INTERNATIONAL STANDARD ISO/IEC 9075-9:2001
TECHNICAL CORRIGENDUM 1

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 9:
Management of External Data (SQL/MED)

TECHNICAL CORRIGENDUM 1

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

Partie 9: Gestion des données externes (SQL/MED)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 9075-9:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

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 9075-9:2001/Cor.1:2003(E)**

Published in Switzerland

Contents

Information technology — Database languages — SQL —

Part 9: Management of External Data (SQL/MED)

TECHNICAL CORRIGENDUM 1

4.5 User mappings

1. Rationale: Do not describe non-schema objects as being in a catalog.

Replace the 2nd paragraph with:

A user mapping is defined by invoking an <user mapping definition>. Invocation of an <user mapping definition> results in the creation of a user mapping descriptor in the SQL-environment. A user mapping descriptor consists of:

— An authorization identifier.

— A foreign server name, identifying a foreign server descriptor.

— A generic options descriptor.

4.8 Datalinks

1. Rationale: Clarify the semantics of built-in functions.

Replace the 2nd paragraph with:

The File Reference of a datalink is accessible by invoking operators defined in this part of ISO/IEC 9075. The character set of the File Reference, referred to as the *datalink character set* is implementation-defined.

Replace the 7th paragraph with:

With the function provided by datalinks and the datalinker, it is possible to specify that access to the files should be mediated by the SQL-server rather than by the external data manager. When access to the files is mediated by an SQL-server, any request to access a file must operate on an SQL-mediated datalink to obtain a character string with which to reference the file, using one of the operators provided for that purpose. This character string is constructed by combining the File Reference of a datalink value with an encrypted value called an *access token*. The generation of the access token and the method of combining it with the File Reference is implementation-dependent. When the application uses the returned character string value to access a file, the datalinker checks to see if the access token is *valid*. If it is valid, then the application is allowed to access the file pointed to by the File Reference. Every attempt by an application to access, without a valid access token, a file referenced by an SQL-mediated datalink is unsuccessful. The time at which a valid access token ceases to be valid is implementation-defined.

2. *Rationale: Clarify assignable and comparable.*

Insert the following paragraph after the 7th paragraph:

Datalinks are not comparable. A datalink is assignable only to sites of type DATALINK.

4.9 Type conversions and mixing of data types

1. *Rationale: Clarify assignable and comparable.*

Delete the subclause.

4.18.1 Handles

1. *Rationale: Editorial.*

In the 2nd paragraph, replace the 4th bullet with:

— **Request handle**: This handle is allocated by the SQL-server to reference an SQL-statement that is to be executed by a foreign server. A request handle may reference a simple statement, such as `SELECT * FROM T`, or it may reference a complex statement that includes predicates, joins, ordering, *etc*. A request handle is used by the foreign-data wrapper to retrieve (for example) the names of foreign tables referenced in the from clause, the names of column references in the select list, *etc*, using foreign-data wrapper interface SQL-server routines. This handle is allocated implicitly.

4.18.4 Return codes

1. *Rationale: Add missing text for Success with Information condition.*

Insert the following into the 1st paragraph as the second bullet:

— A value of 1 (one) indicates **Success with information**. The foreign-data wrapper interface routine executed successfully but a completion condition was raised: *warning*.

4.18.7 Foreign-data wrapper descriptor areas

1. *Rationale: Editorial - misspelled acronym.*

Modify the 2nd bullet of the 5th paragraph with:

— Wrapper Row Descriptor (WRD): This descriptor is allocated by the SQL-server if a foreign-data wrapper requests its allocation. It is used to describe the result of a statement to be executed by that foreign-data wrapper in pass-through mode, and is associated with an ExecutionHandle. The foreign-data wrapper uses the `SetDescriptor()` routine to set information in the WRD. The SQL-server can obtain the handle to a WRD by invoking the `GetWRDHandle()` routine. It can then retrieve the information in that WRD by invoking the `GetDescriptor()` routine.

5.2 <token> and <separator>

1. Rationale: Correct the BNF of <non-reserved word> and <reserved word>.

In the Format, replace the productions for <non-reserved word> and <reserved word> with:

```
<non-reserved word> ::=
      !! All alternatives from ISO/IEC 9075-2
    | !!All alternatives from ISO/IEC 9075-5
    | BLOCKED
    | CONTROL
    | DB
    | FILE | FS
    | INTEGRITY
    | LIBRARY | LIMIT | LINK
    | MAPPING
    | PASSTHROUGH | PERMISSION
    | RECOVERY | RESTORE
    | SELECTIVE | SERVER
    | UNLINK
    | VERSION
    | WRAPPER
    | YES

<reserved word> ::=
      !! All alternatives from ISO/IEC 9075-2
    | !!All alternatives from ISO/IEC 9075-5
    | DATALINK | DLURLCOMPLETE | DLURLPATH | DLURLPATHONLY | DLURLSCHEME
    | DLURLSERVER | DLVALUE
    | IMPORT
```

6.2 <column reference>

1. Rationale: there is no <row value expression> immediately contained in a <set clause>.

Replace Access Rule 1) with:

1) Replace AR 1) If *CR* is a <column reference> whose qualifying table is a base table, a foreign table or a viewed table and that is contained in any of:

   — A <query expression> simply contained in a <cursor specification>, a <view definition> or an <insert statement>.

   — A <sort specification list> contained in a <cursor specification>.

   — A <table expression> immediately contained in a <select statement: single row>.

   — A <search condition> immediately contained in a <trigger definition>, a <delete statement: searched> or an <update statement: searched>.

   — A <select list> immediately contained in a <select statement: single row>.

   — A <value expression> simply contained in an <update source>.

then let *C* be the column referenced by *CR*.

Case:

a) If <column reference> is contained in an <SQL schema statement>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT for *C*.

b) Otherwise, the current privileges shall include SELECT on *C*.

NOTE 24 – ''applicable privileges'' and ''current privileges'' are defined in Subclause 11.1, ''<privileges>''.

6.5 <datalink value>

Replace Syntax Rule 4) with:

4) The character set name, collating sequence, and coercibility characteristic of the File Reference of the result of valuating a the <datalink value constructor> are the character set name, collation, and coercibility characteristic, respectively, of the <data location>.

7.1 <table reference>

1. Rationale: There is no <row value expression> immediately contained in a <set clause>.

Replace Access Rule 1) b) with:

1) b) If *T* is a base table, a foreign table or a viewed table and the <table reference> is contained in any of:

— A <query expression> simply contained in a <cursor specification>, a <view definition>, or an <insert statement>.

— A <table expression> or <select list> immediately contained in a <select statement: single row>.

— A <search condition> immediately contained in a <delete statement: searched> or an <update statement: searched>.

— A <value expression> simply contained in an <update source>.

then

Case:

i) If <table reference> is contained in an <SQL schema statement> then, the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT on at least one column of *T*.

ii) Otherwise, the current privileges shall include SELECT on at least one column of *T*.

NOTE 29 — "applicable privileges" and "current privileges" are defined in Subclause 11.1, "<privileges>".

2. Rationale: Delete incorrect implication that there will only every be 1 TRH.

Replace General Rule 1) b) ix) with:

1) b) ix) Let *TRDH* be the TableReferenceDescriptorHandle allocated for *TRD*.

3. Rationale: Clarify when a descriptor handle is used instead of the descriptor itself.

Replace General Rules 1) b) xvii) and 1) b) xviii) with:

1) b) xvii) Let *NC* be the value of the COUNT descriptor field that would be returned by invocation of the `GetDescriptor()` routine with *TRDH* as the DescriptorHandle parameter, 0 (zero) as the RecordNumber parameter, and the code for COUNT from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter.

1) b) xviii) Let DT_j be the effective data type of the *j*-th column, for 1 (one) ≤ *j* ≤ *NC*, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISON, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be returned by separate invocations of the `GetDescriptor()` routine with *TRDH* as the DescriptorHandle parameter, *j* as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISON, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32 "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter.

4. Rationale: Correct undefined tag.

Replace General Rule 1) b) xxvi) with:

1) b) xxvi) The <table reference> references the table that consists of every row returned by the repeated invocation of the `Iterate( )` routine in the library identified by *WRLN* with *EXH* as the argument until the return code indicates **No data found**.

13.1 <foreign server definition>

1. Rationale: Delete the optional AUTHORIZATION-clause from <foreign server definition>.

In the Format replace the production for <foreign server definition> with:

```
<foreign server definition> ::=
    CREATE SERVER <foreign server name>
      [ TYPE <server type> ]
      [ VERSION <server version> ]
      FOREIGN DATA WRAPPER <foreign-data wrapper name>
      [ <generic options> ]
```

Delete Syntax Rule 3).

Replace General Rule 1) e) with:

1) e) The current authorization identifier.

Replace General Rule 2) with:

2) A privilege descriptor is created that defines the USAGE privilege on this foreign server to the current authorization identifier. The grantor of the privilege descriptor is set to the special grantor value ''_SYSTEM''. This privilege is grantable.

13.4 <foreign-data wrapper definition>

1. Rationale: Delete the optional AUTHORIZATION-clause from <foreign-data wrapper definition>.

In the Format replace the production for <foreign-data wrapper definition> with:

```
<foreign-data wrapper definition> ::=
    CREATE FOREIGN DATA WRAPPER <foreign-data wrapper name>
      [ <library name specification> ]
      <language clause>
      [ <generic options> ]
```

Delete Syntax Rule 3).

Replace General Rule 1) b) with:

1) b) The current authorization identifier.

Replace General Rule 2) with:

2) A privilege descriptor is created that defines the USAGE privilege on this foreign-data wrapper to the current authorization identifier. The grantor of the privilege descriptor is set to the special grantor value ''_SYSTEM''. This privilege is grantable.

14.1 <revoke statement>

1. Rationale: Cater properly for abandoned foreign table descriptors.

Replace Syntax Rule 1) with:

1) Insert after SR 21) Let *T* be any foreign table descriptor included in *S1*. *T* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having USAGE privilege on the foreign server associated with the foreign table described by *T*.

Replace Syntax Rule 3) with:

3) Augment SR 36) Add abandoned foreign server descriptor, and abandoned foreign table descriptor to the list of objects that shall not exist.

Insert the following General Rule:

2) Insert this GR For every abandoned foreign table descriptor *FT*, let *FTN* be the <table name> of *FT*. The following <drop foreign table statement> is effectively executed without further Access Rule checking:

```
DROP FOREIGN TABLE S1.FTN CASCADE
```

14.3 <alter user mapping>

1. Rationale: Correct symbol.

Replace General Rule 1) with:

1) The General Rules of Subclause 11.3, ''<alter generic options>'', are applied to *AGO* with the generic options descriptor included in *UMD* as the applicable generic options descriptor.

18.4 <describe statement>

1. Rationale: Clarify that wrapper row descriptor is meant by WRD.

Replace General Rule 1) a) with:

1) a) Let *EXH* be the ExecutionHandle associated with <SQL statement name>. Let *WPD* and *WRD* be the wrapper parameter descriptor and wrapper row descriptor, respectively, associated with the WPDHandle and WRDHandle, respectively, that would be returned by the invocation of the `GetWPDHandle()` and `GetWRDHandle()` routines with *EXH* as the ExecutionHandle parameter.

18.6 <output using clause>

1. Rationale: Clarify that wrapper row descriptor is meant by WRD, and server row descriptor by SRD.

Replace General Rule 1) c) with:

1) c) Let *EXH* be the ExecutionHandle associated with *SN*. Let *WRD* and *SRD* be the wrapper row descriptor and server row descriptor, respectively, associated with the WRDHandle and SRDHandle, respectively, that would be returned by the invocation of the `GetWRDHandle()` and `GetSRDHandle()` routines with *EXH* as the ExecutionHandle parameter.

22.9 Tables used with SQL/MED

1. Rationale: Remove a row that had been forgotten to be removed when a new data retrieval architecture was introduced.

Delete the following row from Table 33 — Codes used for foreign-data wrapper handle types

Handle type	Code

DataHandle	5

23.2 <foreign-data wrapper interface routine> invocation

1. Rationale: Add missing text for Success with Information condition.

Insert the following General Rule:

4) a) ii) 1.1) If a completion condition is raised: *warning*, then *RC* is set to indicate **Success with information**.

2. Rationale: Add missing text for Success with Information condition.

Insert the following General Rule:

4) b) iii.1) If *RN* is a foreign-data wrapper interface wrapper routine, then the actions of the invoking SQL-server in response to the failed execution of *RN* are implementation-dependent.

23.3.1 AllocWrapperEnv

1. Rationale: Make implementation-defined explicit.

Replace General Rule 3) with:

3) If the implementation-defined maximum number of foreign-data wrapper environments that can be allocated at one time has already been reached, then an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*. A skeleton FDW-environment is allocated and is assigned a unique value that is returned in WrapperEnvHandle.

23.3.3 ConnectServer

1. Rationale: Make implementation-defined explicit.

Replace General Rule 13) with:

13) If the implementation-defined maximum number of FS-connections that can be allocated at one time has already been reached, then FSConnectionHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.

23.3.18 InitRequest

1. Rationale: Make implementation-defined explicit.

Replace General Rule 7) with:

7) If the implementation-defined maximum number of FDW-replies that can be allocated at one time has already been reached, then ReplyHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.

Replace General Rule 10) with:

10) If the implementation-defined maximum number of FDW-executions that can be allocated at one time has already been reached, then ExecutionHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded.*

2. *Rationale: Clarify when a descriptor handle is used instead of the descriptor itself.*

Replace General Rule 14) with:

14) Let *NIDA* be the number of item descriptor areas that must be set up for the server row descriptor. Let *SRDHandle* be the DescriptorHandle that is returned by an invocation of the `AllocDescriptor()` routine with *NIDA* as the MaxDetailAreas parameter. Let *SRD* be the server row descriptor identified by *SRDHandle*. *SRD* is associated with the allocated FDW-execution. For this descriptor area, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the `SetDescriptor()` routine with *SRDHandle* as the DescriptorHandle parameter and *r* as the Record-Number parameter, 1 (one) ≤ *r* ≤ *NIDA*, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *SRD* are initially undefined.

23.3.20 Open

1. *Rationale: Define previously undefined symbols.*

Insert the following General Rule:

5) a.0) Let *SRD* be the SRDHandle that would be returned by an invocation of the `GetSRDHandle()` routine with *EH* as the ExecutionHandle parameter. Let *SPD* be the SPDHandle that would be returned by an invocation of the `GetSPDHandle()` routine with *EH* as the ExecutionHandle parameter. Let *WRD* be the WRDHandle that would be returned by an invocation of the `GetWRDHandle()` routine with *EH* as the ExecutionHandle parameter. Let *WPD* be the WPDHandle that would be returned by an invocation of the `GetWPDHandle()` routine with *EH* as the ExecutionHandle parameter.

2. *Rationale: Define previously undefined symbols.*

Replace General Rule 5) with:

5) c) Let TDT_j be the effective data type of the *j*-th <target specification>, for 1 (one) ≤ $_j$ ≤ *NCR*, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISON, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be set by separate invocations of the `GetDescriptor( )` routine with *SRD* as the DescriptorHandle parameter, $_j$ as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISON, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,

USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, ''Codes used for foreign-data wrapper descriptor fields'', as the FieldIdentifier parameter. TYPE either indicates ROW or is one of the code values in Table 17, ''Codes used for application data types in SQL/CLI''.

23.3.22 TransmitRequest

1. Rationale: Make implementation-defined explicit.

Replace General Rule 6) with:

6) If the implementation-defined maximum number of FDW-executions that can be allocated at one time has already been reached, then ExecutionHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.

2. Rationale: Clarify when a descriptor handle is used instead of the descriptor itself.

Replace General Rules 10), 11), 12), and 13) with:

10) Let *SRDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the server row descriptor. Let *SRDHandle* be the DescriptorHandle that is returned by an invocation of the `AllocDescriptor()` routine with *SRDItemDescriptorAreas* as the MaxDetailAreas parameter. Let *SRD* be the server row descriptor identified by *SRDHandle*. *SRD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the `SetDescriptor()` routine with *SRDHandle* as the DescriptorHandle parameter and *r* as the Record-Number parameter, 1 (one) ≤ *r* ≤ *SRDItemDescriptorAreas*, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *SRD* are initially undefined.

11) Let *SPDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the server parameter descriptor. Let *SPDHandle* be the DescriptorHandle that is returned by an invocation of the `AllocDescriptor()` routine with *SPDItemDescriptorAreas* as the MaxDetailAreas parameter. Let *SPD* be the server parameter descriptor identified by *SPDHandle*. *SPD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the `SetDescriptor()` routine with *SPDHandle* as the DescriptorHandle parameter and *r* as the Record-Number parameter, 1 (one) ≤ *r* ≤ *SPDItemDescriptorAreas*, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *SPD* are initially undefined.

12) Let *WRDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the wrapper row descriptor. Let *WRDHandle* be the DescriptorHandle that is returned by an invocation of the `AllocDescriptor()` routine with *WRDItemDescriptorAreas* as the MaxDetailAreas parameter. Let *WRD* be the wrapper row descriptor identified by *WRDHandle*. *WRD* is associated with

the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the `SetDescriptor()` routine with *WRD Handle* as the DescriptorHandle parameter and *r* as the Record-Number parameter, 1 (one) ≤ *r* ≤ *WRDItemDescriptorAreas*, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *WRD* are initially undefined.

13) Let *WPDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the wrapper parameter descriptor. Let *WPD Handle* be the DescriptorHandle that is returned by an invocation of the `AllocDescriptor()` routine with *WPDItemDescriptorAreas* as the MaxDetailAreas parameter. Let *WPD* be the wrapper parameter descriptor identified by *WPD Handle*. *WPD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the `SetDescriptor()` routine with *WPD Handle* as the DescriptorHandle parameter and *r* as the Record-Number parameter, 1 (one) ≤ *r* ≤ *WPDItemDescriptorAreas*, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *WPD* are initially undefined.

23.4.1 AllocDescriptor

1. Rationale: Make implementation-defined explicit.

Replace General Rule 2) with:

2) If the implementation-defined maximum number of foreign-data wrapper descriptor areas that can be allocated at one time has already been reached, then DescriptorHandle is set to 0 (zero) and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.

23.4.2 FreeDescriptor

1. Rationale: Define a previously undefined symbol.

Insert the following General Rule before General Rule 1):

0.1) Let *DH* be the value of DescriptorHandle.

23.4.4 GetDescriptor

1. Rationale: Correct a bug related to retrieving the value of a field in a foreign-data wrapper descriptor area and editorial changes.

Delete General Rule 7).

Delete General Rule 8).

Replace General Rule 11) with:

11) If *TYPE* is 'HEADER', then header information from the descriptor area *D* is retrieved as follows.

NOTE 68.1 — Let *MBR* be the value of the May Be Retrieved column in the row of Table 34, "Ability to retrieve foreign-data wrapper descriptor fields", that contains *FI* and the column that contains the descriptor type of *D*. If *MBR* is 'No', then the effect on the retrieved value is implementation-dependent.

Case:

a) If *FI* indicates COUNT, then the value retrieved is *N*.

b) If *FI* indicates an implementation-defined descriptor header field, then the value retrieved is the value of the implementation-defined descriptor header field identified by *FI*.

c) Otherwise, if *FI* indicates a descriptor header field defined in Table 32, "Codes used for foreign-data wrapper descriptor fields", then the value retrieved is the value of the descriptor header field identified by *FI*.

Replace General Rule 12) with:

12) If *TYPE* is 'ITEM', then item information from the descriptor area *D* is retrieved as follows:

NOTE 68.2 — Let *MBR* be the value of the May Be Retrieved column in the row of Table 34, "Ability to retrieve foreign-data wrapper descriptor fields", that contains *FI* and the column that contains the descriptor type of *D*. If *MBR* is 'No', then the effect on the retrieved value is implementation-dependent.

Case:

a) If *FI* indicates an implementation-defined descriptor item field, then the value retrieved is the value of the implementation-defined descriptor item field of *IDA* identified by *FI*.

b) Otherwise, if *FI* indicates a descriptor item field defined in Table 32, "Codes used for foreign-data wrapper descriptor fields", then the value retrieved is the value of the descriptor item field identified by *FI*.

23.4.36 SetDescriptor

1. Rationale: Correct a bug related to setting the value of a field in a foreign-data wrapper descriptor area.

Delete General Rule 5).

Delete General Rule 6).

Delete General Rule 7).

Replace General Rule 13) with:

13) Information is set in *D*:

NOTE 70.1 — Let *MBS* be the value of the May Be Set column in the row of Table 34, "Ability to set foreign-data wrapper descriptor fields", that contains *FI* and the column that contains the descriptor type of *D*. If *MBS* is

'No', then the effect on field that the value is set to is implementation-dependent.

Case:

a) If *FI* indicates COUNT, then

   Case:

   i) If the memory requirements to manage the foreign-data wrapper descriptor area cannot be satisfied, then an exception condition is raised: *FDW-specific condition — memory allocation error*.

   ii) Otherwise, the count of the number of foreign-data wrapper item descriptor areas is set to the value of Value.

b) If *FI* indicates OCTET_LENGTH, then the value of the OCTET_LENGTH field of *IDA* is set to the value of Value.

c) If *FI* indicates DATA_POINTER, then the value of the DATA_POINTER field of *IDA* is set to the address of Value. If Value is a null pointer, then the address is set to 0 (zero).

d) If *FI* indicates DATA, then the value of the DATA field of *IDA* is set to the value of Value.

e) If *FI* indicates INDICATOR, then the value of the INDICATOR field of *IDA* is set to the value of Value.

f) If *FI* indicates RETURNED_CARDINALITY, then the value of the RETURNED_CARDINALITY field of *IDA* is set to the value of Value.

g) If *FI* indicates CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, or CHARACTER_SET_NAME, then:

   i) Let *BL* be the value of BufferLength.

   ii) Case:

      1) If *BL* is not negative, then let *L* be *BL*.

      2) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.

   iii) Case:

      1) If *L* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.

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

         TRIM (BOTH '' FROM *FV*)

   iv) Let *ML* be the maximum length in characters allowed for an <identifier> as specified in the Syntax Rules of Subclause 5.4, "Names and identifiers", in ISO/IEC 9075-2, and let *TFVL* be

the length in characters of *TFV*.

v) Case:

1) If *TFVL* is greater than *ML*, then *FV* is set to the first *ML* characters of *TFV* and a completion condition is raised: *warning — string data, right truncation*.

2) Otherwise, *FV* is set to *TFV*.

vi) Case:

1) If *FI* indicates CHARACTER_SET_CATALOG and *FV* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid catalog name*.

2) If *FI* indicates CHARACTER_SET_SCHEMA and *FV* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid schema name*.

3) If *FI* indicates CHARACTER_SET_NAME and *FV* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid character set name*.

vii) The value of the field of *IDA* identified by *FI* is set to the value of *FV*.

h) Otherwise, the value of the field of *IDA* identified by *FI* is set to the value of Value.

23.5.1 GetDiagnostics

1. *Rationale: Remove a rule that should have been removed when a new data retrieval architecture was introduced.*

Delete General Rule 3) e).

25.1 ATTRIBUTES view

1. *Rationale: Remove an unintentional incompatibility between SQL-92 and SQL-99, correct references to CURRENT_ROLE, recognise roles in the WHERE clause of the TABLES view and make the value of the ATTRIBUTE/COLUMN DEFAULT visible to any user who has some privilege on the attribute/column.*

In the Definition, replace the column specification for ATTRIBUTE_DEFAULT with:

```
ATTRIBUTE_DEFAULT,
```

2. *Rationale: Remove an unintentional incompatibility between SQL-92 and SQL-99, correct references to CURRENT_ROLE, recognise roles in the WHERE clause of the TABLES view and make the value of the ATTRIBUTE/COLUMN DEFAULT visible to any user who has some privilege on the attribute/column.*

In the Definition, replace the <where clause> with:

```
WHERE ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME )
```

```
        IN ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG,
                    UDTP.USER_DEFINED_TYPE_SCHEMA, UDTP.USER_DEFINED_TYPE_NAME
               FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
              WHERE ( UDTP.GRANTEE IN ( 'PUBLIC', CURRENT_USER )
                      OR
                      UDTP.GRANTEE IN ( SELECT ROLE_NAME
                                          FROM ENABLED_ROLES )
                    )
           )
        AND
        A.UDT_CATALOG = ( SELECT CATALOG_NAME
                            FROM INFORMATION_SCHEMA_CATALOG_NAME )
;
```

25.2 COLUMN_OPTIONS view

1. *Rationale: Remove an unintentional incompatibility between SQL-92 and SQL-99, correct references to CURRENT_ROLE, recognise roles in the WHERE clause of the TABLES view and make the value of the ATTRIBUTE/COLUMN DEFAULT visible to any user who has some privilege on the attribute/column.*

In the Definition, replace the <where clause> with:

```
WHERE ( CO.TABLE_CATALOG, CO.TABLE_SCHEMA, CO.TABLE_NAME,
        CO.COLUMN_NAME )
       IN ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME,
                   CP.COLUMN_NAME
              FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
             WHERE ( CP.GRANTEE IN ( 'PUBLIC', CURRENT_USER )
                     OR
                     CP.GRANTEE IN ( SELECT ROLE_NAME
                                       FROM ENABLED_ROLES )
                   )
          )
       AND
       CO.TABLE_CATALOG = ( SELECT CATALOG_NAME
                              FROM INFORMATION_SCHEMA_CATALOG_NAME )
;
```

25.3 COLUMNS view

1. *Rationale: Remove an unintentional incompatibility between SQL-92 and SQL-99, correct references to CURRENT_ROLE, recognise roles in the WHERE clause of the TABLES view and make the value of the ATTRIBUTE/COLUMN DEFAULT visible to any user who has some privilege on the attribute/column.*

In the Definition, replace the column specification for COLUMN_DEFAULT with:

```
COLUMN_DEFAULT,
```

2. *Rationale: Remove an unintentional incompatibility between SQL-92 and SQL-99, correct references to CURRENT_ROLE, recognise roles in the WHERE clause of the TABLES view and make the value of the ATTRIBUTE/COLUMN DEFAULT visible to any user who has some privilege on the attribute/column.*

In the Definition, replace the <where clause> with:

```
WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
        C.COLUMN_NAME )
```