INTERNATIONAL STANDARD ISO/IEC 9075-3:1999
TECHNICAL CORRIGENDUM 2

Published 2003-06-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Database languages — SQL —

Part 3:
Call-Level Interface (SQL/CLI)

TECHNICAL CORRIGENDUM 2

Technologies de l'information — Langages de base de données — SQL —

Partie 3: Interface de niveau d'appel (SQL/CLI)

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 9075-3:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*. ISO/IEC 9075-3:1999/Cor. 2:2003 cancels and replaces ISO/IEC 9075-3:1999/Cor. 1:2000.

Statement of purpose for rationale:

A statement indicating the rationale for each change to ISO/IEC 9075 is included. This is to inform the users of that standard as to the reason why it was judged necessary to change the original wording. In many cases the reason is editorial or to clarify the wording; in some cases it is to correct an error or an omission in the original wording.

Notes on numbering:

Where this Corrigendum introduces new Syntax, Access, General and Conformance Rules, the new rules have been numbered as follows:
Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7) a.1), 7) a.2), etc.]. Those inserted before Rule 1) are numbered 0.1), 0.2), etc.
Where this Corrigendum introduces new Subclauses, the new subclauses have been numbered as follows:
Subclauses inserted between, for example, Subclause 4.3.2 and 4.3.3 are numbered 4.3.2a, 4.3.2b, etc.
Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

ICS 35.060 **Ref. No. ISO/IEC 9075-3:1999/Cor.2:2003(E)**

Published in Switzerland

Contents

Page

Information technology — Database languages — SQL —

Part 3:
Call-Level Interface (SQL/CLI)

TECHNICAL CORRIGENDUM 2

3.1.1.Definitions provided in Part 3

1. Rationale: Provide missing definition

Insert the following item:

a.0) **data source**: A synonym for the SQL-server that is part of the current connection.

3.3.2.1 Clause, subclause and table relationships

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace the following rows from Table 1 "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of /ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 5.15, "Data type correspondences"	Subclause 13.6, "Data type correspondences"	ISO/IEC 9075-2
Table 44, "Data type correspondences for Ada"	Table 18, "Data type correspondences for Ada"	ISO/IEC 9075-2
Table 45, "Data type correspondences for C"	Table 18, "Data type correspondences for C"	ISO/IEC 9075-2
Table 46, "Data type correspondences for COBOL"	Table 18, "Data type correspondences for COBOL"	ISO/IEC 9075-2
Table 47, "Data type correspondences for Fortran"	Table 18, "Data type correspondences for Fortran"	ISO/IEC 9075-2
Table 48, "Data type correspondences for MUMPS"	Table 18, "Data type correspondences for MUMPS"	ISO/IEC 9075-2
Table 49, "Data type correspondences for Pascal"	Table 18, "Data type correspondences for Pascal"	ISO/IEC 9075-2
Table 50, "Data type correspondences for PL/I"	Table 18, "Data type correspondences for PL/I"	ISO/IEC 9075-2

with the following rows:

Clause, Subclause, or Table in this part of /ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 5.15, "SQL/CLI data type correspondences"	*none*	*none*
Table 44, "SQL/CLI data type correspondences for Ada"	*none*	*none*

Table 45, "SQL/CLI data type correspondences for C"	*none*	*none*
Table 46, "SQL/CLI data type correspondences for COBOL"	*none*	*none*
Table 47, "SQL/CLI data type correspondences for Fortran"	*none*	*none*
Table 48, "SQL/CLI data type correspondences for MUMPS"	*none*	*none*
Table 49, "SQL/CLI data type correspondences for Pascal"	*none*	*none*
Table 50, "SQL/CLI data type correspondences for PL/I"	*none*	*none*

4.4.5 Connection attributes

1. *Rationale: Remove the anomaly in <savepoint specifier> and correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace the 5th paragraph with:

> The SAVEPOINT NAME connection attribute specifies the savepoint to be referenced in an invocation of the EndTran routine that uses the SAVEPOINT NAME ROLLBACK or SAVEPOINT NAME RELEASE CompletionType. The SAVEPOINT NAME attribute is set to a zero-length string when the SQL-connection is allocated.

5.1 <CLI routine>

1. *Rationale: Add missing rule.*

Insert the following Syntax Rule:

> 8.1) There shall be no <separator> between the <CLI name prefix> and the <CLI generic name>

2. *Rationale: Separating data type correspondence tables in Foundation and CLI.*

Replace Syntax Rule 12) with:

> 12) Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "SQL data type column" and the "host data type column".

5.3 Description of CLI item descriptor areas

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 6) with:

6) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the *SQL data type column* and the *host data type column*.

5.5 Implicit DESCRIBE USING clause

1. Rationale: Editorial - typographic error.

Replace General Rule 5) c) iv) 1) with:

5) c) iv) 1) If TYPE indicates a <character string type>, then LENGTH is set to the length or maximum length in characters of the character string. OCTET_LENGTH is set to the maximum possible length in octets of the character string. If *HL* is C, then the lengths specified in LENGTH and OCTET_LENGTH do not include the implementation-defined null character that terminates a C character string. CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are set to the <character set name> of the character string's character set. COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are set to the <collation name> of the character string's collation.

5.11 Deferred parameter check

1. Rationale: Editorial - style error.

Replace General Rule 4) with:

4) Let *L2* be the set of all allocated SQL-statements in *L1* that have an associated deferred parameter number.

5.13 Description of CLI item descriptor areas

1. Rationale: Remove mangled text

Replace Syntax Rule 5) c) xiv) with:

5) c) xiv) TYPE indicates ARRAY or ARRAY LOCATOR, the value of CARDINALITY is a valid value for the cardinality of an array, there is exactly one immediately subordinate descriptor area of IDA, and that item descriptor area is valid.

2. Rationale: Editorial - misplaced row.

Delete the following row from Table 6 — Fields in SQL/CLI row and parameter descriptor areas

Field	Data Type
Fields in item descriptor areas	
COUNT	SMALLINT

Insert the following row into Table 6 — Fields in SQL/CLI row and parameter descriptor areas

Field	Data Type
Header fields	
COUNT	SMALLINT

5.14 Other tables associated with CLI

1. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

In Table 14 — Codes used for transaction termination, replace the row:

Termination type	Code
SAVEPOINT NAME COMMIT	2

with:

Termination type	Code
SAVEPOINT NAME ROLLBACK	2

2. Rationale: Remove the anomaly in <savepoint specifier>.

In Table 14 — Codes used for transaction termination, delete the following rows:

Termination type	Code
SAVEPOINT NUMBER COMMIT	3
SAVEPOINT NUMBER RELEASE	5

In Table 16 — Codes used for connection attributes, delete the following row:

Attribute	Code	May be set
SAVEPOINT NUMBER	10028	Yes

In Table 19 — Data types of attributes, delete the following row:

Attribute	Data type	Values
Statement attributes		
SAVEPOINT NUMBER	INTEGER	Not specified

3. *Rationale: Correct the contents of the Type column of Table 20 "Codes used for descriptor fields".*

In Table 20 — Codes used for descriptor fields, delete the following rows:

Field	Code	SQL Item Descriptor Name	Type
CARDINALITY	1040	CARDINALITY	Status
CURRENT_TRANSFORM_GROUP	1039	(Not applicable)	Status
DEGREE	1041	DEGREE	Status
RETURNED_CARDINALITY_ POINTER	1043	RETURNED_CARDINALITY	Status
SCOPE_CATALOG	1033	SCOPE_CATALOG	Status
SCOPE_NAME	1035	SCOPE_NAME	Status
SCOPE_SCHEMA	1034	SCOPE_SCHEMA	Status
SPECIFIC_TYPE_CATALOG	1036	(Not applicable)	Status
SPECIFIC_TYPE_NAME	1038	(Not applicable)	Status
SPECIFIC_TYPE_SCHEMA	1037	(Not applicable)	Status

In Table 20 — Codes used for descriptor fields, add the following rows:

Field	Code	SQL Item Descriptor Name	Type
CARDINALITY	1040	CARDINALITY	Item
CURRENT_TRANSFORM_GROUP	1039	(Not applicable)	Item
DEGREE	1041	DEGREE	Item
RETURNED_CARDINALITY_ POINTER	1043	RETURNED_CARDINALITY	Item
SCOPE_CATALOG	1033	SCOPE_CATALOG	Item
SCOPE_NAME	1035	SCOPE_NAME	Item
SCOPE_SCHEMA	1034	SCOPE_SCHEMA	Item
SPECIFIC_TYPE_CATALOG	1036	(Not applicable)	Item

SPECIFIC_TYPE_NAME	1038	(Not applicable)	Item
SPECIFIC_TYPE_SCHEMA	1037	(Not applicable)	Item

4. *Rationale: Add the missing codes.*

In Table 37 — Codes used for concise data types, add the following rows:

Date Type	Code
USER-DEFINED TYPE	17
ROW	19
ARRAY	50

5.15 Data type correspondences

1. *Rationale: Separating data type correspondence tables in Foundation and CLI.*

Change the name of this subclause to "SQL/CLI data type correspondences".

2. *Rationale: Separating data type correspondence tables in Foundation and CLI.*

In the Function replace the 1st paragraph with:

Replace first paragraph Specify the SQL/CLI data type correspondences for SQL data types and host language types associated with the required parameter mechanisms, as shown in Table 3, "Supported calling conventions of SQL/CLI routines by language".

3. *Rationale: Separating data type correspondence tables in Foundation and CLI.*

In the Tables section:

Replace the title of Table 44 — "Data type correspondences for Ada" with:
Table 44 — "SQL/CLI data type correspondences for Ada".

Replace the title of Table 45 — "Data type correspondences for C" with:
Table 45 — "SQL/CLI data type correspondences for C".

Replace the title of Table 46 — "Data type correspondences for COBOL" with:
Table 46 — "SQL/CLI data type correspondences for COBOL".

Replace the title of Table 47 — "Data type correspondences for Fortran" with:
Table 47 — "SQL/CLI data type correspondences for Fortran".

Replace the title of Table 48 — "Data type correspondences for MUMPS" with:
Table 48 — "SQL/CLI data type correspondences for MUMPS".

Replace the title of Table 49 — "Data type correspondences for Pascal" with:
Table 49 — "SQL/CLI data type correspondences for Pascal".

Replace the title of Table 50 — "Data type correspondences for PL/I" with:
Table 50 — "SQL/CLI data type correspondences for PL/I".

6.5 BindCol

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 8) with:

8) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.6 BindParameter

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 9) with:

9) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.9 ColAttribute

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
ColAttribute (
     StatementHandle      IN  INTEGER,
     ColumnNumber         IN  SMALLINT,
     FieldIdentifier      IN  SMALLINT,
     CharacterAttribute   OUT CHARACTER(L),
     BufferLength         IN  SMALLINT,
     StringLength         OUT SMALLINT,
     NumericAttribute     OUT INTEGER )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.10 ColumnPrivileges

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
ColumnPrivileges (
      StatementHandle      IN  INTEGER,
      CatalogName          IN  CHARACTER(L1),
      NameLength1          IN  SMALLINT,
      SchemaName           IN  CHARACTER(L2),
      NameLength2          IN  SMALLINT,
      TableName            IN  CHARACTER(L3),
      NameLength3          IN  SMALLINT,
      ColumnName           IN  CHARACTER(L4),
      NameLength4          IN  SMALLINT )
      RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, and *L4* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.11 Columns

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
Columns (
      StatementHandle      IN  INTEGER,
      CatalogName          IN  CHARACTER(L1),
      NameLength1          IN  SMALLINT,
      SchemaName           IN  CHARACTER(L2),
      NameLength2          IN  SMALLINT,
      TableName            IN  CHARACTER(L3),
      NameLength3          IN  SMALLINT,
      ColumnName           IN  CHARACTER(L4),
      NameLength4          IN  SMALLINT )
      RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, and *L4* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.12 Connect

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
Connect (
     ConnectionHandle     IN  INTEGER,
     ServerName           IN  CHARACTER(L1),
     NameLength1          IN  SMALLINT,
     UserName             IN  CHARACTER(L2),
     NameLength2          IN  SMALLINT,
     Authentication       IN  CHARACTER(L3),
     NameLength3          IN  SMALLINT )
     RETURNS SMALLINT
```

where

:

— *L1* has a maximum value of 128.

— *L2* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

— *L3* has an implementation-defined maximum value.

6.14 DataSources

1. Rationale: Correct definition of parameter length.

Replace the Definition with:

```
DataSources (
     EnvironmentHandle    IN  INTEGER,
     Direction            IN  SMALLINT,
     ServerName           OUT CHARACTER(L1),
     BufferLength1        IN  SMALLINT,
     NameLength1          OUT SMALLINT,
     Description          OUT CHARACTER(L2),
     BufferLength2        IN  SMALLINT,
     NameLength2          OUT SMALLINT )
     RETURNS SMALLINT
```

where *L1* and *L2* have maximum values equal to the implementation-defined maximum length of a variable-length character string.

6.15 DescribeCol

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
DescribeCol (
     StatementHandle       IN  INTEGER,
     ColumnNumber          IN  SMALLINT,
     ColumnName            OUT CHARACTER(L),
     BufferLength          IN  SMALLINT,
     NameLength            OUT SMALLINT,
     DataType              OUT SMALLINT,
     ColumnSize            OUT INTEGER,
     DecimalDigits         OUT SMALLINT,
     Nullable              OUT SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.17 EndTran

1. Rationale: Remove the anomaly in <savepoint specifier> and correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace General Rule 13) with:

13) If *CT* indicates SAVEPOINT NAME RELEASE, then:

a) If *HT* is not CONNECTION HANDLE, then an exception condition is raised: *CLI-specific condition — invalid handle*.

b) Let *SP* be the value of the SAVEPOINT NAME connection attribute of *C*.

c) If *SP* does not specify a savepoint established within the current SQL-transaction, then an exception condition is raised: *savepoint exception — invalid specification.*

d) The savepoint identified by *SP* and all savepoints established by the current SQL-transaction subsequent to the establishment of *SP* are destroyed.

2. Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.

Replace General Rule 14) d) with:

14) d) Every valid locator value is marked invalid.

3. *Rationale: Remove the anomaly in <savepoint specifier> and correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace General Rule 15) with:

15) If *CT* indicates SAVEPOINT NAME ROLLBACK, then:

a) If *HT* is not CONNECTION HANDLE, then an exception condition is raised: *CLI-specific condition — invalid handle*.

b) Let *SP* be the value of the SAVEPOINT NAME connection attribute of *C*.

c) If *SP* does not specify a savepoint established within the current SQL-transaction, then an exception condition is raised: *savepoint exception — invalid specification*.

d) If an atomic execution context is active and *SP* specifies a savepoint established before the beginning of the most recent atomic execution context, then an exception condition is raised: *savepoint exception — invalid specification*.

e) Any changes to SQL-data or schemas that were made by the current SQL-transaction subsequent to the establishment of *SP* are cancelled.

f) All savepoints established by the current SQL-transaction subsequent to the establishment of *SP* are destroyed.

g) Every valid locator generated in the current SQL-transaction subsequent to the establishment of *SP* is marked invalid.

h) For every open cursor *OC* in *L3* that was opened subsequent to the establishment of *SP*:

i) Let *S* be the allocated SQL-statement with which *OC* is associated.

ii) *OC* is placed in the closed state and its copy of the select source is destroyed.

iii) Any fetched row associated with *OC* is removed from association with *S*.

i) The status of any open cursors in *L3* that were opened by the current SQL-transaction before the establishment of *SP* is implementation-defined.

NOTE 26 — The current SQL-transaction is not terminated, and there is no other effect on the SQL-data or schemas.

6.18 Error

1. *Rationale: Correct definition of parameter length*

Replace the Definition with:

```
Error (
     EnvironmentHandle     IN  INTEGER,
     ConnectionHandle      IN  INTEGER,
     StatementHandle       IN  INTEGER,
```

```
      Sqlstate              OUT CHARACTER(5),
      NativeError           OUT INTEGER,
      MessageText           OUT CHARACTER(L),
      BufferLength          IN  SMALLINT,
      TextLength            OUT SMALLINT )
      RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.19 ExecDirect

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
ExecDirect (
      StatementHandle       IN  INTEGER,
      StatementText         IN  CHARACTER(L),
      TextLength            IN  INTEGER )
      RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.21 Fetch

1. Rationale: Editorial.

Replace General Rule 13) with:

13) Let *RS* be the number of rows in *NR*. For *RN* ranging from 1 (one) to *RS*:

a) Let *RNR* be the *RN*-th row of *NR*. Let *ROWS_PROCESSED* be 0 (zero).

Case:

i) If an exception condition is raised during derivation of any <derived column> associated with *RNR* and *ASP* is not a null pointer, then set the *RN*-th element of *ASP* to 5 (indicating **Row error**). For all diagnostic records that result from the application of this rule, the ROW_NUMBER field is set to RN and the COLUMN_NUMBER field is set to the appropriate column number, if any.

ii) Otherwise, the row *RNR* is fetched and *ROWS_PROCESSED* is incremented by 1 (one).

6.22 FetchScroll

1. Rationale: Editorial.

Replace General Rule 18) ii) with:

18) Let *RS* be the number of rows in *NR*. For *RN* ranging from 1 (one) to *RS*:

a) Let *R* be the *RN*-th row of *NR*. Let *ROWS_PROCESSED* be 0 (zero).

Case:

i) If an exception condition is raised during derivation of any <derived column> associated with *R* and *ASP* is not a null pointer, then set the *RN*-th element of *ASP* to 5 (indicating **Row error**). For all diagnostic records that result from the application of this Rule, the ROW_NUMBER field is set to *RN* and the COLUMN_NUMBER field is set to the appropriate column number, if any.

ii) Otherwise the row *R* is fetched and *ROWS_PROCESSED* is incremented by 1 (one).

6.23 ForeignKeys

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
ForeignKeys (
      StatementHandle      IN  INTEGER,
      PKCatalogName        IN  CHARACTER(L1),
      NameLength1          IN  SMALLINT,
      PKSchemaName         IN  CHARACTER(L2),
      NameLength2          IN  SMALLINT,
      PKTableName          IN  CHARACTER(L3),
      NameLength3          IN  SMALLINT,
      FKCatalogName        IN  CHARACTER(L4),
      NameLength4          IN  SMALLINT,
      FKSchemaName         IN  CHARACTER(L5),
      NameLength5          IN  SMALLINT,
      FKTableName          IN  CHARACTER(L6),
      NameLength6          IN  SMALLINT )
      RETURNS SMALLINT
```

where each of *L1*, *L2*, *L3*, *L4*, *L5*, and *L6* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.28 GetConnectAttr

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete General Rule 6)

6.29 GetCursorName

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetCursorName (
     StatementHandle      IN  INTEGER,
     CursorName           OUT CHARACTER(L),
     BufferLength         IN  SMALLINT,
     NameLength           OUT SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.30 GetData

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 17) with:

17) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.32 GetDescRec

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetDescRec (
     DescriptorHandle     IN  INTEGER,
     RecordNumber         IN  SMALLINT,
     Name                 OUT CHARACTER(L),
     BufferLength         IN  SMALLINT,
     NameLength           OUT SMALLINT,
     Type                 OUT SMALLINT,
     SubType              OUT SMALLINT,
     Length               OUT INTEGER,
     Precision            OUT SMALLINT,
     Scale                OUT SMALLINT,
     Nullable             OUT SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.34 GetDiagRec

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetDiagRec (
      HandleType           IN  SMALLINT,
      Handle               IN  INTEGER,
      RecordNumber         IN  SMALLINT,
      Sqlstate             OUT CHARACTER(5),
      NativeError          OUT INTEGER,
      MessageText          OUT CHARACTER(L),
      BufferLength         IN  SMALLINT,
      TextLength           OUT SMALLINT )
      RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.36 GetFeatureInfo

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
GetFeatureInfo (
      ConnectionHandle     IN  INTEGER,
      FeatureType          IN  CHARACTER(L1),
      FeatureTypeLength    IN  SMALLINT,
      FeatureId            IN  CHARACTER(L2),
      FeatureIdLength      IN  SMALLINT,
      SubFeatureId         IN  CHARACTER(L3),
      SubFeatureIdLength   IN  SMALLINT,
      Supported            OUT SMALLINT )
      RETURNS SMALLINT
```

where each of *L1*, *L2*, and *L3* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.40 GetParamData

1. Rationale: Separating data type correspondence tables in Foundation and CLI.

Replace General Rule 17) with:

17) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.15, "SQL/CLI data type correspondences". Refer to the two columns of the *operative data type correspondence table* as the *SQL data type column* and the *host data type column*.

6.45 GetTypeInfo

Replace General Rule 8) h) with:

8) h) The value of CASE_SENSITIVE is 1 (one) if the data type is a character string type and the default collation for its implementation-defined implicit character set would result in a case sensitive comparison when two values with this data type are compared. Otherwise, the value of CASE_SENSITIVE is 0 (zero).

6.50 Prepare

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
Prepare (
     StatementHandle       IN  INTEGER,
     StatementText         IN  CHARACTER(L),
     TextLength            IN  INTEGER )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.51 PrimaryKeys

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
PrimaryKeys (
     StatementHandle    IN  INTEGER,
     CatalogName        IN  CHARACTER(L1),
     NameLength1        IN  SMALLINT,
     SchemaName         IN  CHARACTER(L2),
     NameLength2        IN  SMALLINT,
     TableName          IN  CHARACTER(L3),
     NameLength3        IN  SMALLINT )
     RETURNS SMALLINT
```

where each of *L1*, *L2*, and *L3* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.54 SetConnectAttr

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete General Rule 5)

6.55 SetCursorName

1. Rationale: Correct definition of parameter length

Replace the Definition with:

```
SetCursorName (
     StatementHandle      IN  INTEGER,
     CursorName           IN  CHARACTER(L),
     NameLength           IN  SMALLINT )
     RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

6.56 SetDescField

1. Rationale: Editorial - typographic error.

Replace General Rule 13) i) iii) 2) with:

13) i) iii) 2) Otherwise, let *FV* be the first *L* octets of Value and let *TFV* be the value of

```
TRIM ( BOTH ' ' FROM FV )
```

2. Rationale: Editorial - typographic error.

Replace General Rule 14) a) with:

14) a) If *RI* is 1 (one) and Value is not 0 (zero), then an exception condition is raised: *dynamic SQL error — invalid LEVEL value.*

3. Rationale: Editorial - typographic error.

Replace General Rule 14) b) i) with:

14) b) i) If Value is *K*+1 and TYPE in *PIDA* does not indicate ROW, ARRAY, or ARRAY LOCATOR, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value.*