLUMNS IN THE KEY OF INDEX *index-name***

Explanation: The number of limit key value specifications provided in at least one of the PART specifications of the CREATE INDEX or ALTER INDEX statement is either zero or greater than the number of columns in the index key.

System Action: The statement cannot be executed. The specified index was not created.

Programmer Response: Correct the statement so

that each PART specification contains exactly the same number of limit key value specifications as there are columns in the index key.

SQLSTATE: 53038

-665 THE PART CLAUSE OF AN ALTER STATEMENT IS OMITTED OR INVALID

Explanation: The ALTER statement is invalid for one of the following reasons:

- The table space or index is not partitioned and the PART clause is specified.
- The table space or index is partitioned, a partition attribute (FREEPAGE or PCTFREE) is specified, and the PART clause is not specified.
- The integer specified in the PART clause does not identify a partition of the table space or index.
- A USING, PRIQTY, SECQTY, or ERASE clause is used to alter storage attributes, but the partition is not specified.
- A GBPCACHE clause is used to alter the group buffer pool caching attributes, but the partition is not specified.
- The VALUES clause is specified without the PART clause. You must specify PART to change VALUES.

System Action: The SQL statement cannot be executed.

Programmer Response: Determine whether the table space or index you want to alter is partitioned. If it is partitioned, specify a PART clause that identifies the partition you want to alter. If it is not partitioned, do not specify the PART clause.

SQLSTATE: 53039

-666 *stmt-verb object* CANNOT BE EXECUTED BECAUSE *function* IS IN PROGRESS

Explanation: The SQL statement could not be executed because the named function was executing at the time.

stmt-verb
The type of data definition language (DDL) statement

object The DB2 object type

function
A utility, the governor, or the distributed data facility (DDF)

If the object is part of the communications database, it cannot be dropped while the DDF is active.

System Action: The statement was not executed.

Programmer Response: If the function is a utility, wait for the function to complete or stop. Then resubmit the statement for execution.

If the function is the governor, the statement cannot be executed until the resource limit facility is stopped or switched to a different resource limit specification table (RLST). In a DB2 data sharing environment, the resource limit facility must be stopped on all members of the DB2 data sharing group or all members must switch to an RLST that is not associated with the object.

If the function is the DDF, the facility must be stopped before the object can be dropped. In a DB2 data sharing environment, the facility must be stopped on all members of the DB2 data sharing group.

SQLSTATE: 57005

-667 THE CLUSTERING INDEX FOR A PARTITIONED TABLE SPACE CANNOT BE EXPLICITLY DROPPED

Explanation: The DROP INDEX statement attempted to drop the cluster index for a table residing in a partitioned table space. The cluster index for such a table cannot be dropped explicitly with the DROP INDEX statement.

System Action: The statement cannot be executed. The specified index was not dropped.

Programmer Response: The cluster index for a table in a partitioned table space can only be dropped implicitly when the associated partitioned table space is dropped.

SQLSTATE: 42917

-668 THE COLUMN CANNOT BE ADDED TO THE TABLE BECAUSE THE TABLE HAS AN EDIT PROCEDURE

Explanation: The ALTER TABLE statement attempted to add a column to a table that has an edit procedure. If a table has an edit procedure, no columns can be added to it.

System Action: The statement cannot be executed. The specified table was not altered.

Programmer Response: Verify that the correct table was specified in the ALTER statement. Do not attempt to ALTER the definition of a table that has an installation-written edit procedure associated with it.

SQLSTATE: 56018

-669 THE OBJECT CANNOT BE EXPLICITLY DROPPED. REASON *reason-code*

Explanation: The DROP statement failed for the reason indicated by the *reason-code* as follows:

0001 The DROP TABLE statement attempted to drop a table that resides in a partitioned table space.

0002 The DROP INDEX statement attempted to drop an index required to enforce the primary key, unique key, or referential constraint of the table.

System Action: The statement cannot be processed. The object is not dropped.

Programmer Response: If the statement is a DROP TABLE statement, the table of a partitioned table space can only be dropped implicitly when the table space itself is dropped.

If the statement is a DROP INDEX statement and you do not want to keep the primary key, unique key, or referential constraint, use the DROP CONSTRAINT clause of the ALTER TABLE statement to remove the constraint, then drop the index.

SQLSTATE: 42917

-670 THE RECORD LENGTH OF THE TABLE EXCEEDS THE PAGE SIZE LIMIT

Explanation: The row length for a table cannot exceed the page size of the table space in which that table resides (or is to reside). The page size of the table space is determined by the buffer pool used by that table space.

One of following conditions may occur:

- As defined in a CREATE TABLE statement, the row length for the table would exceed the page size of the specified (or default) table space.
- In the case of an ALTER TABLE statement, addition of the specified column would cause the row length of the table to exceed the page size of the table space.
- In the case of an ALTER TABLE statement used to alter the length of an existing variable length column, the new length of the altered column would cause the row length of the table to exceed the page size of the table space.
- The row length in the result of a join exceeds the page size of a work file table space.
- The row length of a large sort record exceeds the page size of a work file table space. The sort record includes columns that are being sorted and columns that the user selects. The length of the columns that are being sorted is the sort key length. The length of the columns that the user selects is the sort data length.

System Action: The statement cannot be executed. The object table was not created or altered.

Programmer Response: In the case of CREATE TABLE, either (1) the row length of the table must be reduced (by eliminating or reducing the lengths of one or more of the columns), or (2) the table must be assigned to a table space that uses a larger buffer pool (assuming that the row length of the table does not exceed that page size limit).

In the case of ALTER TABLE, either (1) the length of the column to be added to the table must be reduced or, (2) if the row length of the table is already at the maximum, the table cannot be altered to add any additional columns.

In the case of a row length that exceeds the page size of a work file table space, eliminate columns from the result of the join.

In the case of a large sort record in which the row length exceeds the page size of a work file table space, eliminate columns from the SELECT list or reduce the number of columns that are being sorted.

SQLSTATE: 54010

-671 THE BUFFERPOOL ATTRIBUTE OF THE TABLE SPACE CANNOT BE ALTERED AS SPECIFIED BECAUSE IT WOULD CHANGE THE PAGE SIZE OF THE TABLE SPACE

Explanation: For example, the change to the buffer pool attribute for the table space specified in the ALTER TABLESPACE statement would change the page size of the table space—either from 4KB to 32KB, 8KB to 16KB, or vice versa.

Use of the ALTER TABLESPACE statement to change the page size of a table space is not permitted.

System Action: The statement cannot be executed. The table space definition was not altered.

Programmer Response: For example, if the table space uses one of the 4KB buffer pools (for example, BP0, BP1, or BP2), it can be reassigned to one of the other 4KB buffer pools (but not buffer pool BP32K). If, however, it is assigned to buffer pool BP32K, the buffer pool assignment cannot be subsequently altered.

SQLSTATE: 53040

-672 OPERATION DROP NOT ALLOWED ON TABLE *table_name*

Explanation: The DROP operation failed for one of the following reasons:

- The table being dropped has the RESTRICT ON DROP attribute.
- The table space or database being dropped contains the specified table, which has the RESTRICT ON DROP attribute.

System Action: The DROP statement cannot be executed.

Programmer Response: Before dropping the table, alter the table, specifying DROP RESTRICT ON DROP.

For DROP TABLESPACE or DROP DATABASE, make sure that there are no other tables within the table space or database with the RESTRICT ON DROP attribute. The following SELECT statement can identify the tables:

```
   SELECT CREATOR, NAME
FROM SYSIBM.SYSTABLES
WHERE TYPE = 'T'
AND CLUSTERTYPE = 'Y'
AND DBNAME = 'database_name'
AND TSNAME = 'tablespace_name';
```

SQLSTATE: 55035

-676 **ONLY A 4K PAGE BUFFERPOOL CAN BE USED FOR AN INDEX**

Explanation: A buffer pool having a page size other than 4KB was specified in the CREATE INDEX statement. Only 4KB buffer pools (that is, BP0, BP1, and BP2) can be specified for indexes.

System Action: The statement cannot be executed. The specified index was not created.

Programmer Response: Correct the statement to specify a 4KB buffer pool. Refer to Chapter 6 of *DB2 SQL Reference* for information about the syntax of SQL statements.

SQLSTATE: 53041

-677 **INSUFFICIENT VIRTUAL STORAGE FOR BUFFERPOOL EXPANSION**

Explanation: An attempt to either open (create) or expand a buffer pool has failed because insufficient virtual storage was available.

This error may occur under either of two circumstances:
- An attempt to create a buffer pool while opening a table space or index(space), or
- An attempt to expand a buffer pool from its current size to its maximum size.

System Action: The statement cannot be executed.

Programmer Response: If this error should occur during interactive execution of an SQL statement or execution of an application program, installation administration should be notified.

Installation Action: It may be necessary to reexamine the buffer pool storage strategy.

One of the following messages has also been sent to the MVS console: DSNB601I, DSNB603I, or DSNB605I. Refer to “Part 3. Section 3. DB2 Messages” on page 135 for explanations of these messages.

SQLSTATE: 57011

-678 **THE LITERAL literal SPECIFIED FOR THE INDEX LIMIT KEY MUST CONFORM TO THE DATA TYPE** *data-type* **OF THE CORRESPONDING COLUMN** *column-name*

Explanation: The index limit key value 'literal' has been specified incorrectly in the CREATE INDEX or ALTER INDEX statement.

Limit key value specifications must conform to the data type of the corresponding index key column. In this case, the 'literal' must be of data type 'data-type' to conform to the data type of column 'column-name'.

System Action: The statement cannot be executed. The specified index was not created.

Programmer Response: Correct the statement so that each limit key value literal is of precisely the same data type as that of the corresponding index key column.

SQLSTATE: 53045

-679 **THE OBJECT** *name* **CANNOT BE CREATED BECAUSE A DROP IS PENDING ON THE OBJECT**

Explanation: The application program has executed a DROP for the specified object, and then tried to re-create an object with the same name (and of the same type) before the DROP was completed.

System Action: The statement cannot be executed. The specified object was not created.

Programmer Response: The logic of the application program must be modified to issue a COMMIT (or the IMS or CICS equivalent) between the DROP and CREATE statements.

SQLSTATE: 57006

-680 **TOO MANY COLUMNS SPECIFIED FOR A TABLE, VIEW OR TABLE FUNCTION**

Explanation: The maximum number of columns permitted per table, view, or table function is 750. The statement attempted to perform one of the following actions:

- CREATE or ALTER a table to contain more than 750 columns
- CREATE a view with more than 750 columns
- CREATE a table function with more than 750 columns in the RETURNS TABLE clause

System Action: The statement cannot be executed.

Programmer Response: Change the CREATE statement to not include more than 750 columns, or do not try to alter an existing table to contain more than 750 columns.

SQLSTATE: 54011

-681 **COLUMN** *column-name* **IN VIOLATION OF INSTALLATION DEFINED FIELD PROCEDURE. RT:** *return-code*, **RS:** *reason-code*, **MSG:** *message-token*

Explanation: An installation field procedure returned an error for 'column-name'. The 'reason-code' and 'message-token' are defined by the field procedure. They may give additional information to help determine the cause of the problem.

Return code

	Error
4	Invalid value on encode or decode or invalid column description on define
8	Invalid parameter value
12	Field procedure error on any function

Use 'reason-code' and 'message-token' for additional information.

System Action: The statement cannot be executed.

Programmer Response: If it is not a field procedure error, determine the requirements imposed by the field procedure. If it is a field procedure error, examine the field procedure.

SQLSTATE: 23507

-682 **FIELD PROCEDURE** *procedure-name* **COULD NOT BE LOADED**

Explanation: The field procedure 'procedure-name' cannot be loaded.

System Action: The statement cannot be executed.

Programmer Response: The application should either commit or roll back to previous COMMIT. Then, in general, the application should terminate.

SQLSTATE: 57010

-683 **THE SPECIFICATION FOR COLUMN, DISTINCT TYPE, FUNCTION, OR PROCEDURE** *data-item* **CONTAINS INCOMPATIBLE CLAUSES**

Explanation: There is an error in the data item specification in an SQL statement. Incompatible specifications are present such as ″INTEGER and FOR BIT DATA″, or ″INTEGER AS LOCATOR″. The location of the error is given by *data-item* as follows:

- For a CREATE or ALTER TABLE statement, *data-item* gives the name of the column containing the error. The error could be an invalid specification of FOR BIT DATA, FOR SBCS DATA, FOR MIXED DATA, or FIELDPROC for column *data-item*.
- For a CREATE FUNCTION or CREATE PROCEDURE statement, *data-item* is a token that identifies the area of the problem in the statement. For example, ″PARAMETER 3″ or ″RETURNS″ or ″CAST FROM″.
- For a CREATE FUNCTION or CREATE PROCEDURE statement, or some other statement that includes a function parameter list, *data-item* is a token that identifies the area of the problem in the statement.
- For a CREATE DISTINCT TYPE statement, *data-item* gives the name of the type being defined.
- Otherwise, data-item is a token that identifies the area of the problem in a parameter list for a function. For example, ″PARAMETER 5″.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement by removing the incompatible specification.

SQLSTATE: 42842

-684 **THE LENGTH OF LITERAL LIST BEGINNING** *string* **IS TOO LONG**

Explanation: The length of the literal list beginning with 'string', excluding insignificant blanks and delimiting parentheses is greater than 255.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement.

SQLSTATE: 54012

-685 **INVALID FIELD TYPE,** *column-name*

Explanation: The field description returned by the field procedure is invalid. The data type code denotes a long string or has an invalid value.

System Action: The statement cannot be executed.

Programmer Response: Correct the field procedure so that it returns a valid data type code.

SQLSTATE: 58002

-686 **COLUMN DEFINED WITH A FIELD PROCEDURE CAN NOT COMPARE WITH ANOTHER COLUMN WITH DIFFERENT FIELD PROCEDURE**

Explanation: The columns specified are not compatible. Different field procedures are specified, or only one field procedure is specified.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement. Refer to Chapter 3 of *DB2 SQL Reference* for comparison restrictions between columns defined with a field procedure.

SQLSTATE: 53043

-687 FIELD TYPES INCOMPARABLE

Explanation: One column cannot be compared with another column that has incompatible field types.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement. Refer to Chapter 3 of *DB2 SQL Reference* for comparison restrictions between columns defined with a field procedure.

SQLSTATE: 53044

-688 INCORRECT DATA RETURNED FROM FIELD PROCEDURE, *column-name*, *msgno*

Explanation: Unexpected data returned from field procedure for column 'column-name'. For more information see 'msgno'.

System Action: The statement cannot be executed.

Programmer Response: Correct the field procedure so that it returns values that are consistent with their descriptions.

SQLSTATE: 58002

-689 TOO MANY COLUMNS DEFINED FOR A DEPENDENT TABLE

Explanation: The maximum number of columns allowed for a dependent table is 749. The code is used to report that the statement is invalid because of one of the following:

- A CREATE TABLE statement is creating a dependent table with 750 columns.
- An ALTER TABLE statement is altering a dependent table with 749 columns to add a column, or altering a table with 750 columns to become a dependent table.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement to conform to the column limit for a dependent table.

SQLSTATE: 54011

-690 THE STATEMENT IS REJECTED BY DATA DEFINITION CONTROL SUPPORT. REASON *reason-code*

Explanation: The code is issued by the data definition control support to report that the statement is rejected for the reason indicated by 'reason-code' after consulting the application registration table and object registration table.

The explanation of the given reason code:

- 0001

  Data definition control support is running under the controlling by application name mode. The statement is rejected because the current application is not registered in application registration table with DEFAULTAPPL on.
- 0002

  Data definition control support is running under the controlling by application name with exceptions mode. The statement is rejected because the object is not registered in object registration table and the current application is not registered in application registration table with DEFAULTAPPL on.
- 0003

  Data definition control support is running under the controlling by application name with exceptions mode. The statement is rejected because the object is registered in object registration table but the current application does not match.
- 0004

  Data definition control support is running under the controlling by object name with exceptions mode. The statement is rejected because the object is registered in object registration table but the current application does not match.
- 0005

  Data definition control support is running under the controlling by object name mode. The statement is rejected because the object is registered in object registration table but the current application does not match.
- 0006

  Data definition control support is running under the controlling by object name mode. The statement is rejected because the object is not registered in object registration table.

System Action: The statement cannot be executed.

Programmer Response: None if valid rejection. Otherwise, check to see if data definition control support is running under the desired mode. Check one or both registration table(s) to determine if the entries of the registration table(s) are correct. If they are not, then update the registration table(s).

SQLSTATE: 23508

-691 THE REQUIRED REGISTRATION TABLE *table-name* **DOES NOT EXIST**

Explanation: The data definition control support assumes the existence of the application registration table and object registration table. But either one or both tables is not defined.

System Action: The statement cannot be executed.

Programmer Response: Determine whether the required registration tables do exist. If not, create the required tables.

SQLSTATE: 57018

-692 **THE REQUIRED UNIQUE INDEX *index-name* FOR DDL REGISTRATION TABLE *table-name* DOES NOT EXIST**

Explanation: A unique index must be defined for each registration table. The code is issued when either the required index does not exist or the index defined is not a unique index.

System Action: The statement cannot be executed.

Programmer Response: Determine whether the required unique index does exist. If not, create the required index. If the index does exist, but is not unique, drop it and recreate it as a unique index.

SQLSTATE: 57018

-693 **THE COLUMN *column-name* IN DDL REGISTRATION TABLE OR INDEX *table-name (index-name)* IS NOT DEFINED PROPERLY**

Explanation: An error occurred during verification of the application or object registration table.

The table is improperly defined for the following reasons:
- A required column is missing.
- A column description is invalid because its name, column number, data type, length, or null attribute is incorrect.

or

The index is improperly defined for the following reasons:
- A required key column is missing.
- A key column description is invalid because of its key sequence or because its ordering is incorrect.
- A defined key column should not be part of the unique key.

System Action: The statement cannot be executed.

Programmer Response: Correct or alter the definition of the required registration table or index.

SQLSTATE: 55003

-694 **THE DDL STATEMENT CANNOT BE EXECUTED BECAUSE A DROP IS PENDING ON THE DDL REGISTRATION TABLE *table-name***

Explanation: An error occurred while accessing the application registration table or object registration table. The application registration table or object registration table was dropped, but the DROP statement was not committed.

System Action: The statement cannot be executed.

Programmer Response: Resubmit the job. If the same error happens, check for the application that issued the DROP statement for the application registration table or the object registration table. Either commit the DROP statement which dropped the table and create a new application registration table or object registration table, or issue a ROLLBACK for the DROP statement to put the tables back.

SQLSTATE: 57023

-696 **THE DEFINITION OF TRIGGER *trigger-name* INCLUDES AN INVALID USE OF CORRELATION NAME OR TRANSITION TABLE NAME *name*. REASON CODE=*reason-code***

Explanation: The trigger definition included an invalid use of *name*

trigger-name
The trigger that encountered the error

name The transition variable correlation name or transition table name

reason-code
A reason-code indicating the specific problem as follows:
1. NEW correlation name and NEW_TABLE name are not allowed in a DELETE trigger.
2. OLD correlation name and OLD_TABLE name are not allowed in an INSERT trigger.
3. OLD_TABLE name and NEW_TABLE name are not allowed in a BEFORE trigger.

System Action: The statement cannot be executed. The trigger was not created.

Programmer Response: Remove the invalid correlation name or transition table name along with the preceding keyword.

SQLSTATE: 42898

-697 **OLD OR NEW CORRELATION NAMES ARE NOT ALLOWED IN A TRIGGER DEFINED WITH THE FOR EACH STATEMENT CLAUSE. OLD_TABLE OR NEW_TABLE NAMES ARE NOT ALLOWED IN A TRIGGER WITH THE BEFORE CLAUSE.**

Explanation: The trigger, as defined, includes a REFERENCING clause with one of the following invalid combinations:
- OLD or NEW correlation names specified (or both) and the FOR EACH STATEMENT clause.
- NEW_TABLE or OLD_TABLE correlation names specified (or both) and the BEFORE clause.

System Action: The statement cannot be executed. The trigger was not defined.

Programmer Response: Remove invalid correlation

names or change the trigger granularity to FOR EACH ROW.

SQLSTATE: 42899

-713 **THE REPLACEMENT VALUE** *value* **FOR** *special-register* **IS INVALID**

Explanation: The *value* specified in the SET special-register statement is not a valid value of the indicated special register.

System Action: The statement cannot be executed.

Programmer Response: Correct the replacement value. See *DB2 SQL Reference* for an explanation of the valid values of each special register.

SQLSTATE: 42815

-715 **PROGRAM** *program-name* **WITH MARK** *release-dependency-mark* **FAILED BECAUSE IT DEPENDS ON FUNCTIONS OF THE RELEASE FROM WHICH FALLBACK HAS OCCURRED**

Explanation: Program 'program-name' depends on a function of DB2 that is not supported by the current active release.

program-name
Name of the application program.

release-dependency-mark
A 1-character mark showing the oldest DB2 release supporting this program.

System Action: The BIND operation for this plan or package is not performed.

User Response: The program cannot be used until the DB2 subsystem is remigrated to the newer release.

Operator Response: Notify the system programmer.

System Programmer Response: Warn users not to use plans or packages containing this program until the DB2 subsystem has been remigrated to the newer release.

SQLSTATE: 56064

-716 **PROGRAM** *program-name* **PRECOMPILED WITH INCORRECT LEVEL FOR THIS RELEASE**

Explanation: Program 'program-name' was precompiled under a release not supported by the current level of DB2, or the contents of the DBRM have been modified after the precompilation phase.

User Response: Precompile the named program again using the current precompiler. Reissue the BIND subcommand.

Problem Determination: If the program was precompiled at an appropriate release, and the problem persists, collect the following:

- A hexadecimal print of the first record of the failing DBRM
- The listing from the precompile job that generated the DBRM
- Output from the BIND attempt.

SQLSTATE: 56065

-717 *bind-type* **FOR** *object-type object-name* **WITH MARK** *release-dependency-mark* **FAILED BECAUSE** *object-type* **DEPENDS ON FUNCTIONS OF THE RELEASE FROM WHICH FALLBACK HAS OCCURRED**

Explanation: The plan or package indicated depends on a function of DB2 which is not supported by the currently active release.

bind-type
REBIND

object-type
PLAN | PACKAGE

object-name
Name of the application plan, or the package

release-dependency-mark
A one-character mark showing the oldest release of DB2 can support this plan or package. The release dependency mark for the plan is kept in the IBMREQD columns in the DB2 catalog in SYSIBM.SYSPLAN (plans), SYSIBM.SYSPACKAGE(packages).

System Action: The REBIND operation for this plan or package is not performed.

User Response: The plan or package cannot be used until the DB2 subsystem is remigrated to the newer release.

Operator Response: Notify the system programmer.

System Programmer Response: Warn users not to use the plan or package until the DB2 subsystem has been remigrated to the newer release.

SQLSTATE: 56066

-718 **REBIND OF PACKAGE** *package-name* **FAILED BECAUSE IBMREQD OF** *ibmreqd* **IS INVALID**

Explanation: The IBMREQD column of the SYSIBM.SYSPACKAGE catalog table for the named package contains an unrecognizable character.

package-name
Name of the package (location.collection.package.version)

System Action: The REBIND failed.

User Response: You must do a BIND

ACTION(REPLACE) for this package.

SQLSTATE: 56067

-719 **BIND ADD ERROR USING** *auth-id* **AUTHORITY PACKAGE** *package-name* **ALREADY EXISTS**

Explanation: An attempt is made to add a package that already exists. The combination of 'location.collection.package.version' must be unique in the SYSIBM.SYSPACKAGE table. In addition, the combination of 'location.collection.package.consistency-token' must be unique.

auth-id Authorization ID of the invoker of the BIND subcommand.

package-name
Name of the package (location.collection.package.version).

System Action: No package is created.

System Programmer Response: Check the SYSIBM.SYSPACKAGE catalog table for names of existing application packages. Re-invoke the BIND subcommand with a 'location.collection.package.version' that is not in use.

SQLSTATE: 42710

-720 **BIND ERROR, ATTEMPTING TO REPLACE PACKAGE =** *package_name* **WITH VERSION =** *version2* **BUT THIS VERSION ALREADY EXISTS**

Explanation: An attempt is made to create a version of a package that already exists. The version specified in the REPLVER keyword is different from the version specified for the precompile. The version specified for the precompile already exists in the catalog. The combination of 'location.collection.package.version' must be unique in the SYSIBM.SYSPACKAGE catalog table. A common mistake is that the user may believe that the version he is creating is the one specified in the REPLVER keyword. This is not the case. The version specified in the REPLVER keyword is the name of the version being replaced. The version that will be created is the version that was given to the program when it was precompiled.

package_name
Fully qualified package name

version2
Version-id of package that is to be created

System Action: The bind will fail.

System Programmer Response: There are two approaches to solve this problem. The first is to precompile the program again with a new version name and reissue the original BIND subcommand. The other approach is not to do the precompile but reissue the BIND subcommand with REPLVER(SAME).

SQLSTATE: 42710

-721 **BIND ERROR FOR PACKAGE =** *pkg-id* **CONTOKEN =** *contoken'X IS NOT UNIQUE SO IT CANNOT BE CREATED*

Explanation: An attempt is made to add or replace a package with a consistency token that is not unique for that package. In other words, the combination of *location.collection.package.consistency-token* already exists.

pkg-id Fully qualified name of the package.

contoken
Consistency token in hexadecimal.

System Action: The BIND will fail.

System Programmer Response: Check the SYSIBM.SYSPACKAGE catalog table for names of existing application packages with the indicated consistency token. Reissue the BIND subcommand such that the *location.collection.package.consistency-token* is unique within the catalog. The following SQL statement can be used to query the catalog:

```
SELECT COLLID,NAME
FROM loc-id.SYSIBM.SYSPACKAGE
WHERE HEX(CONTOKEN) = contoken
```

SQLSTATE: 42710

-722 *bind-type* **ERROR USING** *auth-id* **AUTHORITY PACKAGE** *package-name* **DOES NOT EXIST**

Explanation: The indicated subcommand was issued against a package that does not exist. The individual variable fields contain:

bind-type
Type of bind subcommand (BIND | REBIND | FREE).

auth-id Authorization ID of the invoker of the BIND subcommand.

package-name
Name of the package (location.collection.package.version)

System Action: Package not rebound or freed.

System Programmer Response: Check the SYSPACKAGE catalog table for the correct 'location.collection.package.version' to use.

SQLSTATE: 42704

-723 **AN ERROR OCCURRED IN A TRIGGERED SQL STATEMENT IN TRIGGER** *trigger-name***, SECTION NUMBER** *section-number***. INFORMATION RETURNED: SQLCODE** *sqlerror***, SQLSTATE** *sqlstate***, AND MESSAGE TOKENS** *token-list*

Explanation: During execution of an UPDATE, INSERT, or DELETE statement, a trigger was activated. One of the triggered SQL statements received an SQL error condition.

trigger-name
The trigger that was activated when the error occurred.

section-number
The section number associated with the failing triggered SQL statement. For triggers that contain a WHEN clause, the WHEN clause is section number one. The triggered SQL statements are numbered sequentially, beginning with section number two. This is true for triggers with or without a WHEN clause.

sqlcode
The SQLCODE received by the activated trigger.

sqlstate
The corresponding SQLSTATE for the SQLCODE received by the activated trigger.

token-list
The list of tokens from the original SQL error. This list might be truncated.

System Action: The trigger and the original INSERT, UPDATE, or DELETE statement cannot be processed. The triggering table is unchanged.

Programmer Response: Contact your Database Administrator to determine why the trigger named in the message received the error.

System Programmer Response: Use the trigger name and section number to determine the failing SQL statement. If the trigger definition is available, use the section number to determine the failing statement. Alternatively, the failing statement can be retrieved from the SYSIBM.SYSPACKSTMT catalog table: SELECT TEXT, SEQNO FROM SYSIBM.SYSPACKSTMT WHERE COLLID = 'schema-name' AND NAME = 'trigger-name' AND SECTNO = section-number ORDER BY SEQNO Refer to the explanation of the reported SQLCODE. Follow the action suggested by that SQLCODE.

SQLSTATE: 09000

-724 **THE ACTIVATION OF THE** *object-type* **OBJECT** *object-name* **WOULD EXCEED THE MAXIMUM LEVEL OF INDIRECT SQL CASCADING**

Explanation: Cascading of indirect SQL occurs when a trigger, user-defined function or stored procedure invokes another trigger, user-defined function or stored procedure which in turn invokes another. The activation of some of the triggers in this chain might be due to the enforcement of referential constraint delete rules. The depth of this cascading is limited to 16.

Note that recursive situations where a trigger includes a triggered SQL statement that directly or indirectly causes the same trigger to be activated are very likely to cause this error. The trigger should contain logic to check for a terminating condition to prevent this error.

object-type
Names the type of object being called. Object type is TRIGGER, FUNCTION, or PROCEDURE.

object-name
Specifies the name of the trigger, user-defined function or stored procedure that would have be activated at the seventeenth level of cascading.

System Action: The original statement could not be executed. All SQL statements executed by all triggers, user-defined functions, and stored procedures in the cascade chain are rolled back. External actions performed by the indirect SQL, such as sending a network message might have already occurred.

Programmer Response: Start with the indirect SQL that is activated by the original SQL operation. Check for recursive patterns in any invoked user-defined functions or in any triggers defined on the subject of an update operation. If the chain is not recursive, the cascade chain must be simplified by altering the triggers, user-defined functions, or stored procedures involved.

SQLSTATE: 54038

-725 **THE SPECIAL REGISTER** *register* **AT LOCATION** *location* **WAS SUPPLIED AN INVALID VALUE**

Explanation: DB2 received a SET statement with an invalid value. Valid SETs might be allowed or retained. Further processing stops at the named site until the SET statement is corrected.

System Action: All SQL statements at the named remote site are rejected until the SET statement that was in error is corrected.

Programmer Response: The SET statement should be reissued with a valid value. This situation can be corrected with a local SET statement or with a DRDA SET statement executed at the remote site. Once the

special register has been supplied a valid value, the application can resume execution.

SQLSTATE: 42721

-726 **BIND ERROR ATTEMPTING TO REPLACE PACKAGE = *package-name*. THERE ARE ENABLE OR DISABLE ENTRIES CURRENTLY ASSOCIATED WITH THE PACKAGE**

Explanation: The BIND subcommand was issued to replace a package that has ENABLE or DISABLE entries currently associated with the package.

System Action: Package not bound.

System Programmer Response: FREE the package first and then BIND the package.

SQLSTATE: 55030

-728 **DATA TYPE *data-type* IS NOT ALLOWED IN DB2 PRIVATE PROTOCOL PROCESSING**

Explanation: An SQL statement uses a data type that cannot be used with DB2 private protocol. Data types such as LOBs (large objects), row IDs and user defined types cannot be accessed using DB2 private protocol.

System Action: The statement cannot be executed.

Programmer Response: If this is an invalid data type, correct the statement. If this is a valid (but disallowed) data type, you can:

- Remove access to the data type listed, and re-execute the statement.
- Change the application program so that DRDA is used to access the data.
- BIND the package containing the statement to the remote site with bind option DBPROTOCOL(DRDA) and rerun the program.

Refer to the *DB2 SQL Reference* for more information.

SQLSTATE: 56080

-729 **A STORED PROCEDURE SPECIFYING COMMIT ON RETURN CANNOT BE THE TARGET OF A NESTED CALL STATEMENT**

Explanation: A stored procedure defined with the COMMIT ON RETURN attribute was called from a stored procedure, user-defined function, or trigger. Stored procedures defined with COMMIT ON RETURN cannot be nested in this way.

System Action: The SQL statement is not executed. If the CALL statement references a remote server, the unit of work is placed in a must rollback state.

Programmer Response: Remove the CALL to the stored procedure that was defined with the COMMIT ON RETURN attribute.

SQLSTATE: 429B1

-730 **THE PARENT OF A TABLE IN A READ-ONLY SHARED DATABASE MUST ALSO BE A TABLE IN A READ-ONLY SHARED DATABASE**

Explanation: An attempt was made to define a relationship between a table in a read-only shared database and a table that is not.

System Action: The statement cannot be executed.

Programmer Response: Insure that the correct tables are being used for the relationship being defined.

SQLSTATE: 56053

-731 **USER-DEFINED DATASET *dsname* MUST BE DEFINED WITH SHAREOPTIONS(1,3)**

Explanation: The VSAM SHAREOPTIONS must be (1,3) for all of the indexes and table spaces in the database. The user-defined data set identified did not meet this requirement.

System Action: The statement cannot be executed.

Programmer Response: Insure that the data sets used in the shared database are defined with the proper SHAREOPTIONS.

SQLSTATE: 56054

-732 **THE DATABASE IS DEFINED ON THIS SUBSYSTEM WITH THE ROSHARE READ ATTRIBUTE BUT THE TABLE SPACE OR INDEX SPACE HAS NOT BEEN DEFINED ON THE OWNING SUBSYSTEM**

Explanation: Prior to creating a table space or index in a database with the ROSHARE READ attribute, that object must first be defined on the owning subsystem.

System Action: The statement cannot be executed.

Programmer Response: Verify that the table space or index has been defined on the owning system in a ROSHARE OWNER database.

SQLSTATE: 56055

-733 **THE DESCRIPTION OF A TABLE SPACE, INDEX SPACE, OR TABLE IN A ROSHARE READ DATABASE MUST BE CONSISTENT WITH ITS DESCRIPTION IN THE OWNER SYSTEM**

Explanation: This code is issued while creating a table space, index, or table in the ROSHARE READ

database. These objects must be consistent with their descriptions in the ROSHARE OWNER database as follows:

- For a table space, the following attributes must be the same:
  - Page size
  - Segment size
  - Number of partitions
- For an index, the following attributes must be the same:
  - Number of partitions
  - Number of subpages
  - Table OBID
  - Total key length
  - Index type (Type 1 or Type 2 Index)
  - Number of key columns
- For a table, the following attributes must be the same:
  - Table OBID
  - Maximum record length
  - Number of columns
  - Whether an edit procedure exists

System Action: The statement cannot be executed.

Programmer Response: Ensure that the definition of the table space, index, or table is consistent with that in the ROSHARE OWNER database.

SQLSTATE: 56056

-734 THE ROSHARE ATTRIBUTE OF A DATABASE CANNOT BE ALTERED FROM ROSHARE READ

Explanation: An attempt was made to ALTER a database from ROSHARE READ to either ROSHARE OWNER or ROSHARE NONE.

System Action: The statement cannot be executed.

Programmer Response: Verify that the correct database was specified on the ALTER DATABASE statement. The ROSHARE attribute of a read-only shared database cannot be altered. To change this, DROP and recreate the database.

SQLSTATE: 56057

-735 DATABASE dbid CANNOT BE ACCESSED BECAUSE IT IS NO LONGER A SHARED DATABASE

Explanation: An attempt was made to access an object in the database identified by *dbid*, that is known to the system as having the ROSHARE READ attribute. The database, however, is no longer defined as ROSHARE OWNER on the owning subsystem.

System Action: The statement cannot be processed.

Programmer Response: Verify that the correct object is specified on the statement.

SQLSTATE: 55004

-736 INVALID OBID *obid* SPECIFIED

Explanation: An invalid OBID value was given on the CREATE statement. The OBID is invalid for one of the following reasons:

- The specified OBID does not fall within the acceptable range for OBIDs, which is 1 to 32767.
- The specified OBID is already in use for the given database.

System Action: The statement cannot be executed.

Programmer Response: Verify that the given OBID is a valid value for an OBID. If so, ensure that the OBID is correct for the object to be created, then query the catalog to find the object that is already defined as having the same OBID in the database. If an invalid OBID was given for the object to be created, correct the statement and execute it again. If the existing object is in error, then DROP and CREATE that object using the correct OBID value.

It is possible that the OBID “in use” is the OBID for an object that had been previously dropped. If that is the case, and the CREATE was issued for a table in a non-ROSHARE READ database, then select a different OBID for use in the OBID clause. If the object had been previously dropped and the CREATE was issued for a table in a ROSHARE READ database, COMMIT and re-submit the CREATE TABLE request.

SQLSTATE: 53014

-737 IMPLICIT TABLE SPACE NOT ALLOWED

Explanation: A CREATE TABLE statement was issued using an implicit table space. An implicit table space may not be used in a database that has been defined as a read-only shared database.

System Action: The statement cannot be executed.

Programmer Response: CREATE a table space for the table, using the same name as is given on the owning system. Then resubmit the CREATE TABLE statement.

SQLSTATE: 56056

-739 CREATE OR ALTER FUNCTION *function-name* FAILED BECAUSE FUNCTIONS CANNOT MODIFY DATA WHEN THEY ARE PROCESSED IN PARALLEL.

Explanation: The function cannot be created or altered because ALLOW PARALLEL and MODIFIES SQL DATA were both specified explicitly or implicitly. A function cannot be parallelized if it modifies data.

System Action: The statement cannot be processed.

Programmer Response: Specify DISALLOW PARALLEL or change the MODIFIES SQL DATA to NO SQL, CONTAINS SQL or READS SQL DATA.

SQLSTATE: 56088

-740 **FUNCTION *name* IS DEFINED WITH THE OPTION MODIFIES SQL DATA WHICH IS NOT VALID IN THE CONTEXT IN WHICH IT WAS INVOKED**

Explanation: A user-defined function defined with MODIFIES SQL DATA is only allowed in:

- VALUES clause of an INSERT statement
- SET clause of an UPDATE statement
- VALUES statement in a trigger
- SET Assignment statement
- CALL procedure statement

System Action: The SQL statement failed.

Programmer Response: Remove the user-defined function from the failing statement. or remove the MODIFIES SQL DATA option from the definition of the function.

SQLSTATE: 51034

-741 **A *database-type* DATABASE IS ALREADY DEFINED FOR MEMBER *member-name***

Explanation: A CREATE DATABASE statement was issued for a WORK FILE or TEMP database, but the database can not be created because one is already defined for the named DB2 subsystem or data sharing group member.

database-type
WORK FILE or TEMP

member-name
Name of the DB2 subsystem or data sharing group member that already has a *database-type* database.

System Action: The statement cannot be executed.

Programmer Response: Verify the identity and validity of the existing *database-type* database for the named DB2 subsystem or data sharing member. The existing database can be altered or dropped if necessary. If the existing database is dropped, resubmit the CREATE DATABASE statement.

SQLSTATE: 55020

-742 **DSNDB07 IS THE IMPLICIT WORK FILE DATABASE**

Explanation: The WORKFILE clause cannot be used on a CREATE DATABASE statement to create a work file database for a DB2 subsystem that is not a member of a DB2 data sharing group. The system database, DSNDB07, is the implicit work file database.

System Action: The statement cannot be executed.

Programmer Response: To create the work file database for a DB2 subsystem that is not a member of a DB2 data sharing group, create database DSNDB07 without the WORKFILE clause.

SQLSTATE: 53004

-746 **THE SQL STATEMENT IN AN EXTERNAL FUNCTION, TRIGGER, OR IN STORED PROCEDURE *name* VIOLATES THE NESTING SQL RESTRICTION**

Explanation: If a table is being modified (by INSERT, DELETE or UPDATE), the table can not be accessed by the lower level nesting SQL statement.

If any table is being accessed by a SELECT statement, no table can be modified (by INSERT, DELETE or UPDATE) in any lower level nesting SQL statement.

System Action: The SELECT, INSERT, DELETE or UPDATE SQL statement failed.

Programmer Response: Remove the failing statement from the named external function, trigger or the stored procedure.

SQLSTATE: 57053

-747 **TABLE *table-name* IS NOT AVAILABLE UNTIL THE AUXILIARY TABLES AND INDEXES FOR ITS EXTERNALLY STORED COLUMNS HAVE BEEN CREATED**

Explanation: An attempt was made to access or reference a table with one or more LOB columns, however either

- an auxiliary table for storing one of the LOB columns has not been created, or
- an index has not been created for an auxiliary table, or
- there is not an auxiliary table for each partition of the table space.

System Action: The statement was not executed.

Programmer Response:
1. Use CREATE TABLESPACE to create a LOB table space.
2. Use CREATE TABLE to create the auxiliary table for storing the column.
3. Use CREATE INDEX to create an index on the auxiliary table.
4. Resubmit the statement that failed.

SQLSTATE: 57054

-748 **AN INDEX ALREADY EXISTS ON AUXILIARY TABLE** *table-name*

Explanation: The CREATE INDEX statement would create a second index on the specified auxiliary table. An auxiliary table can have only one index.

System Action: The statement cannot be executed.

Programmer Response: An index already exists. Another index cannot be created.

SQLSTATE: 54042

-750 **THE SOURCE TABLE** *source-name* **CANNOT BE RENAMED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS OR TRIGGER DEFINITIONS**

Explanation: The *source table* in a RENAME statement cannot be renamed because it is referenced in one or more existing view definitions or it is referenced as the triggering table in one or more existing triggers. The table has check constraints defined.

System Action: The statement cannot be executed.

Programmer Response: Change the source name to the name of an object that can be renamed and reissue the statement. Drop any triggers defined on the table before issuing the RENAME statement. These can be found by querying the system catalog: SELECT * FROM SYSIBM.SYSTRIGGERS WHERE TBNAME = 'source-name'

SQLSTATE: 42986

-751 *object-type object-name* **(SPECIFIC NAME** *specific name***) ATTEMPTED TO EXECUTE AN SQL STATEMENT** *statement* **THAT IS NOT ALLOWED**

Explanation: A stored procedure or user-defined function attempted to execute an SQL statement that is not allowed.

Stored procedure
A stored procedure issued an SQL statement that forced the DB2 thread to roll back the unit of work. The SQL statement that caused the thread to be placed in the MUST_ROLLBACK state is one of the following:
COMMIT
ROLLBACK

All further SQL statements are rejected until the SQL application that issued the SQL CALL statement rolls back the unit of work. When control returns to the SQL application that issued the SQL CALL statement, the SQL application must roll back the unit of work. This can be done by issuing an SQL ROLLBACK statement or the equivalent IMS or CICS operation.

User-defined function
External function *object-name* issued one of the following SQL statements:
- COMMIT
- ROLLBACK

System Action: The statement cannot be executed.

Programmer Response: Remove the unsupported statement from your stored procedure or user-defined function.

SQLSTATE: 38003

-752 **THE CONNECT STATEMENT IS INVALID BECAUSE THE PROCESS IS NOT IN THE CONNECTABLE STATE**

Explanation: The application process attempted to execute a CONNECT statement while in the unconnectable state. See the description of the CONNECT statement in the *DB2 SQL Reference* for an explanation of connection states.

System Action: The statement cannot be executed. The connection state of the application process is unchanged.

Programmer Response: Modify the application program to execute a commit or rollback operation prior to executing the CONNECT statement.

SQLSTATE: 0A001

-763 **INVALID TABLE SPACE NAME** *table-space-name*

Explanation: The named table space is invalid for one of the following reasons:
- It is a LOB table space and therefore cannot reside in a work file database.
- It is a LOB table space and therefore cannot contain a non-auxiliary table.
- It is not a LOB table space and therefore cannot contain an auxiliary table.

System Action: The statement cannot be executed.

Programmer Response: Either
- Create the LOB table space in a non-workfile database.
- Create the table in a non-LOB table space.
- Create the auxiliary table in a LOB table space.

SQLSTATE: 560A1

-764 A LOB TABLE SPACE AND ITS ASSOCIATED BASE TABLE SPACE MUST BE IN THE SAME DATABASE

Explanation: An attempt was made to create an auxiliary table in a LOB table space that is not in the same database as the associated base table space.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement to specify a LOB table space in the same database as the associated base table space.

SQLSTATE: 560A2

-765 TABLE IS NOT COMPATIBLE WITH DATABASE

Explanation: A CREATE TABLE or ALTER TABLE statement defines a LOB column in a table whose database attribute is ROSHARE OWNER or ROSHARE READ. This is not permitted.

System Action: The statement is not executed.

Programmer Response: If CREATE TABLE, either assign the table to an ROSHARE NONE database or create the table without a LOB column. If ALTER TABLE, redefine the column as a non-LOB column or move the table to an ROSHARE NONE database.

SQLSTATE: 560A3

-766 THE OBJECT OF A STATEMENT IS AN AUXILIARY TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED

Explanation: An auxiliary table was named in one of the following statements:

- ALTER TABLE
- CREATE ALIAS
- CREATE FUNCTION
- CREATE SYNONYM
- CREATE VIEW
- DELETE
- DESCRIBE TABLE
- INSERT
- SELECT
- UPDATE

There are no attributes of an auxiliary table that can be altered.

Aliases and synonyms cannot be created on an auxiliary table.

Data in an auxiliary table cannot be accessed by specifying the auxiliary table name in the SELECT, INSERT, DELETE, UPDATE, CREATE PROCEDURE, or CREATE FUNCTION statement. Data in an auxiliary table can only be accessed through operations on the base table columns.

System Action: The statement cannot be executed.

Programmer Response: Correct the statement to specify the corresponding base table instead of the auxiliary table and resubmit the statement.

SQLSTATE: 560A4

-767 MISSING OR INVALID COLUMN SPECIFICATION FOR INDEX *index-name*

Explanation: The CREATE INDEX statement failed for one of the following reasons:

- An index on a non-auxiliary table must specify the columns on which the index is defined.
- An index on an auxiliary table must not have a column specification.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the CREATE INDEX statement.

- To create an index on a non-auxiliary table, specify the columns on which the index is defined.
- To create an index on an auxiliary table, do not specify the name of any column.

SQLSTATE: 42626

-768 AN AUXILIARY TABLE ALREADY EXISTS FOR THE SPECIFIED COLUMN OR PARTITION

Explanation: An attempt was made to create an auxiliary table, but an auxiliary table for the specified column or partition already exists. When the base table belongs to a non-partitioned table space, there can be only one auxiliary table per LOB column of the table. When the base table belongs to a partitioned table space, for any given LOB column, all values of the LOB column for a given partition are stored in their own auxiliary table. There must be one auxiliary table per partition of the base table space.

System Action: The statement is not executed.

Programmer Response: Check that the correct table name, column name, and if applicable, partition number have been specified. If a different name is desired for the existing auxiliary table, the RENAME TABLE statement can be used to rename the auxiliary table.

SQLSTATE: 560A5

-769 **SPECIFICATION OF CREATE AUX TABLE DOES NOT MATCH THE CHARACTERISTICS OF THE BASE TABLE**

Explanation: Either an attempt was made to create an auxiliary table

- using the PART clause and the specified base table is not partitioned or
- without using the PART clause and the specified base table is partitioned

If the base table is not partitioned, then the PART keyword is not allowed on the CREATE AUXILIARY TABLE statement. If the base table is partitioned, then the PART keyword must be specified.

System Action: The auxiliary table was not created.

Programmer Response: Check whether the name of the base table specified in the CREATE AUXILIARY TABLE statement is correct. If it is correct and the table is not partitioned, remove the PART clause from the statement. If it is correct and the table is partitioned, add the PART clause to the statement. If the table name is not correct, correct the name and also check that the correct column name is specified.

SQLSTATE: 53096

-770 **TABLE** *table-name* **CANNOT HAVE A LOB COLUMN UNLESS IT ALSO HAS A ROWID COLUMN**

Explanation: An attempt was made to create a table with a LOB column or to add a LOB column to a table, but the table does not have a ROWID column. A table with a LOB column must also have a ROWID column.

System Action: The statement was not executed.

Programmer Response: If creating a table with a LOB column, define a column with type ROWID in the same table. If using ALTER to add a LOB column to a table, first use ALTER to add a column with type ROWID to the table.

SQLSTATE: 530A6

-771 **INVALID SPECIFICATION OF A ROWID COLUMN**

Explanation: For an ALTER or CREATE TABLE statement, the specification of a ROWID column might be invalid for one of the following reasons:

- A ROWID column cannot be added to a temporary table.
- The *referential-constraint* clause cannot specify a ROWID column as a column of a foreign key.
- A ROWID column cannot be a column of a primary key or unique key.
- A ROWID column cannot be a column in a table with an EDITPROC.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax and resubmit the statement.

SQLSTATE: 428C7

-797 **ATTEMPT to CREATE TRIGGER** *trigger-name* **WITH AN UNSUPPORTED TRIGGERED SQL STATEMENT**

Explanation: The trigger definition includes an unsupported triggered SQL statement. The SQL statements allowed as a triggered SQL statement depend on the type of trigger:

- A BEFORE trigger can include the following triggered SQL statements:
  - a fullselect or VALUES statement
  - a SET transition-variable statement (not allowed in a BEFORE DELETE trigger)
  - a SIGNAL SQLSTATE statement
  - a CALL statement
- An AFTER trigger can include the following triggered SQL statements:
  - a fullselect or VALUES statement
  - an INSERT statement
  - a searched UPDATE statement
  - a searched DELETE statement
  - a SIGNAL SQLSTATE statement
  - a CALL statement

System Action: The CREATE TRIGGER statement cannot be executed, and the trigger is not created.

Programmer Response: Check the triggered SQL statements in the trigger for any statement that is not listed above and remove it.

SQLSTATE: 42987

-798 **YOU CANNOT INSERT A VALUE INTO A COLUMN THAT IS DEFINED WITH THE OPTION GENERATED ALWAYS COLUMN** *column-name*

Explanation: When inserting into a table, a value was specified for column *column-name* with the GENERATE ALWAYS attribute. GENERATED ALWAYS columns should not be specified in the column-list for an insertion unless the corresponding entry in the VALUES list is DEFAULT.

System Action: The INSERT is not performed.

Programmer Response: Remove the column from the column-list or specify DEFAULT for the GENERATED ALWAYS column in the VALUES clause.

You may also use the OVERRIDING clause as a possible solution for this situation. See INSERT in DB2 SQL Reference for more information about the

OVERRIDING USER VALUE clause.

SQLSTATE: 428C9

-802 **EXCEPTION ERROR** *exception-type* **HAS OCCURRED DURING** *operation-type* **OPERATION ON** *data-type* **DATA, POSITION** *position-number*

Explanation: An exception error has occurred in the processing of an SQL arithmetic function or arithmetic expression. This error may be indicated by *exception-type*. Possible values for *exception-type* are FIXED POINT OVERFLOW, DECIMAL OVERFLOW, ZERO DIVIDE, DIVIDE EXCEPTION, EXPONENT OVERFLOW, or OUT OF RANGE. The exception error occurred in one of the following areas:

- In the SELECT list of an SQL SELECT statement.
- In the search condition of a SELECT, UPDATE, or DELETE statement.
- In the SET clause of the UPDATE statement.
- Found during the evaluation of a column function.

Data-type may indicate the data types of the items being manipulated. Possible values for *data-type* are INTEGER, SMALLINT, DECIMAL, and FLOAT. The data type displayed in the message may indicate the data type of the temporary internal copy of the data, which may differ from the actual column or literal data type due to conversions by DB2.

Operation-type may indicate the arithmetic operation that was being performed at the time of the error. The possible operation-types are ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION, NEGATION, BUILT-IN FUNCTION, COLUMN FUNCTION, and JAVA CONVERSION.

Position-number may indicate the position of the expression in a SELECT list if the error was in the SELECT list of an outer SELECT statement

Note: Parts of *exception-type*, *data-type*, *operation-type*, and/or *position-number* may not be returned to the SQL communication area, depending on where the error was detected.

System Action: The statement cannot be executed. In the case of an INSERT or UPDATE statement, no data is updated or deleted. If the statement was a cursor-controlled FETCH, then the CURSOR will remain open unless the exception occurred while processing a column function (indicated by *operation-type* of COLUMN FUNCTION), in which case the CURSOR will be closed. If the CURSOR is closed, subsequent attempts to use that cursor without first doing an OPEN for it receive an SQLCODE -501. If the statement was a cursor-controlled OPEN then the CURSOR will remain closed.

Programmer Response: Examine the SQL statement to see if the cause of the problem can be determined. The problem may be data-dependent, in which case it will be necessary to examine the data that was being processed at the time the error occurred.

If the arithmetic expression in error was within the SELECT list of the outer SELECT statement, then it is advisable to include an indicator variable for all expressions in the SELECT list. This allows processing to continue so that non-error column and expression values can be returned.

See the explanation of SQLCODE -405 for allowed ranges of numeric data types.

Problem Determination: A fixed point overflow can occur during any arithmetic operation on either INTEGER or SMALLINT fields.

A divide exception can occur on a decimal division operation when the quotient exceeds the specified data field size. A zero divide occurs on a division by zero.

An exponent overflow can occur when the result characteristic of any floating-point operation exceeds 127 and the result fraction is not zero, for example, the magnitude of the result exceeds approximately 7.2E+75.

A decimal overflow exception can occur under either of the following circumstances:

- One or more non-zero digits are lost because the destination field in any decimal operation is too short to contain the result.
- A Java stored procedure or user-defined function sets a decimal value in an output parameter that has a precision or scale too small for the value. *operation-type* is JAVA CONVERSION. *data-type* is DECIMAL. *position-number* indicates which parameter of the CALL statement or user-defined function invocation is in error.

Any of the exceptions/overflows can occur during the processing of a Built-In Function. If the *operation-type* is FUNCTION, then the error occurred while processing either an input, intermediate, or final value. The exception may occur because the value of a parameter is out of range.

SQLSTATE: 22012 if ZERO DIVIDE.

22003 if other than ZERO DIVIDE.

-803 **AN INSERTED OR UPDATED VALUE IS INVALID BECAUSE THE INDEX IN INDEX SPACE** *indexspace-name* **CONSTRAINS COLUMNS OF THE TABLE SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THOSE COLUMNS. RID OF EXISTING ROW IS X***rid*

Explanation: The table that is the object of the INSERT or UPDATE operation is constrained (by UNIQUE INDEX in the INDEX SPACE

indexspace-name to have unique values in certain columns. Completion of the requested INSERT or UPDATE would result in duplicate values occurring in row *rid*.

If a view is the object of the INSERT or UPDATE statement, the table that defines the view is constrained. The update might also be caused by a DELETE operation of a parent row that cascades to a dependent row with a delete rule of SET NULL.

System Action: The INSERT, UPDATE, or DELETE statement cannot be executed. The object table is unchanged.

Programmer Response: Examine the definitions for UNIQUE INDEX in the INDEX SPACE *indexspace-name* to determine the uniqueness constraint imposed. Please refer to SYSIBM.SYSINDEXES for the *indexspace-name* and the associated *index-name*.

For an UPDATE statement, verify that the specified operation is consistent with the uniqueness constraint. If this does not indicate the error, examine the object table to determine the cause of the problem.

For an INSERT statement, examine the object table to determine which values violate the uniqueness constraint. If the INSERT statement contains a subquery, match the contents of the table addressed by the subquery and the contents of the object table to determine the cause of the problem.

For a DELETE statement, examine the index key columns in the table that defines the index. These columns contain a foreign key, which when set NULL on a cascade delete from the object table, causes the duplicate values.

SQLSTATE: 23505

-804 **AN ERROR WAS FOUND IN THE APPLICATION PROGRAM INPUT PARAMETERS FOR THE SQL STATEMENT, REASON** *reason*

Explanation: The call parameter list or the SQLDA is invalid.

- The call parameter list, which is created by the precompiler, might be invalid if the application programmer has modified the output of the precompiler, used a variable name beginning with 'SQL' in the application program, or overwritten the call parameter list in some other way.
- The SQLDA, which is created by the application program, has an invalid data type or data length.
- The value of SQLDABC is not consistent with the value of SQLD.

The following is the list of reason codes:

01 Open issued for non-cursor.

02 Close issued for non-cursor.

03 Prepare of EXECUTE IMMEDIATE.

04 Statement is not recognized.

05 No statement string present.

06 Bad SQLDA format in parameter list.

07 SQLDA length is invalid.

08 Unrecognized input data type.

09 Invalid length for input variable.

10 Invalid data length for output variable.

11 The value of SQLDABC is not consistent with the value of SQLD.

12 Invalid input data pointer.

13 Invalid output data pointer.

14 SQLN has too many items for SQLDABC.

15 Input RDI pointer is invalid.

16 Unrecognized output data type.

17 The value of the 7th byte of SQLDAID is not consistent with the data types contained in the SQLDA. The SQLDA contains a LOB type host variable, but the 7th byte of SQLDAID is not set to '2' or greater to indicate that the extended SQLVARs have been allocated.

System Action: The statement cannot be executed.

System Programmer Response: Examine the application program for any of the errors noted under the explanation above. In general, the application programmer should not attempt to modify the output of the precompiler.

SQLSTATE: 07002

-805 **DBRM OR PACKAGE NAME** *location-name.collection-id.dbrm-name.consistency -token* **NOT FOUND IN PLAN** *plan-name***. REASON** *reason*

Explanation: An application program attempted to use a DBRM or package 'location-name.collection-id.dbrm-name.consistency-token' that was not found.

Collection id is blank ('location-name..dbrm-name.consistency-token') if the CURRENT PACKAGESET special register was blank for the local program execution.

The REASON token is blank if the length of 'location-name' is 16, the length of 'collection-id' is 18, and the length of 'dbrm-name' is 8 due to the length of SQLERRMT.

The DBRM or package name was not found for one or more of the following reasons:

- 01

  - The DBRM name was not found in the member list of the plan and there is no package list for the plan. Refer to the first SQL statement under problem determination for assistance in determining the problem.
  - The package name was not found because there is no package list for the plan. Refer to the second SQL statement under Problem Determination for assistance in determining the problem.
- 02

  The DBRM name 'dbrm-name' did not match an entry in the member list or the package list. Any of the following conditions could be the problem:

  Bind conditions:
  - The 'collection-id' in the package list was not correct when the application plan 'plan-name' was bound. Refer to the second SQL statement under Problem Determination for assistance in determining the problem.
  - The 'location-name' in the package list was not correct when the application 'plan-name' was bound. Refer to the second SQL statement under Problem Determination for assistance in determining the problem.
  - The 'location-name' in the CURRENTSERVER option for the bind subcommand was not correct when the application plan 'plan-name' was bound. Refer to the third SQL statement under Problem Determination for assistance in determining the problem.

  Application conditions:
  - The CURRENT PACKAGESET special register was not set correctly by the application.
  - The application was not connected to the proper location.
- 03

  The DBRM name 'dbrm-name' matched one or more entries in the package list and the search of those entries did not find the package. The conditions listed under reason 02 or the following conditions might be the problem.
  - The DBRM of the version of the application program being executed was not bound (A package with the same consistency token as that of the application program was not found.) Refer to the fourth and fifth SQL statements under the Problem Determination section.
  - The incorrect version of the application program is being executed.
- 04

  The package, 'collection-id.dbrm-name.consistency-token', does not exist at the remote site, 'location-name'. Refer to the fifth SQL statement under the Problem Determination section.

System Action: The statement cannot be executed.

System Programmer or Programmer response: Based on the above reasons, the programmer can perform **one or more** of the following operations for each reason to correct the error.

- 01
  - Add the DBRM name 'dbrm-name' to the MEMBER list of the BIND subcommand and bind the application plan 'plan-name', or
  - Add the PKLIST option with the appropriate package list entry to the REBIND subcommand and rebind the application plan 'plan-name'.
- 02
  - Correct the dbrm-name of the entry in the PKLIST option and use the REBIND subcommand to rebind the application plan 'plan-name', or
  - Correct the location-name of the entry in the PKLIST option and use the REBIND subcommand to rebind the application plan 'plan-name', or
  - Correct the location-name in the CURRENTSERVER option and use the REBIND subcommand to rebind the application plan 'plan-name', or
  - Set the CURRENT PACKAGESET special register correctly, or
  - Connect to the correct location name.
- 03

  All the operations under reason 02 above might fix the problem, plus the following operations.
  - Correct the collection-id of the entry in the PKLIST option and use the REBIND subcommand to rebind the application plan 'plan-name', or
  - Bind the DBRM of the version of the application program to be executed into the collection 'collection-id', or
  - Execute the correct version of the application program. The consistency token of the application program is the same as the package that was bound.
- 04

  All the operations under reason 02 and 03 might fix the problem.

Problem Determination: The following queries aid in determining the problem. Run these queries at the local location.

1. This query displays the DBRMs in the member list for the plan. If no rows are returned, then the plan was bound without a member list.

   ```
   SELECT PLCREATOR, PLNAME, NAME, VERSION
   FROM SYSIBM.SYSDBRM
   WHERE PLNAME = 'plan-name';
   ```
2. This query displays the entries in the package list for the plan. If no rows are returned, then the plan was bound without a package list.

```
SELECT LOCATION, COLLID, NAME
FROM SYSIBM.SYSPACKLIST
WHERE PLANNAME = 'plan-name';
```

3. This query displays the CURRENTSERVER value specified on the BIND subcommand for the plan.

```
SELECT NAME, CURRENTSERVER
FROM SYSIBM.SYSPLAN
WHERE NAME = 'plan-name';
```

4. This query displays if there is a matching package in SYSPACKAGE. If the package is remote, put the location name in the FROM clause. If no rows are returned, the correct version of the package was not bound.

```
SELECT COLLID, NAME, HEX(CONTOKEN), VERSION
FROM <location-name.>SYSIBM.SYSPACKAGE
WHERE NAME = 'dbrm-name'
AND HEX(CONTOKEN) = 'consistency-token';
```

5. This query displays if there is a matching package in SYSPACKAGE. If the package is remote, put the location name in the FROM clause. Use this query when collection-id is not blank. If no rows are returned, the correct version of the package was not bound.

```
SELECT COLLID, NAME, HEX(CONTOKEN), VERSION
FROM <location-name.>SYSIBM.SYSPACKAGE
WHERE NAME = 'dbrm-name'
AND HEX(CONTOKEN) = 'consistency-token'
AND COLLID = 'collection-id';
```

SQLSTATE: 51002

-807 **ACCESS DENIED: PACKAGE** *package-name* **IS NOT ENABLED FOR ACCESS FROM** *connection-type connection-name*

Explanation: Access is denied for one of the following reasons:

- It is disabled, either from the 'connection-type' or from the 'connection-type' with the specific 'connection-name'.
- The attach library that you are using is from a previous release of DB2 that does not support the ENABLE and DISABLE options of the bind operation.

The variables are:

package-name
The package name (collection.package-id).

connection-type
One of the following: BATCH, DB2CALL, REMOTE, IMSBMP, IMSMPP, CICS, DLIBATCH, or UNKNOWN.

connection-name
Name of the connection that is restricted. If all connection names from a specific connection-type are restricted, this value is not specified.

System Action: The statement is not executed and the package is not allocated.

System Programmer Response: One of the following:

- Rebind the package to enable it to execute with the required connection type and name.
- Check the SYSPLSYSTEM or SYSPKSYSTEM catalog table to find a connection from which the package can be executed.
- Correct the attach library.

SQLSTATE: 23509

-808 **THE CONNECT STATEMENT IS NOT CONSISTENT WITH THE FIRST CONNECT STATEMENT**

Explanation: The CONNECT semantics that apply to an application process are determined by the first CONNECT statement executed (successfully or unsuccessfully) by the application process. One of the following rules was violated:

- A type 2 CONNECT statement cannot be executed after a type 1 CONNECT statement was executed.
- A type 1 CONNECT statement cannot be executed after a type 2 CONNECT statement was executed.

System Action: The statement cannot be executed.

Programmer Response: The probable cause of this error is that different programs in the application process were precompiled with different CONNECT options.

Ensure that the application process uses either type 1 or type 2 CONNECT statements and then resubmit the job. The type of CONNECT to be used is a precompiler option. The default is type 2 CONNECT.

SQLSTATE: 08001

-811 **THE RESULT OF AN EMBEDDED SELECT STATEMENT OR A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE**

Explanation: Execution of an embedded SELECT statement or a subselect in the SET clause of an UPDATE statement has resulted in a result table that contains more than one row. Alternatively, a subquery contained in a basic predicate has produced more than one value.

System Action: The statement cannot be executed.

Programmer Response: Examine the syntax of the statement to ensure that it contains the proper condition specifications. If the statement syntax is correct, there might be a problem with the data that is causing more

than one row or value to be returned when you do not expect it.

SQLSTATE: 21000

-812 **THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE A BLANK COLLECTION-ID WAS FOUND IN THE CURRENT PACKAGESET SPECIAL REGISTER WHILE TRYING TO FORM A QUALIFIED PACKAGE NAME FOR PROGRAM** *program-name.consistency-token* **USING PLAN** *plan-name*

Explanation: A last or only entry in the package list for the plan contained asterisk (*) the 'collection-id ' which requires the 'CURRENT PACKAGESET' special register to be set to a nonblank 'collection-id' in order to form a qualified package name.

System Action: The statement cannot be executed.

Programmer Response: Set the 'CURRENT PACKAGESET' special register to the desired 'collection-id' or have your system administrator check the plan's package list for correctness.

SQLSTATE: 22508

-815 **A GROUP BY OR HAVING CLAUSE IS IMPLICITLY OR EXPLICITLY SPECIFIED IN A SUBSELECT OF A BASIC PREDICATE OR THE SET CLAUSE OF AN UPDATE STATEMENT**

Explanation: A subselect of a basic predicate or a SET clause of an UPDATE statement either:

- directly contains a GROUP BY or HAVING clause
- specifies as its object a view having a definition that includes a GROUP BY or HAVING clause

Neither construct is permitted.

System Action: The statement cannot be executed. No data was retrieved.

Programmer Response: The implied function is not supported by DB2.

No coding workaround exists for the subselect. A GROUP BY or HAVING clause cannot be used within the subselect of a basic predicate because the subselect is allowed to return only a single value. For more information on basic predicates, refer to *DB2 SQL Reference*.

SQLSTATE: 42920

-817 **THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE STATEMENT WILL RESULT IN A PROHIBITED UPDATE OPERATION.**

Explanation: The application attempted to execute an SQL statement that would result in updates to user data or to the subsystem catalog. This is prohibited for one of the following reasons:

- The application is running as an IMS inquiry-only transaction.
- The application is an IMS or CICS application that is attempting to update data at a remote DBMS that does not support two-phase commit.
- The application is attempting to update data at multiple locations and one of the locations does not support two-phase commit.
- A trigger defined with activation time BEFORE was activated and its triggered action caused updates to the database.

These SQL statements include INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, GRANT, and REVOKE.

System Action: The statement cannot be executed.

Programmer Response: If the application is running as an IMS inquiry-only transaction, see your IMS system programmer about changing the inquiry-only status of the transaction under which your application is running.

If the IMS or CICS application is attempting a remote update, either the application must be changed to run as a local application on the server DBMS, or the server DBMS must be upgraded to support two-phase commit.

If the application is attempting to update data at multiple locations, either the application must be changed, or all DBMSs involved must be upgraded to support two-phase commit.

If the error is due to an invalid statement during a trigger activation, contact your system administrator to correct the trigger definition.

SQLSTATE: 25000

-818 **THE PRECOMPILER-GENERATED TIMESTAMP** *x* **IN THE LOAD MODULE IS DIFFERENT FROM THE BIND TIMESTAMP** *y* **BUILT FROM THE DBRM** *z*

Explanation: The SQL precompiler places timestamp 'y' in the DBRM, and time stamp 'x' in the parameter list in the application program for each SQL statement. At BIND time, DB2 stores the DBRM timestamp for run-time use. At run-time, timestamp 'x', for the SQL statement being processed, is compared with timestamp 'y' derived from the DBRM 'z' at BIND time. If the two timestamps do not match, the DBRM and the

application program were not the result of the same precompile.

This problem can occur if you:
- Precompile, compile, and link, without doing a BIND of the application,
- Precompile and BIND, without doing the compile and link for the application program, or
- BIND the application using a DBRM that resulted from a different precompile of the application program than that which produced the object module that is linked into the application module.

The timestamps 'x' and 'y' are DB2 internal timestamps. They do not have an external interpretation.

System Action: The statement cannot be executed.

Programmer Response: BIND the application again, using the DBRM for the application program that matches the object module.

SQLSTATE: 51003

-819 THE VIEW CANNOT BE PROCESSED BECAUSE THE LENGTH OF ITS PARSE TREE IN THE CATALOG IS ZERO

Explanation: SYSIBM.SYSVTREE.VTREE is a varying-length string column that contains the parse trees of views. In processing a view, the length control field of its parse tree was found to be zero.

System Action: The statement cannot be executed.

Programmer Response: This is a system error. If you suspect an error in DB2, refer to Part 2 of *DB2 Diagnosis Guide and Reference* for information on identifying and reporting the problem.

SQLSTATE: 58004

-820 THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE *catalog-table* CONTAINS A VALUE THAT IS NOT VALID IN THIS RELEASE

Explanation: A column of the indicated catalog table contains a value that prevents further processing of an SQL statement. The meaning of the value is unknown to the release of DB2. If a fall back has occurred, the value is probably the result of the use of new function prior to the fall back.

System Action: The statement cannot be executed.

Programmer Response: Verify that the statement refers to the intended tables or views and that the problem is the result of a fall back. If this is the case, the statement cannot be corrected because it depends on a function that is not supported in the current release. If the problem is not the result of a fallback, -820 is a system error. If you suspect an error in DB2, refer to Part 2 of *DB2 Diagnosis Guide and Reference* for information on identifying and reporting the problem.

SQLSTATE: 58004

-822 THE SQLDA CONTAINS AN INVALID DATA ADDRESS OR INDICATOR VARIABLE ADDRESS

Explanation: The application program has placed an invalid address in the SQLDA.

System Action: The statement cannot be executed.

Programmer Response: Correct the application program such that valid addresses are placed in SQLDA.

SQLSTATE: 51004

-840 TOO MANY ITEMS RETURNED IN A SELECT OR INSERT LIST

Explanation: The number of items returned in the select list or presented in an insert list exceeds the allowable maximum of 750.

System Action: The statement cannot be executed.

Programmer Response: Determine whether all the information is actually needed. (Note that the number of items returned by the select list * in the SQL statement SELECT * FROM A, B, C is the sum of the number of columns in all three tables.) If not, rewrite the SQL statement so that only the necessary items of information are returned. If so, break the SQL statement up into two or more statements, as required.

SQLSTATE: 54004

-842 A CONNECTION TO *location-name* ALREADY EXISTS

Explanation: One of the following situations occurred:
- A CONNECT statement identifies a location with which the application process has a private connection, using system-directed access.
- SQLRULES(STD) is in effect and a CONNECT statement identifies an existing SQL connection.
- A private connection, using system-directed access, cannot be established because of an existing SQL connection to that location.
- A CONNECT (type 2) request that includes the USER/USING clause identifies an existing SQL connection.

System Action: The statement cannot be executed.

Programmer Response: The correction depends on the error, as follows:
- If the location name is not the intended name, correct it.
- If SQLRULES(STD) is in effect and the CONNECT statement identifies an existing SQL connection, replace the CONNECT with SET CONNECTION or change the option to SQLRULES(DB2).

- If the CONNECT statement identifies an existing private connection, destroy that connection (by using the RELEASE statement in a previous unit of work) before executing the CONNECT statement. If the SQL statements following the CONNECT can be executed using system-directed access, an alternative solution is to change the application to use that method.
- If system-directed access cannot be used, destroy the conflicting SQL connection (by using the RELEASE statement in a previous unit of work) before executing the SQL statement that requires system-directed access. An alternative solution is to change the application so that only application-directed access is used.
- Destroy the connection (by using the RELEASE statement in a previous unik of work) before executing the CONNECT statement which includes the USER/USING clause.

Correct the error in the application, rebind the plan or package, and resubmit the job.

SQLSTATE: 08002

-843 THE SET CONNECTION OR RELEASE STATEMENT MUST SPECIFY AN EXISTING CONNECTION

Explanation: One of the following rules was violated:

- A SET CONNECTION statement must identify an existing SQL connection of the application process.
- A RELEASE statement must identify an existing connection of the application process.

System Action: The statement cannot be executed.

Programmer Response: The correction depends on the error, as follows:

- If the location name is not the intended name, correct it.
- If the location name does not identify an existing SQL connection, replace the SET CONNECTION with a CONNECT statement.
- If RELEASE CURRENT was executed in the unconnected state or the specified location name does not identify an existing SQL or DB2 private connection, delete the RELEASE statement.

Correct the error in the application, rebind, the plan and resubmit the job.

SQLSTATE: 08003

-846 INVALID SPECIFICATION OF AN IDENTITY COLUMN

Explanation: For an ALTER or CREATE TABLE statement, the specification of an identity column may be invalid for one of the following reasons:

- The data type of the associated column was not one of the allowed data types for an identity column. The allowed data types are the following:
  - INTEGER
  - SMALLINT
  - DECIMAL with a scale of zero
- An invalid value was specified for INCREMENT BY. This value can be any positive or negative value that could be assigned to this column, but it must be within the range defined for the INTEGER type and can not be 0.
- An invalid value was specified for MINVALUE and/or MAXVALUE. MINVALUE must be less than MAXVALUE.
- The CREATE TABLE statement cannot specify an EDITPROC for a table that has an identity column.
- The ALTER TABLE statement cannot define an identity column on a global temporary table.
- The CREATE TABLE LIKE statement cannot create a table with an edit procedure like another table that has an identity column.

System Action: DB2 cannot process the statement.

Programmer Response: Correct the syntax and resubmit the statement.

SQLSTATE: 42815

-867 INVALID SPECIFICATION OF A ROWID COLUMN

Explanation: For an ALTER or CREATE TABLE statement, the specification of a ROWID column might be invalid for one of the following reasons:

- A ROWID column can not be added to a temporary table.
- The *referential-constraint* clause can not specify a ROWID column as a column of a foreign key.
- A ROWID column can not be a column of a primary key.
- A ROWID column can not be a column in a table with an EDITPROC.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax and resubmit the statement.

SQLSTATE: 428C7

-870 THE NUMBER OF HOST VARIABLES IN THE STATEMENT IS NOT EQUAL TO THE NUMBER OF DESCRIPTORS

Explanation: The number of host variables in the SQL statement does not match the number of host variable descriptors.

System Action: The statement cannot be executed.

Programmer Response: Correct the application program. The most likely cause of this problem is a missing colon before a host variable.

Problem Determination: If the SQL statement is bound locally, descriptors are built by the DB2 precompiler. For a remote SQL statement, descriptors are built by DDF and are passed in the array SQLSTTVRB.

SQLSTATE: 58026

-872 **A VALID CCSID HAS NOT YET BEEN SPECIFIED FOR THIS SUBSYSTEM**

Explanation: A valid CCSID was not specified on either the ASCII CODED CHAR SET, EBCDIC CODED CHAR SET, or UNICODE CODED CHAR SET subsystem parameter on installation panel DSNTIPF.

System Action: The statement cannot be executed.

Programmer Response: Contact your system administrator to have the necessary CCSID defined for your system.

SQLSTATE: 51032

-873 **DATA ENCODED WITH DIFFERENT CCSIDS CANNOT BE REFERENCED IN THE SAME SQL STATEMENT**

Explanation: You cannot refer to a column defined in a table in one encoding scheme in the same SQL statement as a column defined in a table of another encoding scheme.

This situation can occur when a table created in either the ASCII, EBCDIC, or UNICODE encoding schemes is referenced in a statement with a table that is not in the same encoding scheme.

This situation can also occur when a table is created; a DECP CCSID value is changed; another table is created, and then the two tables are referenced in a single SQL statement.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement.

SQLSTATE: 53090

-874 **THE ENCODING SCHEME SPECIFIED FOR THE TABLE IS NOT THE SAME AS THAT USED FOR THE TABLE SPACE CONTAINING THIS TABLE**

Explanation: If CCSID ASCII was specified, then the containing table space is EBCDIC or UNICODE. If CCSID EBCDIC was specified, then the containing table space is ASCII or UNICODE. If CCSID UNICODE was specified, then the containing table space is ASCII or EBCDIC.

The encoding scheme of a table must be the same as the table space which contains the table.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement.

SQLSTATE: 53091

-875 *operand* **CANNOT BE USED WITH THE ASCII DATA REFERENCED**

Explanation: ASCII data was referenced in one of the following situations:

- A LIKE predicate refers to a mixed data column in an ASCII table. The LIKE predicate is not supported for mixed ASCII data.
- A VARGRAPHIC function was specified for a column in an ASCII table. The VARGRAPHIC function is not supported for ASCII data.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement.

SQLSTATE: 42988

-876 **'*object*' CANNOT BE CREATED, REASON '*reason*'**

Explanation: The object cannot be created in the SQL statement.

Possible values for '*object*':

TYPE 1 INDEX
The object being created is a type 1 index.

INDEX The object being created is an index.

Possible values for '*reason*':

TABLE DEFINED AS ASCII
The underlying table is defined as ASCII. Only type 2 indexes are supported for ASCII tables.

PIECESIZE IS NOT VALID
PIECESIZE is only valid for non-partitioned indexes.

PIECESIZE 4GB IS NOT VALID
PIECESIZE 4GB is only valid for non-partitioned indexes on LARGE tables.

System Action: The statement cannot be executed.

Programmer Response: Correct the SQL statement for the object being created.

SQLSTATE: 53092

-877 **CCSID ASCII OR CCSID UNICODE IS NOT ALLOWED FOR THIS DATABASE OR TABLE SPACE**

Explanation: The database or table space specified is required to be in EBCDIC.

System Action: The statement cannot be executed.

Programmer Response: Remove the CCSID ASCII or CCSID UNICODE clause from the statement.

SQLSTATE: 53093

-878 THE PLAN_TABLE USED FOR EXPLAIN CANNOT BE ASCII OR UNICODE

Explanation: PLAN_TABLEs must be encoded in EBCDIC for use with EXPLAIN.

System Action: The statement cannot be executed.

Programmer Response: Drop the existing PLAN_TABLE, and recreate it with the EBCDIC encoding scheme.

SQLSTATE: 53094

-879 CREATE or ALTER STATEMENT FOR *obj-name* CANNOT DEFINE A COLUMN, DISTINCT TYPE, FUNCTION OR STORED PROCEDURE PARAMETER AS MIXED OR GRAPHIC WITH ENCODING SCHEME *encoding-scheme*

Explanation: A CREATE or ALTER TABLE statement for *object-name* attempted to define a column, distinct type, or parameter of a user-defined function or stored procedure as mixed data or graphic when the system does not have an appropriate CCSID defined for the *encoding-scheme* encoding scheme.

- A CREATE, with CCSID UNICODE clause specified, cannot be processed because the proper level of OS/390 is not installed. Refer to the Program Directory for information on the level of OS/390 required for UNICODE support.
- A CREATE DISTINCT TYPE statement cannot define a distinct type, on EBCDIC or ASCII data, with a source type of CHAR FOR MIXED DATA, or GRAPHIC, VARGRAPHIC, or DBCLOB, when the MIXED DATA install option is set to NO.
- A CREATE FUNCTION or CREATE PROCEDURE statement cannot define a parameter or specify a RETURNS data type, for ASCII or EBCDIC data, as CHAR FOR MIXED DATA, or GRAPHIC, VARGRAPHIC, or DBCLOB, when the MIXED DATA install option is set to NO.

Note: This error only occurs when the encoding scheme in use is EBCDIC or ASCII. The MIXED DATA install option does not affect Unicode data.

Note: This error can occur when there is an attempt to define a column or parameter as character FOR MIXED DATA even though the keywords FOR MIXED DATA do not appear in the failing statement. This occurs when the MIXED value in DECP is YES, in this case the default subtype for the character types is FOR MIXED DATA.

System Action: The statement cannot be processed.

Programmer Response: Contact your system administrator to properly setup the installation options, or change the data types of the elements in columns in your CREATE or ALTER statement.

SQLSTATE: 53095

-880 SAVEPOINT *savepoint-name* DOES NOT EXIST OR IS INVALID IN THIS CONTEXT

Explanation: The RELEASE TO SAVEPOINT or ROLLBACK TO SAVEPOINT statement does not identify a savepoint that exists.

System Action: DB2 does not process the statement.

Programmer Response: Correct the statement to use a valid savepoint name.

SQLSTATE: 3B001

-881 A SAVEPOINT WITH NAME *savepoint-name* ALREADY EXISTS, BUT THIS SAVEPOINT NAME CANNOT BE REUSED

Explanation: The SAVEPOINT statement uses the same savepoint name as another savepoint, and it cannot be created because at least one of the savepoints was defined with the UNIQUE clause to indicate that the name cannot be reused within the transaction.

System Action: The statement is not executed and a new savepoint is not set. The old savepoint still exists.

Programmer Response: Correct the statement. Either use a different savepoint name, or omit the UNIQUE clause if the other savepoint was created without the UNIQUE clause and your intention is to reuse that savepoint name.

SQLSTATE: 3B501

-882 SAVEPOINT DOES NOT EXIST

Explanation: A ROLLBACK TO SAVEPOINT statement was specified without a savepoint name to rollback to the last active savepoint, but no savepoint exists.

System Action: The statement is not executed.

Programmer Response: Correct the application logic to either set a savepoint or to not attempt to rollback to a savepoint.

SQLSTATE: 3B502

-900 **THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE APPLICATION PROCESS IS NOT CONNECTED TO AN APPLICATION SERVER**

Explanation: A previous failure has placed the application process in the unconnected state. The only SQL statements that can be successfully executed from the unconnected state are CONNECT, COMMIT, ROLLBACK, and local SET statements.

System Action: The statement cannot be executed.

Programmer Response: It is possible that no response is required because checking for -900 is one way of detecting the unconnected state. If this is not the case, the logic of the application program must be changed. For additional information, see the description of the CONNECT statement in Chapter 6 of *DB2 SQL Reference* .

SQLSTATE: 08003

-901 **UNSUCCESSFUL EXECUTION CAUSED BY A SYSTEM ERROR THAT DOES NOT PRECLUDE THE SUCCESSFUL EXECUTION OF SUBSEQUENT SQL STATEMENTS**

Explanation: A system error occurred that prevented successful execution of the current SQL statement. However, the error does not prevent successful execution of further SQL statements.

The error might occur if the length of the SQL statement is less than 0, or is greater than the DB2 maximum length for a statement.

This SQLCODE might also occur when distributed commit processing encounters an error. In this case, all servers in the unit of work that support distributed two-phase commit backed out the unit of work. If a server that does not support distributed two-phase commit has updates in the unit of work, that server must be queried to determine if its updates were committed or backed out.

Finally, this error might occur during commit if post-processing cannot be completed because of an update that changes the partition of a row. Resource unavailable problems can prevent the post-processing from completing. The existence of held cursors can prevent the post-processing from completing. Commit is failed. The transaction is aborted.

System Action: The statement cannot be executed. A X'04E' abend might be requested for the application. The application program can have a recovery routine to recover from such an abend and can retry SQL statements.

Programmer Response: If an abend occurred, notify the system programmer for analysis of the abend that caused this return code.

Even if an abend occurred, an application program receiving this return code can retry and is not prohibited from executing further SQL statements.

SQLSTATE: 58004

-902 **POINTER TO THE ESSENTIAL CONTROL BLOCK (CT/RDA) HAS VALUE 0, REBIND REQUIRED**

Explanation: Pointer to the essential control block, either the CT or the RDA is zeroes. This precludes the successful execution of the current SQL statement, as well as any subsequent SQL statements.

System Action: The statement cannot be executed. The application program is not permitted to issue additional SQL statements. For example, a recovery routine associated with the application program may not issue additional SQL statements.

Programmer Response: Rebind the failing application program and try again. If the problem persists, examine your DBRM and make sure it matches your program.

SQLSTATE: 58005

-904 **UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON** *reason-code*, **TYPE OF RESOURCE** *resource-type*, **AND RESOURCE NAME** *resource-name*

Explanation: The SQL statement could not be executed because resource 'resource-name' of type 'resource-type' was not available at the time for the reason indicated by 'reason-code'. Refer to Table 3 in “Appendix B. Problem determination” on page 1291 for an explanation of resource type codes. Refer to “Part 4. Section 4. DB2 Codes” on page 715 for an explanation of the given reason code.

System Action: The SQL statement cannot be executed. If the SQL statement being executed was a cursor FETCH, DB2 closes the cursor. Subsequent attempts to use that cursor without first doing an OPEN for it receive an SQLCODE -501.

Programmer Response: Verify the identity of the resource that was not available. To determine why the resource was unavailable, refer to the specified 'reason-code'.

SQLSTATE: 57011

-905 **UNSUCCESSFUL EXECUTION DUE TO RESOURCE LIMIT BEING EXCEEDED, RESOURCE NAME =** *resource-name* **LIMIT =** *limit-amount1* **CPU SECONDS (***limit-amount2* **SERVICE UNITS) DERIVED FROM** *limit-source*

Explanation: The execution of the SQL statement was terminated because a resource limit was exceeded.

resource-name
The name of the resource whose limit was exceeded. It is also the name of the column in the DB2 table from which the limit was derived. The *resource-name* can be ASUTIME, which is the number of CPU seconds permitted for each SQL statement.

limit-amount1
The maximum number of CPU seconds permitted

limit-amount2
The maximum number in service units permitted

limit-source
The source used to derive the limit-amount: the name of a resource limit specification table, a system parameter, or the SYSIBM.SYSROUTINES catalog table. If the source is a system parameter, the resource limit specification table did not contain an applicable entry or an error occurred while accessing the table.

System Action: If the *limit-source* was a resource limit specification table or a system parameter, the execution of this SQL statement is terminated. A record containing more detailed information about this failure is generated. If an SQL cursor is associated with the failed instruction, its position is unchanged and a CLOSE or PREPARE command can be issued. If any other operation is attempted with the cursor, it cannot be executed and SQLCODE -905 is returned. If there is no cursor, this statement was rolled back.

Programmer Response: Determine why this SQL statement or stored procedure took so long and take appropriate action. Consider simplifying the SQL statement, restructuring tables and indexes, or contacting the installation group responsible for maintaining the resource limit specification tables.

If the *limit-source* was a resource limit specification table or a system parameter, the application program that receives this return code can execute additional SQL statements.

SQLSTATE: 57014

-906 **THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THIS FUNCTION IS DISABLED DUE TO A PRIOR ERROR**

Explanation: Execution of the SQL statement failed because the requested function had been disabled by a prior error. This situation can arise if the application program has intercepted an abend (for instance, by an ON ERROR condition in a PL/I program) and continued to execute SQL statements. This situation may also arise if a DB2 CICS transaction encountered a create thread error yet continued to issue SQL requests without issuing a SYNCPOINT ROLLBACK first.

System Action: The statement cannot be executed.

Programmer Response: In general, an application program should terminate upon receipt of this return code. All subsequent attempts by the application to execute other SQL statements will also fail with the same return code. In the case of a DB2 CICS transaction, if the SQLERRP field in the SQLCA contains the module name DSNCEXT1, the transaction may issue a SYNCPOINT ROLLBACK and continue processing. If the transactions chooses to ROLLBACK and continue processing, it must be capable of correcting the situation that caused the create thread error to occur originally.

SQLSTATE: 51005

-908 *bind-type* **ERROR USING** *auth-id* **AUTHORITY. BIND, REBIND OR AUTO-REBIND OPERATION IS NOT ALLOWED**

Explanation: For BIND and REBIND, the indicated authorization ID is not allowed to perform the indicated bind-type against a plan or package. An entry in the resource limit specification table (RLST) prohibits binding and rebinding by this authorization ID, or all authorization IDs. For AUTO-REBIND, the system parameter controlling AUTO-REBIND operations is set to disallow AUTO-REBIND.

bind-type
Type of bind operation (BIND, REBIND or AUTO-REBIND).

auth-id Authorization ID of the invoker of the BIND subcommand or primary authorization ID of the invoker of the plan for AUTO-REBIND operations.

System Action: The plan or package is not bound.

System Programmer Response: If the indicated authorization id should be allowed to bind, change the entry in the active RLST table. If AUTO-REBIND operations are disabled, rebind the package before reexecuting the package.

SQLSTATE: 23510

-909 **THE OBJECT HAS BEEN DELETED**

Explanation: The application program has either:

1. Dropped a table and then attempted to accesss it.
2. Dropped an index and then tried to access its object table using that index.

System Action: The statement cannot be executed.

System Programmer Response: The logic of the application program must be corrected such that it does not attempt to access or use an object after it has been dropped.

Dropping indexes within an application program is

especially hazardous, because there is no way of determining whether or not the plan that has been generated for the application (by BIND or REBIND) actually uses a particular index for access to its object table.If the indicated authorization id should be allowed to bind, change the entry in the active RLST table. If AUTO-REBIND operations are disabled, rebind the package before reexecuting the package.

SQLSTATE: 57007

-910 **THE SQL STATEMENT CANNOT ACCESS AN OBJECT ON WHICH A DROP OR ALTER IS PENDING**

Explanation: The application program has issued a DROP or ALTER against an object, and then attempted to access that object before the DROP or ALTER is completed.

System Action: The statement cannot be executed.

Programmer Response: In the case of ALTER, the logic of the application program must be modified so that a COMMIT (or the IMS or CICS equivalent) is executed between the ALTER and the failing SQL statement.

For DROP, the logic of the application program should be modified such that there is no attempt to access an object after the DROP has been executed.

Note that DROP includes the case when rollback to a savepoint includes rolling back to a CREATE.

SQLSTATE: 57007

-911 **THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON** *reason-code*, **TYPE OF RESOURCE** *resource-type*, **AND RESOURCE NAME** *resource-name*

Explanation: The current unit of work was the victim in a deadlock, or experienced a timeout, and had to be rolled back.

The reason code indicated whether a deadlock or timeout occurred. Refer to message DSNT500I under “Chapter 17. DSNT... Messages” on page 377 for an explanation of 'resource-type' and 'resource-name'. Refer to Table 3 in “Appendix B. Problem determination” on page 1291 for an explanation of resource type codes.

Note: The changes associated with the unit of work must be entered again.

SQLERRD(3) also contains the reason-code which indicates whether a deadlock or timeout occurred. The most common reason codes are:

- 00C90088 - deadlock
- 00C9008E - timeout

The changes associated with the unit of work must be entered again.

System Action: The statement cannot be executed. The application is rolled back to the previous COMMIT.

Programmer Response: A long-running application, or an application that is likely to encounter a deadlock, should (if possible) issue frequent COMMIT commands. This can lessen the possibility of a deadlock occurring. See message DSNT376I for other possible ways to avoid future deadlocks or timeouts. On receipt of the SQLCODE -911, the application should, in general, terminate.

For more information about how IMS, CICS, and TSO handle deadlocks, see Part 4 of *DB2 Application Programming and SQL Guide*.

SQLSTATE: 40001

-913 **UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT. REASON CODE** *reason-code*, **TYPE OF RESOURCE** *resource-type*, **AND RESOURCE NAME** *resource-name*

Explanation: The application was the victim in a deadlock or experienced a timeout. The reason code indicates whether a deadlock or timeout occurred.

Refer to message DSNT500I under “Chapter 17. DSNT... Messages” on page 377 for an explanation of 'resource-type' and 'resource-name'. Refer to Table 3 in “Appendix B. Problem determination” on page 1291 for an explanation of resource type codes.

System Action: The SQL statement cannot be executed. If the SQL statement being executed was a cursor FETCH, DB2 closes the cursor.

SQLERRD(3) also contains the reason-code which indicates whether a deadlock or timeout occurred. The most common reason codes are:

- 00C90088 - deadlock
- 00C9008E - timeout

Programmer Response: The application should either commit or roll back to the previous COMMIT. Then, generally, the application should terminate. See message DSNT376I for possible ways to avoid future deadlocks or timeouts.

For more information about how CICS and TSO handle deadlocks, see Part 4 of *DB2 Application Programming and SQL Guide*.

SQLSTATE: 57033

-917 BIND PACKAGE FAILED

Explanation: An error has occurred which prevents the package from being created. This SQLCODE can be issued during bind or commit processing.

System Action: The bind fails and the package is not created. If issued during commit processing, all changes to the database are rolled back. If issued during bind processing, only package creation fails. Other changes within the logical unit of work are committable.

Programmer Response: Correct the cause of the problem and try again.

Problem Determination: Inspect the SQLCODES issued for the SQL statements of the package.

SQLSTATE: 42969

-918 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE A CONNECTION HAS BEEN LOST

Explanation: Execution of the SQL statement failed because a communications link between the local DB2 and at least one remote server no longer exists. A previous failure caused this condition.

System Action: In the IMS and CICS environments, all SQL statements are rejected until the rollback occurs. In the other environments, all SQL statements other than a static ROLLBACK are rejected until a static ROLLBACK is executed.

Programmer Response: In general, an application program should issue a static ROLLBACK. Attempts by the application to issue SQL statements other than static ROLLBACK might fail. Once the static ROLLBACK is issued, the application can resume execution.

SQLERRP contains the name of the module that detected the previous failure and placed the application in the must-abort state.

SQLSTATE: 51021

-919 A ROLLBACK OPERATION IS REQUIRED

Explanation: The unit of work was placed in a state where a rollback operation is required. This can happen for the following reasons:

- An SQL statement updated a distributed database server, but the database server can be used only for read-only operations. Either updates are currently restricted to servers that support distributed two-phase commit and this application server does not support distributed two-phase commit, or updates are restricted to a single server that does not support distributed two-phase commit and this application server is not that server. The unit of work must be terminated by a rollback operation because the update made (but not committed) at the application server cannot be committed consistently with other current or future updates made to this distributed unit of work.
- An abend occurred during the execution of a stored procedure, or a restricted SQL statement was issued from a stored procedure.
- An abend occurred during the execution of a function, or a restricted SQL statement was issued from a function.

System Action: In the IMS and CICS environments, all SQL statements are rejected until the rollback occurs. In the other environments, all SQL statements other than a static ROLLBACK are rejected until a static ROLLBACK is executed.

Programmer Response: Correct the application, function, or stored procedure, rebind it, and resubmit the job.

SQLERRP contains the name of the module that detected the previous failure and placed the application in the must-abort state.

SQLSTATE: 56045

-922 AUTHORIZATION FAILURE: *error-type* **ERROR. REASON** *reason-code*

Explanation: Authorization failed because of the error indicated by *error-type*

error-type
Types of authorization failure
- User authorization
- Plan access
- Duplicate exit requested
- Installation error
- Connect

reason-code
DB2 reason code associated with authorization failure

System Action: The statement cannot be processed. The connection to DB2 is not established.

Programmer Response: If *error-type* is *user authorization:*, the authorization-ID specified to DB2 through your attachment facility is not valid for DB2. See your system programmer or your CICS, IMS, or TSO system administrator.

If *error-type* is *plan access*, then the authorization ID associated with this connection is not authorized to use the specified plan name or the specified plan name does not exist. See your system administrator.

If *error-type* is *duplicate exit*, then you requested a duplicate exit.

If *error-type* is *installation error*, a connection or sign-on exit denied your request. See your system programmer.

If *error-type* is *Connect*, an SQL CONNECT request failed to connect to the local DB2 with USER/USING specified. See the reason code for a description of the failure. The application program has been palced in the connectable and unconneted state. The only SQL statements that can be successfully completed in this state are CONNET, COMMIT, ROLLBACK, and local SET statements. Any attempt to execute other SQL statements will result in an error (SQLCODE -900).

Look up the reason code in "Part 4. Section 4. DB2 Codes" on page 715 for further information.

Any attempts to issue SQL statements following the -922 SQLCODE when *error-type* is not *Connect* causes unpredictable results.

SQLSTATE: 42505

-923 **CONNECTION NOT ESTABLISHED: DB2** *condition* **REASON** *reason-code*, **TYPE** *resource-type*, **NAME** *resource-name*

Explanation: The connection to DB2 failed for the reason indicated by *condition*, which can be any of the following:

- DB2 not up
- DB2 not operational
- DB2 shutdown in progress
- DB2 restricted access mode
- Allocation error
- DB2 - CICS attachment not up
- DB2 - CICS ENTRY disabled
- The object is dependent on facilities of a release of DB2 that is newer than the release that you are currently running (fall back).

Possible causes of an allocation error are:

- The application plan does not exist.
- The application plan is inoperative. An explicit REBIND or BIND is required.
- The application plan is invalid. Underlying resources have changed.
- A required database, table space, table, or index is unavailable.
- Data set allocation failed for a required data set.
- There is insufficient virtual storage.
- The application is trying to execute the plan from a system (environment) that was restricted when the plan was bound or rebound. Check the SYSPLSYSTEM table to determine from which systems (for example, IMS or CICS) the plan can be executed.

If the *condition* is ″CICS attachment not up″, then NAME indicates the DB2 subsystem that is not available. The *reason code* indicates the reason the attachment is not available.

If the *condition* is ″CICS entry disabled″, then NAME indicates the entry that is disabled.

System Action: The statement cannot be executed. The connection to DB2 is not established.

Programmer Response: If the connection failed because either DB2 or a required database, table space, table, or index was unavailable, wait until it is available before invoking the application again.

If allocation failed for an application plan, REBIND the plan to determine the problem. Error messages are produced explaining why the plan could not be allocated.

For other types of allocation errors, installation action might be required to correct the problem.

For CICS attachment failures, resolve the primary cause as noted by the reason code. Then restart the attachment.

Problem Determination: The *reason-code*, *resource-type*, and *resource-name* might not be available. If they are not available, nothing appears. If they are available, refer to Part 4. Section 4. DB2 Codes for an explanation of the *reason-code*, *resource-type*, and *resource-name*.

Refer to message DSNT500I under Chapter 17. DSNT... Messages for an explanation of resource type and resource name. Refer to Table 3 in "Appendix B. Problem determination" on page 1291 for an explanation of resource type codes. Any attempts to issue SQL statements after receiving SQLCODE -923 will cause unpredictable results.

SQLSTATE: 57015

-924 **DB2 CONNECTION INTERNAL ERROR,** *function-code, return-code, reason-code*

Explanation: Connection to DB2 has failed because of an unexpected internal error, identified by the 'reason-code'.

System Action: The statement cannot be executed. The connection to DB2 is not established.

Programmer Response: Look up the abend 'reason-code' in Part 4. Section 4. DB2 Codes for further information. The requested 'function-code' and 'return-code' may provide additional information. Any attempts to issue SQL statements following the SQLCODE -924 will cause unpredictable results.

SQLSTATE: 58006

-925 **COMMIT NOT VALID IN IMS, CICS OR RRSAF ENVIRONMENT**

Explanation: An application executing in either an IMS or CICS environment or an application executing in an RRSAF environment when DB2 is not the only resource manager has attempted to execute a COMMIT

statement. The SQL COMMIT statement cannot be executed in these environments.

System Action: The statement cannot be executed. No commit is performed.

Programmer Response: The IMS, CICS or RRS protocols should be used to commit work in these environments.

If a stored procedure is being called from IMS or CICS, ensure that the stored procedure is not defined to perform a commit on return.

SQLSTATE: 2D521

-926 ROLLBACK NOT VALID IN IMS, CICS OR RRSAF ENVIRONMENT

Explanation: An application executing in either an IMS or CICS environment or an application executing in an RRSAF environment when DB2 is not the only resource manager has has attempted to execute a ROLLBACK statement. The SQL ROLLBACK statement cannot be executed in these environments.

System Action: The statement cannot be executed. No roll back is performed.

Programmer Response: The IMS, CICS or RRS protocols should be used to rollback work in these environments.

SQLSTATE: 2D521

-927 THE LANGUAGE INTERFACE (LI) WAS CALLED WHEN THE CONNECTING ENVIRONMENT WAS NOT ESTABLISHED. THE PROGRAM SHOULD BE INVOKED UNDER THE DSN COMMAND

Explanation: In the TSO environment, the user has attempted to execute an application program without first establishing the correct execution environment by issuing the DSN command. In the IMS, CICS, or call attachment facility (CAF) environment, the user has attempted to execute an application program that is not using the correct language interface module.

System Action: The statement cannot be executed.

Programmer Response: In the TSO environment, DB2 application programs should be invoked under the RUN subcommand of the DSN command processor. In the IMS, CICS or CAF environment check that the application was link-edited with or is dynamically allocating the correct language interface module. The language interface modules required in each environment are as follows:

- IMS: DFSLI000
- CICS: DSNCLI
- CAF: DSNALI
- TSO: DSNELI

The DYNAM option can result in the incorrect language interface module being loaded at runtime.

SQLSTATE: 51006

-929 FAILURE IN A DATA CAPTURE EXIT: *token*

Explanation: 'token' is an information string provided by DPROP's exit routine which captures data changes in tables defined with DATA CAPTURE CHANGES.

System Action: The information string is placed in in the SQLERRM area of the SQLCA.

System Programmer Response: For documentation of the actions associated with this SQLCODE, refer to the Data Propagator (DPROP) publications.

SQLSTATE: 58002

-939 ROLLBACK REQUIRED DUE TO UNREQUESTED ROLLBACK OF A REMOTE SERVER

Explanation: A dynamic commit was executed preceding the execution of this request. The remote server to which the application was CONNECTed during the dynamic COMMIT successfully committed. However, at least one other remote server (which was read-only) rolled back its portion of the distributed unit of work during the dynamic commit.

To ensure that an application that uses cursor-hold cursors does not incorrectly assume cursor position is being maintained at any remote server that rolled back, the application must perform a rollback operation.

Communications are still established with all remote servers.

System Action: In the IMS and CICS environments, all SQL statements are rejected until the rollback occurs. In the other environments, all SQL statements other than a static ROLLBACK are rejected until a static ROLLBACK is executed.

Programmer Response: Issue the appropriate (depending on the environment) request to cause a rollback. Re-establish any cursor positioning and continue the application with the first request that received the -939 SQLCODE.

SQLERRP contains the name of the module that detected the previous failure and placed the application in the must-abort state.

SQLSTATE: 51021

-947 THE SQL STATEMENT FAILED BECAUSE IT WILL CHANGE A TABLE DEFINED WITH DATA CAPTURE CHANGES, BUT THE DATA CANNOT BE PROPAGATED

Explanation: The DPROP SUPPORT option on the installation panel is set to 2 (support DPROP only). The SQL statement would have changed a table defined with DATA CAPTURE CHANGES. However, the data cannot be propagated because the SQL statement did not originate from an IMS subsystem, or monitor trace class 6 was not active at the beginning of the unit of work for that change.

System Action: The statement is not executed.

System Programmer Response: Take one of the following actions:

- Change the installation option to 1 (no propagation) or 3 (permit changes from any subsystem).
- Change the application program that receives this SQLCODE so that it can be run in an IMS subsystem, and activate monitor trace class 6.

If the installation option is changed to 3, SQL changes to tables defined with DATA CAPTURE CHANGES are permitted from any subsystem, but they are not propagated unless the environment is set up for propagation.

SQLSTATE: 56038

-948 DISTRIBUTED OPERATION IS INVALID

Explanation: The unit of work was initiated before DDF was started, and the application attempted to perform a distributed operation. The unit of work must be terminated by a rollback operation.

System Action: In the IMS and CICS environments, all SQL statements are rejected until a rollback occurs. In the other environments, all SQL statements other than a static ROLLBACK are rejected until a static ROLLBACK is executed.

Programmer Response: An application that performs local database updates before DDF is started cannot perform distributed operations within the same unit of work. The current unit of work must be terminated by a rollback operation and a new unit of work must be initiated before any SQL operations can be performed.

Restart the current unit of work.

SQLSTATE: 56062

-950 THE LOCATION NAME SPECIFIED IN THE CONNECT STATEMENT IS INVALID OR NOT LISTED IN THE COMMUNICATIONS DATABASE

Explanation: One of the following conditions applies:

- The location name is blank.
- The data type of the host variable is not character string.
- The length attribute of the host variable is greater than 16.
- The location name does not appear in the LOCATIONS column of the SYSIBM.LOCATIONS table, nor is it the name of the local DB2 subsystem.

System Action: The statement cannot be executed. The application process is placed in the unconnected state.

Programmer Response: If the location name is specified as the value of a host variable, ensure that the name is left justified in the host variable and, if necessary, padded on the right with blanks. If this is not the problem, either SYSIBM.LOCATIONS must be updated to include the specified name, or the specified name must be changed to match a name in SYSIBM.LOCATIONS.

SQLSTATE: 42705

-981 THE SQL STATEMENT FAILED BECAUSE THE RRSAF CONNECTION IS NOT IN A STATE THAT ALLOWS SQL OPERATIONS, REASON *reason-code***.**

Explanation: The application attempted to execute an SQL operation but the RRSAF connection was not in a state that allows the processing of SQL statements.

System Action: The statement cannot be executed.

Programmer Response: See `reason-code` in "Part 4. Section 4. DB2 Codes" on page 715 for an explanation of the problem. Correct the error in the application, REBIND, and run the application again.

SQLSTATE: 57015

-991 CALL ATTACH WAS UNABLE TO ESTABLISH AN IMPLICIT CONNECT OR OPEN TO DB2. RC1= *rc1* **RC2=** *rc2*

Explanation: Call attach attempted to perform an implicit connect and open as the result of an SQL statement. The connect or open failed with the returned values.

rc1 The value returned in FRBRC1 for the failed CONNECT or OPEN request.

rc2 The value returned in FRBRC2 for the failed CONNECT or OPEN request.

System Action: The statement cannot be executed.

Programmer Response: Verify that the application intended to use the call attachment facility (CAF) as the mechanism to connect to DB2. For functions or stored procedures running in the WLM-established stored procedure address space the application must be link-edited with or dynamically allocate the RRS attachment language interface module (DSNRLI), not CAF.

SQLSTATE: 57015

-1760 **CREATE PROCEDURE FOR *procedure-name* MUST HAVE VALID LANGUAGE AND EXTERNAL CLAUSES**

Explanation: A LANGUAGE or EXTERNAL clause is missing in the CREATE statement for procedure *procedure-name*. This clause must be specified.

System Action: The statement cannot be executed.

Programmer Response: Change the CREATE statement to include the missing clause and reissue the statement.

SQLSTATE: 42601

-2001 **THE NUMBER OF HOST VARIABLE PARAMETERS FOR A STORED PROCEDURE IS NOT EQUAL TO THE NUMBER OF EXPECTED HOST VARIABLE PARAMETERS. ACTUAL NUMBER *sqldanum*, EXPECTED NUMBER *opnum***

Explanation: DB2 received an SQL CALL statement for a stored procedure. However, the number of host variable parameters supplied on the CALL statement does not match the expected number of host variable parameters.

sqldanum
The number of host variable parameters as determined by examining the SQLDA.

opnum The expected number of host variable parameters as determined by parsing the statement.

System Action: The statement cannot be executed.

Programmer Response: If the SQL CALL statement is coded incorrectly, modify the SQL application to provide the correct number of parameters on the SQL CALL statement.

SQLSTATE: 53089

-5012 **HOST VARIABLE *host-variable* IS NOT EXACT NUMERIC WITH SCALE ZERO**

Explanation: HOST VARIABLE *host-variable* was specified, but it is not valid in the context in which it was used. HOST VARIABLE *host-variable* was specified as part of ABSOLUTE or RELATIVE in a FETCH statement. The host variable was not usable.

host-variable
Name of the host variable containing the ABSOLUTE or RELATIVE value.

System Action: The statement cannot be processed.

Programmer Response: Change the host variable to be an exact numeric with a scale of zero.

SQLSTATE: 42618

-20003 **GBPCACHE NONE CANNOT BE SPECIFIED FOR TABLESPACE OR INDEX IN GRECP**

Explanation: GBPCACHE NONE was specified on an ALTER TABLESPACE or ALTER INDEX statement, but the table space, index, or the partition to be altered is in GRECP.

System Action: The statement cannot be processed.

User Response: Use the START DATABASE command to recover the table space or index from the GRECP then STOP the table space or index before reissuing the ALTER statement.

SQLSTATE: 560A7

-20004 **8K or 16K BUFFERPOOL PAGESIZE INVALID FOR A WORKFILE OBJECT**

Explanation: This message is issued when:

- A CREATE or ALTER DATABASE statement for a workfile database specifies an 8K or 16K pagesize in the BUFFERPOOL clause.
- A CREATE or ALTER TABLESPACE statement for a workfile database specifies an 8K or 16K pagesize in the BUFFERPOOL clause.

System Action: The statement cannot be executed.

User Response: Correct the statement to specify a 4K or 32K bufferpool pagesize.

SQLSTATE: 560A8

-20005 **THE INTERNAL ID LIMIT OF *limit* HAS BEEN EXCEEDED FOR OBJECT TYPE *object-type***

Explanation: An internal ID is used to uniquely identify objects of type *object-type*. The limit for internal IDs for this type of object is *limit* and this limit has been exceeded.

This could occur during a CREATE DISTINCT TYPE, a CREATE FUNCTION, or a CREATE PROCEDURE statement. This could also occur during processing for a CREATE TABLE or ALTER TABLE ADD COLUMN statement where a new identity column is created.

System Action: The SQL statement cannot be executed.

SQLSTATE: 54035

-20006 LOBS CANNOT BE SPECIFIED AS PARAMETERS WHEN NO WLM ENVIRONMENT IS SPECIFIED

Explanation: On a CREATE PROCEDURE statement, one or more LOBs (or distinct types based on LOBs) were specified in the parameter list, and the NO WLM ENVIRONMENT option was also specified.

Option NO WLM ENVIRONMENT cannot be used with LOBs in the parameter list for a stored procedure.

System Action: The statement is not executed.

Programmer Response: Either do not specify a LOB as a parameter, or specify a WLM ENVIRONMENT name rather than NO WLM ENVIRONMENT on your CREATE PROCEDURE statement.

SQLSTATE: 53097

-20008 UNSUPPORTED OPTION *keyword* **SPECIFIED**

Explanation: *keyword* is a deprecated feature that was supported in releases prior to DB2 Version 7, and is no longer supported.

For indexes, only one type is currently supported — type 2.

System Action: The statement cannot be executed.

Programmer Response: Correct the syntax of the SQL statement to remove reference to the unsupported keyword. Refer to the *DB2 SQL Reference* for more information.

SQLSTATE: 560A9

-20019 THE RESULT TYPE RETURNED FROM THE FUNCTION BODY CANNOT BE ASSIGNED TO THE DATA TYPE DEFINED IN THE RETURNS CLAUSE

Explanation: The data type of the value returned by the function body must be assignable to the parameter specified in the RETURNS clause.

System Action: The statement cannot be processed.

Programmer Response: Change the RETURNS type or change the expression in the RETURN statement so that the result type of the expression returned from the function body can be made.

SQLSTATE: 42866

-20070 AUXILIARY TABLE *table-name* **CANNOT BE CREATED BECAUSE COLUMN** *column-name* **IS NOT A LOB COLUMN**

Explanation: An auxiliary table cannot be created for a non-LOB column. A CREATE AUXILIARY TABLE statement must refer to a LOB column in the associated base table.

System Action: The statement cannot be executed. The specified table was not created.

Programmer Response: Change the name of the column to correctly refer to a LOB column in the base table.

SQLSTATE: 53098

-20071 WLM ENVIRONMENT NAME MUST BE SPECIFIED *function-name*

Explanation: The WLM ENVIRONMENT option was not specified on CREATE FUNCTION, and there is no default WLM environment for the installation.

System Action: The statement could not be processed.

Programmer Response: Select a WLM ENVIRONMENT name and include it in the CREATE FUNCTION statement. Contact your system administrator to find out the names of the WLM environments that have been defined for your installation.

SQLSTATE: 53099

-20072 *csect-name bind-type bind-subtype* **ERROR USING** *auth-id* **AUTHORITY OPERATION IS NOT ALLOWED ON A TRIGGER PACKAGE** *package-name*

Explanation: The package specified is a trigger package.The bind-type operation cannot be performed against a trigger package using this statement or subcommand.

csect-name
Name of the CSECT that issued the message.

bind-type
Type of bind operation (BIND, REBIND, DROP, or FREE).

bind-subtype
Subtype of bind operation (COPY or blank).

auth-id Authorization ID of the invoker of the BIND, REBIND or FREE subcommand or primary authorization ID of the currently executing process for a DROP statement.

package-name
Name of the package in the following format: 'collection.package'.

System Action: The plan or package is not bound, copied, or freed.

System Programmer Response: A trigger package cannot be explicitly bound. A trigger package cannot be copied. To rebind a trigger package locally, the REBIND TRIGGER PACKAGE subcommand must be used. A trigger package cannot be rebound remotely. To drop a trigger package, the DROP TRIGGER statement must be used.

SQLSTATE: 56052

-20073 **THE FUNCTION *function-name* CANNOT BE ALTERED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS**

Explanation: The *function-name* in an ALTER FUNCTION statement cannot be altered to NOT DETERMINISTIC or EXTERNAL ACTION. It is referenced in one or more existing view definitions.

System Action: The statement cannot be executed.

Programmer Response: Drop the views that reference the function before issuing the ALTER FUNCTION statement.

SQLSTATE: 42927

-20074 **THE OBJECT *object-name* CANNOT BE CREATED BECAUSE THE FIRST THREE CHARACTERS ARE RESERVED FOR SYSTEM OBJECTS**

Explanation: In general, SYS is a reserved prefix for names. The only exceptions are:
- SYSPROC is a valid schema name for stored procedures.
- SYSADM is a valid schema name.

This condition is similar to the condition reported in precompiler message DSNH794I.

This message is also issued if an attempt is made to grant the CREATEIN, ALTERIN or DROPIN privileges on a schema with the SYS prefix. The same exceptions apply to the grant.

System Action: The statement is not executed.

Programmer Response: Select a name that does not start with a reserved prefix.

SQLSTATE: 42939

-20091 **A VIEW NAME WAS SPECIFIED AFTER LIKE IN ADDITION TO THE INCLUDING IDENTITY COLUMN ATTRIBUTES CLAUSE**

Explanation: The LIKE clause specified the name of a view in combination with the INCLUDING IDENTITY COLUMN ATTRIBUTES clause. This usage is not supported.

System Action: The statement is not executed.

Programmer Response: Remove the INCLUDING IDENTITY COLUMN ATTRIBUTES clause and resubmit the statement to copy the existing view definition without the identity column attributes.

In the case of DECLARE GLOBAL TEMPORARY TABLE, it is possible to get the identity column attributes for a column of a view using the AS *subselect* clause with INCLUDING IDENTITY COLUMN ATTRIBUTES instead. For example:

```
DECLARE GLOBAL TEMPORARY TABLE ....
   AS (SELECT * FROM view-name) DEFINITION ONLY
                                INCLUDING IDENTITY COLUMN ATTRIBUTES
```

SQLSTATE: 560AD

-20092 **A VIEW WAS SPECIFIED FOR LIKE BUT IT INCLUDES A ROWID COLUMN**

Explanation: The LIKE clause specified the name of a view that contains a ROWID column. This is not supported.

System Action: The statement is not executed.

Programmer Response: Specify the name of a view that does not contain a ROWID column (or distinct type column for which the source type is ROWID), or specify the name of a table and resubmit the statement.

SQLSTATE: 560AE

-20100 **AN ERROR OCCURRED WHEN BINDING A TRIGGERED SQL STATEMENT. INFORMATION RETURNED: SECTION NUMBER :** *section-number* **SQLCODE** *sqlerror*, **SQLSTATE** *sqlstate*, **AND MESSAGE TOKENS** *token-list*

Explanation: During execution of a CREATE TRIGGER statement, the SQL statements specified in the triggered action are bound into a trigger package. During that processing, an error was discovered in one of those statements.

section-number
The section number associated with the failing triggered SQL statement. For triggers that contain a WHEN clause, the WHEN clause is section number one. The triggered SQL statements are numbered sequentially,

beginning with section number two. This is true for triggers with or without a WHEN clause.

sqlcode
The SQLCODE received when binding the statement.

sqlstate
The corresponding SQLSTATE for the SQLCODE received when binding the statement.

token-list
The list of tokens from the original SQL error. This list might be truncated.

System Action: The CREATE TRIGGER statement was not processed. The trigger and the trigger package were not created.

Programmer Response: Use the section number determine the failing triggered SQL statement. Refer to the explanation of the reported SQLCODE. Follow the action suggested by that message.

SQLSTATE: 56059

-20101 **THE FUNCTION** *function* **FAILED WITH REASON** *rc*

Explanation: The statement attempted to execute a function *function*. The statement failed, Reason Code *rc*

Possible values for *rc* are: 00E73001, 00E73002, 00E73003, and 00E73004.

System Action: The statement cannot be executed.

Programmer Response: Correct the condition described by the DB2 reason code.

SQLSTATE: 56060

-20102 **CREATE OR ALTER STATEMENT FOR USER-DEFINED FUNCTION** *function-name* **SPECIFIED THE** *option* **OPTION WHICH IS NOT ALLOWED FOR THE TYPE OF ROUTINE**

Explanation: An option was specified that is not allowed for the type of function being created or altered.
- MODIFIES SQL DATA is not allowed for table functions.
- ALLOW PARALLEL is not allowed for table functions.
- CARDINALITY is not allowed for non-table functions.
- LANGUAGE SQL is not allowed for non-SQL functions or procedures.
- LANGUAGE specifying something other than SQL is not allowed for SQL functions or procedures.
- LANGUAGE JAVA is not allowed for table functions.
- PARAMETER STYLE JAVA is not allowed for table functions.

System Action: The statement cannot be processed.

Programmer Response: Remove the option from the statement and reissue the failing statement.

SQLSTATE: 42849

-20104 **AN ATTEMPT TO ALTER A CCSID FROM** *from-ccsid* **TO** *to-ccsid* **FAILED**

Explanation: The statement attempted to alter the CCSID for a database or table space and the statement failed.

from-ccsid represents the CCSID that is currently in use for the database or table space.

to-ccsid is the CCSID that specified on the alter statement.

System Action: The statement cannot be executed.

Programmer Response: The SQL REFERENCE contains a list of CCSIDs that may be specified on this statement. Only CCSIDs specified in this appendix may be altered, and then, only to the corresponding value listed in the table. Altering the CCSID of a database or table space to a value not listed in the table is not permitted.

SQLSTATE: 42856

-20106 **THE CCSID FOR TABLE SPACE OR DATABASE CANNOT BE CHANGED BECAUSE THE TABLE SPACE OR DATABASE ALREADY CONTAINS A TABLE THAT IS REFERENCED IN EXISTING VIEW DEFINITIONS**

Explanation: An ALTER statement cannot be used to alter the CCSID for a table space or database that contains a table that is referenced in existing view definitions.

System Action: The statement cannot be executed.

Programmer Response: To alter the CCSID for the specified space, first drop any existing view definitions that refer to tables contained in the identified space and then reissue the ALTER statement.

SQLSTATE: 42945

-20107 **HOST VARIABLE OR PARAMETER NUMBER** *position-number* **CANNOT BE USED AS SPECIFIED BECAUSE REASON** *reason*

Explanation: DB2 received data that could not be used as specified in the statement because it is not convertible to an acceptable format in this machine environment.

position-number identifies either the host variable number (if the message is issued as a result of an INSERT, UPDATE, DELETE, SELECT, SET, or VALUES statement), or the parameter number (if the message is

issued as the result of a CALL statement, or the invocation of a function).

reason

01 IEEE (BFP) floating point instructions or instruction emulation is not available. This support is called the basic-floating-point-extensions facility, and is discussed in detail in *ESA/390 Principles of Operation, SA22-7201–05* (and subsequent updates).

System Action: The statement cannot be executed.

Programmer Response: This host variable or parameter requires machine instructions that are not available on this machine. These instructions must be available to DB2 to perform the requested operation. Run this statement on a machine that is capable of supporting the required operations. Contact your system administrator.

SQLSTATE: 53022

-20110 CANNOT IMPLICITLY CONNECT TO A REMOTE SITE WITH A SAVEPOINT OUTSTANDING

Explanation: The statement referenced an object at a remote DBMS with an alias or a three-part name when an active savepoint exists. Such a reference requires an implicit connection to the remote DBMS, which is not allowed when there is an outstanding savepoint.

System Action: The statement is not executed.

Programmer Response: Either release the savepoint, or move the data.

SQLSTATE: 51036

-20111 CANNOT ISSUE SAVEPOINT, RELEASE SAVEPOINT, ROLLBACK TO SAVEPOINT FROM A TRIGGER, FROM A USER-DEFINED FUNCTION, OR FROM A GLOBAL TRANSACTION

Explanation: SAVEPOINT, RELEASE SAVEPOINT, and ROLLBACK TO SAVEPOINT statements can not be used in the body of a trigger or a global transaction.

System Action: The statement is not executed.

Programmer Response: Correct the logic of the application program so that this error does not occur.

SQLSTATE: 3B503

-20123 CALL TO STORED PROCEDURE *procedure* FAILED BECAUSE THE RESULT SET RETURNED FOR CURSOR *cursor* IS SCROLLABLE, BUT THE CURSOR IS NOT POSITIONED BEFORE THE FIRST ROW

Explanation: A scrollable result set for cursor *cursor* has been returned by a CALL to stored procedure *procedure*, and one or more of these cursors is not positioned before the first row.

System Action: The Call to the stored procedure was unsuccessful. All result set cursors defined in the stored procedure were closed before returning to the caller. The scrollable cursor can not be used to FETCH from the result set. Actions completed by the stored procedure are not rolled back, and any external actions initiated by the stored procedure have completed because the error was detected upon completion of the stored procedure.

Programmer Response: Modify the content of the stored procedure to ensure that the result set cursors are positioned before the first row before returning to the caller.

SQLSTATE: 560B1

-20124 OPEN CURSOR *cursor* FAILED BECAUSE THE CURSOR IS SCROLLABLE BUT THE CLIENT DOES NOT SUPPORT THIS

Explanation: The cursor *cursor* has been defined as scrollable, but the client is downlevel and does not support scrollable cursors. The DRDA Application Requestor is not able to process scrollable result set.

Operator Response: The statement cannot be processed.

Programmer Response: Modify the definition of the cursor to not be scrollable.

SQLSTATE: 560B2

-20125 CALL TO STORED PROCEDURE *procedure* FAILED BECAUSE THE RESULT SET FOR CURSOR *cursor* IS SCROLLABLE, BUT THE CLIENT DOES NOT SUPPORT THIS

Explanation: A scrollable result set for cursor *cursor* has been returned by a CALL to stored procedure *procedure*, but the client is downlevel and does not support scrollable cursors. The DRDA Application Requestor is not able to process scrollable result sets.

System Action: The statement cannot be processed. All result set cursors defined in the stored procedure were closed before returning to the caller. The scrollable cursor can not be used to FETCH from the result set. Actions done by the stored procedure are not rolled

back, and any external actions initiated by the stored procedure have completed because the error was detected at the end of the execution of the stored procedure.

Programmer Response: Modify the content of stored procedure *procedure* to not define result set cursors as scrollable.

SQLSTATE: 560B3

-20126 CURSOR *cursor* IS DEFINED AS SCROLLABLE, BUT THE ENVIRONMENT INVOLVES A HOP SITE

Explanation: The OPEN statement for a result set cursor *cursor* failed because it is defined as scrollable and one or more hop sites are involved. Scrollable result sets cursors are not supported with hop sites. If the error occurs on a CALL statement then a cursor in the procedure is defined as scrollable.

System Action: The statement cannot be processed.

Programmer Response: Modify the definition of the cursor such that it is not scrollable.

SQLSTATE: 560B4

-20127 VALUE SPECIFIED ON FETCH STATEMENT FOR ABSOLUTE OR RELATIVE IS TOO LARGE FOR DRDA

Explanation: The value specified after ABSOLUTE or RELATIVE on a FETCH statement is larger that what will fit in 64 bits. DRDA limits this specification to 64 bits.

System Action: The statement cannot be processed.

Programmer Response: Modify the scroll specification on the FETCH statement.

SQLSTATE: 56051

-20129 LOCAL SPECIAL REGISTER IS NOT VALID AS USED

Explanation: If an assignment statement (SET or VALUES INTO) assigns multiple values, it cannot reference the local special registers. For example, the following special registers are local: CURRENT SERVER and CURRENT PACKAGESET.

Additionally, a SET host-variable statement cannot reference the local special registers if it has parentheses around the host variable and around the name of the special variable.

The following statements are not valid:

```
SET (:hv1) = (CURRENT SERVER);

SET (:hv1,:hv2) = (CURRENT SERVER,CURRENT PATH);

VALUES CURRENT SERVER, CURRENT DATE INTO :c1, :c2;
```

Severity: 8 (error)

System Action: DB2 cannot process the statement.

Programmer Response: To reference local special registers for multiple values, use multiple statements to avoid this error. Also, ensure that a SET statement for a local special register does not use parentheses on either side of the equal sign.

SQLSTATE: 560B5

-20201 THE INSTALL, REPLACE OR REMOVE OF *jar-name* FAILED AS THE JAR NAME IS INVALID

Explanation: The jar name *jar-name* specified on the install, replace, or remove jar procedure was invalid. For example, the jar may be of the improper format, may not exist to be replaced, or cannot be installed as it already exists.

System Action: The statement cannot be processed.

User Response: Ensure the jar identifier is of the correct format. If the jar exists, it may need to be removed before it can be installed. For the remove or replace procedures, ensure the jar exists.

SQLSTATE: 46002

-20202 THE REPLACE OR REMOVE OF *jar-name* FAILED AS *class* IS IN USE

Explanation: A defined routine currently is using the specified class in the jar file, or the replacement jar file does not contain the specified class for which a routine is defined.

System Action: The statement cannot be processed.

SQLSTATE: 46003

User Response: Ensure all routines referencing the classes being removed are dropped and resubmit the replace or remove procedure.

-20204 THE USER-DEFINED FUNCTION OR PROCEDURE *routine-name* WAS UNABLE TO MAP TO A SINGLE JAVA METHOD

Explanation: A CREATE or ALTER FUNCTION or PROCEDURE statement for *routine-name* specified a Java method in the EXTERNAL NAME clause that cannot be used. The identified function or procedure either failed to find a matching Java method, or found more than 1 matching Java method.

System Action: The statement cannot be processed.

SQLSTATE: 46008

User Response: If the failing statement was a CREATE or ALTER, then change the EXTERNAL NAME clause to refer to a valid Java method for the routine.

Otherwise, ...tbd... so that the function or procedure call resolves to a single Java method.

-30000 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT WILL NOT AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS: REASON *reason-code (sub-code)*

Explanation: A DRDA protocol error has resulted which prevented successful execution of the current SQL statement. The error was such that it will not preclude the successful execution of further SQL statements.

System Action: The statement cannot be executed. The SQLCA is formatted. Message DSNL031I or DSNL032I, which might contain additional diagnostic information, might be issued to the MVS console.

Programmer Response: Notify the DBA for assistance in analysis of the SQL statement which yielded this SQLCODE.

Problem Determination: The 'reason-code' identifies the DDM code point which represents the DDM reply message received from the remote server in response to the attempt to execute the SQL statement. These represent internal errors detected at the remote server or possibly, by the local DB2 functions.

The 'reason-code' value is the two-byte hexadecimal code point for the DDM reply message that represents the error and is one of the following:

X'1254' - CMDCHKRM
X'220A' - DSCINVRM
X'220E' - DTAMCHRM
X'1245' - PRCCNVRM
X'2202' - QRYNOPRM
X'220F' - QRYPOPRM
X'2207' - RDBACCRM
X'2204' - RDBNACRM
X'124C' - SYNTAXRM

A two-byte 'sub-code' accompanies 'reason-codes' X'220A' (DSCINVRM), X'1245' (PRCCNVRM), and X'124C' (SYNTAXRM). In all other cases, the 'sub-code' is zero.

The 'sub-code' when nonzero, consists of two bytes such that the high-order byte indicates the site at which the error was detected. This is X'01' if the error was detected by the local DB2; it is X'02' if the error was detected by the remote server. The low-order byte is dependent on the 'reason-code' as follows:

1. Description Error Code (DSCERRCD) if 'reason-code' = X'220A' (DSCINVRM).
2. Syntax Error Code (SYNERRCD) if 'reason-code' = X'124C' SYNTAXRM).
3. Conversational Protocol Error Code (PRCCVNCD) if 'reason-code' = X'1245' (PRCCNVRM).

Refer to the *IBM Distributed Data Management (DDM) Reference Guide* for a detailed discussion of the semantics of the DDM terms DSCERRCD, SYNERRCD, and PRCCNVCD.

SQLSTATE: 58008

-30002 THE SQL STATEMENT CANNOT BE EXECUTED DUE TO A PRIOR CONDITION IN A CHAIN OF STATEMENTS

Explanation: An SQL statement was chained to PREPARE, but the PREPARE statement has received a warning SQLCODE that requires the program or end user to either re-issue the chained statement or issue a different SQL request. This error can occur only in a client/server environment.

- A distributed client using DRDA has chained an OPEN statement to a PREPARE, but the PREPARE statement received SQLCODE +495.

System Action: The statement cannot be executed as chained.

Programmer Response: The statement must be sent again as a separate request.

SQLSTATE: 57057

-30020 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT CAUSED DEALLOCATION OF THE CONVERSATION: REASON <*reason-code (sub-code)***>**

Explanation: A DRDA protocol error has occurred that prevented the successful execution of the current SQL statement or command, as well as any subsequent SQL statements.

The 'reason-code' identifies the DDM code point which represents the DDM reply message received from the remote server in response to the attempt to execute the SQL statement. These represent internal errors detected at the remote server site or possibly, by the local DB2 functions.

The 'reason-code' value is the two-byte hexadecimal code point for the DDM reply message that represents the error and is one of the following:

X'1232' - AGNPRMRM
X'1254' - CMDCHKRM
X'220A' - DSCINVRM
X'220E' - DTAMCHRM
X'0010' - FDODSC
X'1218' - MGRDEPRM
X'1245' - PRCCNVRM
X'241A' - QRYDSC
X'2202' - QRYNOPRM

X'220F' - QRYPOPRM
X'2207' - RDBACCRM
X'2204' - RDBNACRM
X'124C' - SYNTAXRM

A two byte sub-code accompanies 'reason codes'. The sub-code, when nonzero, consists of two bytes such that the high-order byte indicates the site at which the error was detected. This is X'01' if the error was detected by the local DB2; It is X'02' if the error was detected by the remote server. The low-order byte is dependent upon the 'reason code' as follows:
Description Error Code (DSCERRCD) if reason code= X'220A' (DSCINVRM).
Syntax Error Code (SYNERRCD) if reason code = X'124C' (SYNTAXRM).
Conversational Protocol Error Code (PRCCVNCD) if reason code = X'1245' (PRCCNVRM).
Manager Dependency Error Code (DEPERRCD) if reason code = X'1218' (MGRDEPRM).

Refer to *IBM Distributed Data Management (DDM) Reference Guide* for a detailed discussion of the semantics of the DDM terms DSCERRCD, SYNERRCD, PRCCNVCD, and DEPERRCD.

System Action: The statement cannot be executed. The SQLCA is formatted and the conversation on which the error was detected is deallocated. Message DSNL031I or DSNL032I, which might contain additional diagnostic information, might be issued to the MVS console.

Programmer Response: The connection to the server has been broken, and the server has, therefore, rolled back the unit of work. In this case, the only SQL statement that may be successfully executed is ROLLBACK. However, if the requester detects this error on a COMMIT, then it is unknown whether the unit of work was committed or rolled back at the server.

SQLSTATE: 58009

-30021 **EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT WILL AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS: MANAGER** *manager* **AT LEVEL** *level* **NOT SUPPORTED ERROR**

Explanation: A DRDA error occurred that prevented the successful execution of the current SQL statement or command and any subsequent SQL statements.

A manager-level conflict was detected during the processing of the DDM EXCSAT command. Refer to the *IBM Distributed Data Management (DDM) Reference Guide* for a detailed description of EXCSAT processing and errors.

The *manager* value is the 2-byte hexadecimal code point of the DDM manager class identified as not supported in the EXCSATRD that reported the error. Refer to the DDM term MGRLVL in the *IBM Distributed Data Management (DDM) Reference Guide* for the 2-byte hexadecimal values.

The *level* value is the 2-byte hexadecimal value of the manager level identified as not supported in the EXCSATRD that reported the error. Refer to the DDM term MGRLVL in the *IBM Distributed Data Management (DDM) Reference Guide* for the 2-byte hexadecimal values.

System Action: The statement cannot be executed.

Programmer Response: Notify the system programmer for analysis of the condition that caused this SQLCODE.

SQLSTATE: 58010

-30030 **COMMIT REQUEST WAS UNSUCCESSFUL, A DISTRIBUTION PROTOCOL VIOLATION HAS BEEN DETECTED, THE CONVERSATION HAS BEEN DEALLOCATED. ORIGINAL SQLCODE=***original-sqlcode* **AND ORIGINAL SQLSTATE=***original-sqlstate*

Explanation: The application requested COMMIT operation was unsuccessful and the response from the remote server and the SQLCODE that was returned from the remote server are inconsistent. For example, the reply message from the remote server indicates that a COMMIT operation did not complete successfully but the SQLCODE returned from the remote server was greater than or equal to zero. The unit of work has been rolled back and the connection with the remote server has been deallocated.

System Action: An alert was generated. A DSNL031I message may have been written to the console. Refer to the description of this message for further information.

The SQLCODE returned by the remote server is replaced with -30030 and the SQLSTATE returned by the AS is replaced with '158013' and the connection with the remote server has been deallocated.

The SQLCODE and SQLSTATE values that were returned from the remote server are stored in the SQLERRM field in a string of the following format:

```
'original-sqlcode 'FF'X original-sqlstate'
```

Programmer Response: Notify the system programmer for assistance in analyzing the trace data that was generated.

SQLSTATE: 58013

-30040 **EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL NOT AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS. REASON** *reason-code* **TYPE OF RESOURCE** *resource-type* **RESOURCE NAME** *resource-name* **PRODUCT ID** *pppvvrrm* **RDBNAME** *rdbname*

Explanation: The SQL statement or command requires a non RDB resource that is currently unavailable. The *reason-code* identifies why the resource identified by the *resource-type* and *resource-name* is unavailable. The product, product level, and RDB are identified.

Values for *reason-code*

This is product specific and can be found in the remote server documentation.

Values for *resource-type*

Two byte hexadecimal number defined as follows:

This is product specific and can be found in the remote server documentation.

Values for *resource-name*

Variable length field with alphanumeric characters and a maximum length of 35 bytes.

Values for *pppvvrrm*

Eight-byte field with alphanumeric characters defined as follows:

ppp..... DSN for OS/390, ARI for SQL/DS SQL and QSQ for AS/400 (3 bytes)

vv..... version number (2 bytes)

rr..... release level (2 bytes)

m..... modification level (1 byte)

Values for *rdbname*

Sixteen-byte field with the RDBNAME.

System Action: The statement cannot be processed. The local DB2 is disconnected from the remote server.

Programmer Response: Verify the identify of the resource that was not available. The reason the resource is unavailable is identified by the *reason-code*.

Collect the following diagnostic items to help determine the cause of the unavailable resource.

- Console output from the system identified by RDBNAME, and a listing of SYSLOG data set for the period of time spanning the failure.
- Information described for the reason code received.

SQLSTATE: 57012

-30041 **EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS AND SQL STATEMENTS. REASON <***reason-code***> TYPE OF RESOURCE <***resource-type***> RESOURCE NAME <***resource-name***> PRODUCT ID <***pppvvrrm***> RDBNAME <***rdbname***>**

Explanation: The SQL statement or command requires a non RDB resource that is currently unavailable. The <reason-code> identifies why the resource identified by the <resource-type> and <resource-name> is unavailable. The product, product level, and RDB are identified.

Values for <reason-code>

Four byte binary number that is product specific. This can be found in the remote server documentation.

Values for <resource-type>

Two byte hexadecimal number that is product specific. This can be found in the remote server documentation.

Values for <resource-name>

Variable length field with alphanumeric characters and a maximum length of 35 bytes.

Values for <pppvvrrm>

Eight byte field with alphanumeric characters defined as follows:

ppp..... DSN for DB2, ARI for SQL/DS, SQL for OS/2 and QSQ for AS/400 (3 bytes)

vv..... version number (2 bytes)

rr..... release level (2 bytes)

m..... modification level (1 byte)

Values for <rdbname>

Sixteen byte field with the RDBNAME.

System Action: The statement cannot be executed. The local DB2 is disconnected from the remote server.

Programmer Response: Verify the identity of the resource that was not available. The reason the resource is unavailable is identified by the reason-code.

Collect the following diagnostic items to help determine the cause of the unavailable resource.

- Console output from the system identified by RDBNAME, and a listing of SYSLOG data set for the period of time spanning the failure.
- Information described for the <reason-code> received.

SQLSTATE: 57013

-30050 **<*command-or-SQL-statement-type* COMMAND OR SQL STATEMENT INVALID WHILE BIND PROCESS IN PROGRESS**

Explanation: A remote command or remote SQL execution was attempted while a remote bind was in progress. The only commands allowed during bind are, BIND STATEMENT, END BIND, ROLLBACK, or COMMIT.

System Action: The request is rejected. The local DB2 is disconnected from remote server.

Programmer Response: Ensure the remote bind has completed before attempting to execute an SQL statement or process a remote command. COMMIT and ROLLBACK will terminate the bind processing.

SQLSTATE: 58011

-30051 **BIND PROCESS WITH SPECIFIED PACKAGE NAME AND CONSISTENCY TOKEN NOT ACTIVE**

Explanation: Binding of a statement or end bind was attempted while the package was not undergoing bind processing.

System Action: The bind statement request or end bind request is rejected. The local DB2 is disconnected from the remote server.

Programmer Response: Ensure that the server processing the bind request was not abending when this request was being processed, or that the package name hasn't changed before terminating the remote bind package request.

SQLSTATE: 58012

-30052 **PROGRAM PREPARATION ASSUMPTIONS ARE INCORRECT**

Explanation: The program preparation assumptions in effect for binding a statement (i.e. BINDSQLSTT command) are incorrect.

System Action: The statement is rejected.

Programmer Response: The creation of the package following this error is dependent on the package creation options specified at begin bind time.

SQLSTATE: 42932

-30053 **OWNER AUTHORIZATION FAILURE**

Explanation: An authorization error associated with the package owner is detected. The AS, for example, has determined that the binder has specified a package owner that the binder has no authorization to specify.

System Action: The begin bind request is rejected.

Programmer Response: Correct the authorization problem and reissue the request.

SQLSTATE: 42506

-30060 **RDB AUTHORIZATION FAILURE**

Explanation: The user is not authorized to access an RDB.

System Action: The request is rejected.

Programmer Response: Correct the authorization problem and resubmit the job.

SQLSTATE: 08004

-30061 **RDB NOT FOUND**

Explanation: An attempt was made to access an RDB which cannot be found. This usually means that a requester specified the server location name incorrectly. The server responded by indicating that the requester's specification of the server location name is incorrect. Requester or server changes are needed to make the location names consistent. Location names for DB2 systems are defined in the BSDS DDF record and are also in the DSNL004I console message when DDF is started. For more information about defining location names, see Part 3 of *DB2 Installation Guide*.

System Action: The request is not processed.

Programmer Response: Ensure the RDB name was correctly specified and resubmit the job.

SQLSTATE: 08004

-30070 ***command* COMMAND NOT SUPPORTED ERROR**

Explanation: The target does not support a particular command. The error causes termination of the connection between the local DB2 and the remote server.

Values for *command*

Two byte hexadecimal DDM code point.

System Action: The command is rejected. The local DB2 is disconnected from the remote server.

Programmer Response: Ensure the proper command was issued.

SQLSTATE: 58014

-30071 ***object-type* OBJECT NOT SUPPORTED ERROR**

Explanation: The target does not support a particular object. The error causes the termination of the connection between the local DB2 and the remote server.

Values for *object-type*

Two byte hexadecimal DDM code point.

System Action: The command/SQL statement is rejected. The local DB2 is disconnected from the remote server.

Programmer Response: Ensure the object has been correctly specified and resubmit the job or reissue the command.

SQLSTATE: 58015

-30072 *parameter subcode* **PARAMETER NOT SUPPORTED ERROR**

Explanation: A particular parameter is not supported by either the application requester or the remote server.

Values for *parameter*

Two byte hexadecimal DDM code point.

Values for *subcode*

Two byte hexadecimal number defined as follows: This subcode is composed of two distinct parts, but may optionally be zero. The high-order byte (when not zero) indicates the site at which the error was detected. This is X'01' if the error was detected by the local DB2; it is X'02' if the error was detected by the remote server. The low order byte is always zero.

System Action: The command/SQL statement is rejected. A disconnect has occurred.

Programmer Response: The connection to the server has been broken, and the server has therefore rolled back the unit of work. In this case, the only SQL statement that may be successfully executed is ROLLBACK. However, if the requester detects this error on a COMMIT, then it is unknown whether the unit of work was committed or rolled back at the server.

SQLSTATE: 58016

-30073 *parameter subcode* **PARAMETER VALUE NOT SUPPORTED ERROR**

Explanation: The specified parameter value is not supported by either the local DB2 or the remote server.

The *parameter*, is a 2-byte hexadecimal DDM code point. See *IBM Distributed Data Management (DDM) Reference Guide* for a definition of the valid code points. Common values for *parameter* include:

X'0035' Usually means that a DB2 server does not support the single-byte coded character set identifier (CCSID) sent by the requester, or a DB2 requester does not support the single-byte CCSID received from a server. See X'119C' for more information.

X'1144' DRDA servers return this code point when a DB2 client specifies a version for a package and the DRDA server does not support versions.

X'119C'

If the subcode information contains X'01', the requester does not support the single-byte CCSID that the server wants to use. If the subcode information contains X'02', the server does not support the single-byte CCSID that the requester wants to use.

Determine the single-byte CCSIDs for the requester and server systems. The single-byte CCSID for DB2 subsystems is defined in DSNHDECP. One or both partners might be using an incorrect CCSID, or support might need to be added. See *DB2 Installation Guide* for more information on CCSIDs.

X'147A'

DB2 servers can return this code point when a numeric input host variable is not within the range that DB2 supports. See the explanation of SQLCODE -406 for more information.

X'14AC'

DB2 servers return this code point when a DRDA client requests an authentication mechanism that is not supported by the DB2 server. See the explanation of DB2 reason code 00D3010E for more information.

X'2110'.

Usually means that a requester specified the server location name incorrectly. The server responded by indicating that the requester's specification of the server location name is incorrect, and the length of the location name is longer than that supported by the server. Requester or server changes are needed so the location names are consistent. Location names for DB2 systems are defined in the BSDS DDF record and can also be seen in the DSNL004I console message when DDF is started. See Part 3 of *DB2 Installation Guide* for more information about defining location names.

X'2120' Usually means that the server does not support the use of double quote string delimiters in SQL statements. DB2 COBOL applications can use double quote string delimiters in SQL statements. This requires that the program be precompiled with the SQLDELIM(QUOTESQL) option. If the server product does not support double quote SQL statement string delimiters, then the program must be modified to use single quote SQL string delimiters and be precompiled with the SQLDELIM(APOSTSQL) option.

X'2121' Usually means that the there is a semantic error with the string that specifies the character used as a decimal delimiter in SQL statements.

X'213F'

Means that the server was processing a bind

or rebind command that specified an unsupported value for the DYNAMICRULES bind parameter. If the server is a DB2 Version 3 system, specify DYNAMICRULES(RUN) or omit the DYNAMICRULES parameter.

The 'subcode', which is a 2-byte hexadecimal number, consists of two distinct parts and can optionally be zero. The high-order byte (when not zero) indicates the site at which the error was detected. If the error was detected by the local DB2, it is X'01'. If the error was detected by the remote server, it is X'02'. The low-order byte is always zero.

System Action: The command or SQL statement is rejected. A disconnect occurred.

Programmer Response: The connection to the server was broken, and the server rolled back the unit of work. The only SQL statement that can be successfully executed is ROLLBACK. However, if the requester detects this error on a COMMIT, then it is unknown whether the unit of work was committed or rolled back at the server.

SQLSTATE: 58017

-30074 **REPLY MESSAGE WITH** *codepoint (svrcod)* **NOT SUPPORTED ERROR**

Explanation: A reply message *codepoint* is not recognized or the reply message *SVRCOD* is not recognized. The error does not affect the processing of subsequent DRDA commands and SQL statements issued by the application program.

Values for *codepoint*

- The *codepoint* is a two-byte hexadecimal value that represents a DDM reply message. *IBM Distributed Data Management (DDM) Reference Guide* defines the valid *codepoints* for reply messages.

Values for *SVRCOD*

- The *SVRCOD* is a one-byte hexadecimal value which represents the reply messages severity code. *IBM Distributed Data Management (DDM) Reference Guide* defines the valid *SVRCOD* for reply messages.

System Action: Processing continues.

Programmer Response: The cause of this error may be a mismatch in source and targe manager levels or might be an internal error.

SQLSTATE: 58018

-30080 **COMMUNICATION ERROR** *code (subcode)*

Explanation: A SNA communications error was detected. *VTAM for MVS/ESA Programming for LU 6.2* contains the valid *code* and *subcode* values that can appear in this message.

- *code* is VTAM's primary LU6.2 return code (RCPRI).
- *subcode* takes one of the following forms:
  - rcsec-sense
  - rcsec

  The *rcsec* portion of *subcode* is VTAM's secondary LU6.2 return code (RCSEC).

  The *sense* portion of *subcode* is VTAM's sense code. See "Part 6. Section 6. SNA CODES" on page 1279 for descriptions of sense codes that are often associated with SNA network definition errors.

System Action: The statement is not executed. The application was disconnected from the server because of a communication failure. A DSNL500I message containing additional diagnostic information might be issued to the MVS console.

Programmer Response: Review the diagnostic information described in *VTAM for MVS/ESA Programming for LU 6.2* for the particular LU6.2 return codes. Consult with a communications expert to determine the cause of the communication failure.

SQLSTATE: 08001

-30081 *prot* **COMMUNICATION ERROR DETECTED. API=***api***, LOCATION=***loc***, FUNCTION=***func***, ERROR CODES=***rc1 rc2 rc3*

Explanation: A communications error was detected while communicating with a remote client or server.

prot Identifies the communication protocol that encountered the error. Currently, 'TCP/IP' is the only possible value.

api Identifies the communication application programming interface (API) used by DB2. Currently, 'SOCKETS' is the only possible value.

loc The network location of the partner system.

For TCP/IP partners, this field contains the dotted decimal IP address of the partner.

func The communication function that failed.

For TCP/IP partners, this field identifies name of the socket call that failed.

rc1 The first return code indicator.

For TCP/IP partners, this field contains the TCP/IP *return code* in decimal format. The *return code* values are documented in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

rc2 The second return code indicator.

For TCP/IP partners, this field contains the TCP/IP *reason code* in decimal format. The

reason code values are documented in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

rc3 The third return code indicator.

For TCP/IP partners, this field contains zero.

System Action: The statement is not executed. The application is disconnected from the server.

Programmer Response: Consult with a communications expert to determine the cause of the communication failure.

SQLSTATE: 08001

-30082 CONNECTION FAILED FOR SECURITY REASON *reason-code* **(***reason-string***)**

Explanation: The attempt to connect to a remote database server was rejected due to invalid or incorrect security information. The cause of the security error is described by the *reason-code* and *reason-string* values.

The possible values for *reason-code* and *reason-string* appear below:

- 1 (PASSWORD EXPIRED) -- the password used to connect to the remote server has expired.
- 2 (PASSWORD INVALID) -- the password used to connect to the remote server does not match the password stored at the remote server.
- 3 (PASSWORD MISSING) -- the remote server rejected the connection request because the request did not include a password.
- 4 (PROTOCOL VIOLATION) -- the remote server rejected the connection request because the request did not contain the proper security information. Error messages or trace records should be produced by the server system to explain the nature of the security violation.
- 5 (USER-ID MISSING) -- the remote server rejected the connection request because the request did not specify a user-id.
- 6 (USER-ID INVALID) -- the user-id specified in the connection request is not defined at the remote server system.
- 7 (USER-ID REVOKED) -- the user-id specified in the connection request has been revoked.
- 15 (SECURITY FAILURE:*secchkcd:svcerrno*)—Authentication failed at the remote server system. Refer to the *IBM Distributed Data Management (DDM) Reference Guide* for a detailed discussion of the semantics of the DDM terms SECCHKCD and SVCERRNO.
- 16 (NEW PASSWORD INVALID) -- the password specified on a change password request did not meet the server's requirements.
- 17 (UNSUPPORTED FUNCTION) -- the security mechanism specified by the client is invalid for this server. Some typical examples:
  - The client sent a new password value to a server that does not support the DRDA change password function.
  - The client sent DCE authentication information to a server that does not support DCE.
  - The client is configured to send an Already Verified userid but the server does not support Already Verified userids. Client or server changes must be made to correct the inconsistency.
    - DB2 for OS/390 server acceptance of Already Verified userids over TCP/IP connections is controlled by the TCPALVER value of the DSNTIP5 installation panal (DSN6FAC TCPALVER).
    - DB2 for OS/390 client usage of Already Verified userids is controlled by the SECURITY_OUT column of the SYSIBM.LUNAMES or SYSIBM.IPNAMES tables.

System Action: The attempt to connect to the remote database server fails. If the server system is a DB2 for OS/390 and z/OS server, a DSNL030I message at the server system describes the cause.

Programmer Response: DB2 uses the communications database (CDB) to control network security functions. Make the appropriate changes to the CDB to correct the security failure.

SQLSTATE: 08001

-30090 REMOTE OPERATION INVALID FOR APPLICATION EXECUTION ENVIRONMENT

Explanation: An update operation or a dynamic commit or rollback was attempted at a server that was supporting an application that was in a read-only execution environment (IMS or CICS).

System Action: The request is rejected.

Programmer Response: Do not attempt to update data or issue dynamic commits or rollbacks from IMS or CICS applications that are accessing remote data.

SQLSTATE: 25000

-30104 ERROR IN BIND OPTION *option* **AND BIND VALUE** *value***.**

Explanation: While processing the bind parameters that were passed from the requester site, either the bind option or the bind value is not acceptable to the server database, or the option/value pair is not appropriate.

Programmer Response: The BIND or REBIND command failed. Examine the command options and values, determine the error, and resubmit the command.

SQLSTATE: 56095

-30105 BIND OPTION *option1* **IS NOT ALLOWED WITH BIND OPTION** *option2*

Explanation: While processing the bind parameters that were passed from the requester site, it was found that there is a conflict in bind options that are mutually exclusive.

Programmer Response: The BIND or REBIND command failed. Examine the command options, determine the cause of the conflict, and resubmit the command.

SQLSTATE: 56096