INTERNATIONAL STANDARD ISO/IEC 13249-3:1999
TECHNICAL CORRIGENDUM 1

Published 2003-10-15

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 Multimedia and Application Packages —

Part 3: Spatial

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Langages de bases de données — Multimédia SQL et paquetages d'application —

Partie 3: Spatial

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 13249-3:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange.*

Statement of purpose of rationale:

A statement indicating the rationale for each change to ISO/IEC 13249-3:1999(E) 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 Definitional Rules and Descriptions, 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 13249-3:1999/Cor.1:2003(E)**

Published in Switzerland

Contents

Page

3.1.2.3 2-dimensional geometry

1. Rationale: Correct description.

Replace definition with:

a geometry with a geometric dimension of 2

3.1.5.5 meridian

1. Rationale: Improve and correct description.

Replace the definition with:

intersection between an ellipsoid and a plane containing the semi-minor axis of the ellipsoid

NOTE This term is often used for the pole-to-pole arc rather than the complete closed figure.

3.1.2.7 closed curve

1. Rationale: Correct description.

Replace definition with:

a curve such that its start point is equal to its end point

3.1.2.12 linear ring

1. Rationale: Correct description.

Replace definition with:

a linestring that is closed and simple

3.1.2.14 non-closed curve

1. Rationale: Correct description.

Replace definition with:

a curve such that its start point is not equal to its end point

3.1.2.17 ring

1. Rationale: Correct description.

Replace definition with:

a curve that is closed and simple

4.1.10 ST_CurvePolygon

1. Rationale: Correct description.

Replace the second paragraph with:

ST_CurvePolygon values are topologically closed. The boundary of an ST_CurvePolygon consists of an exterior ring and zero or more interior rings. No two rings in the boundary cross. The rings in the boundary of an ST_CurvePolygon value may intersect at a point but only as a tangent. An ST_CurvePolygon shall not have cut lines, spikes or punctures. The interior of every ST_CurvePolygon is a connected point set. The exterior of an ST_CurvePolygon with one or more holes is not connected. Each hole defines a disconnected component of the exterior.

4.1.15 ST_MultiLineString

1. Rationale: Correct type hierarchy description.

Replace the paragraph with:

The ST_MultiLineString type is a subtype of ST_MultiCurve. The elements of an ST_MultiLineString are restricted to ST_LineString values.

4.1.16 ST_MultiSurface

1. Rationale: Correct description.

Replace the paragraph with:

The ST_MultiSurface type is a subtype of ST_GeomCollection. The ST_MultiSurface type may be instantiable. An ST_MultiSurface is a 2-dimensional geometry collection. The elements of an ST_MultiSurface value are restricted to ST_Surface values. The interiors of any two ST_Surface values in an ST_MultiSurface shall not intersect. The boundaries of any two elements in an ST_MultiSurface may intersect at a finite number of ST_Point values.

4.1.17 ST_MultiPolygon

1. Rationale: Correct description.

Replace the first paragraph with:

The ST_MultiPolygon type is a subtype of ST_MultiSurface. The elements of an ST_MultiPolygon value are restricted to ST_Polygon values. The interiors of distinct element of an ST_MultiPolygon do not intersect. The interiors of two ST_Polygon values that are elements of an ST_MultiPolygon shall not intersect. The boundaries of any two ST_Polygon values that are elements of an ST_MultiPolygon shall not cross and may touch at only a finite number of points. An ST_MultiPolygon value is defined to be topologically closed.

Replace the second paragraph with:

An ST_MultiPolygon value shall not have cut lines, spikes or punctures. An ST_MultiPolygon value is a closed point set. The interior of an ST_MultiPolygon value with more than one ST_Polygon value is not a connected point set. The number of connected components of the interior of an ST_MultiPolygon is equal to the number of ST_Polygon values in the ST_MultiPolygon. The boundary of an ST_MultiPolygon value is a set of linear rings corresponding to the boundaries of the ST_Polygon elements.

5.1.13 ST_Buffer Method

1. Rationale: Correct the parameter value description.

Replace Description 2) with:

2) The parameter *adistance* is measured in an implementation-defined linear unit of measure in the spatial reference system of SELF.

5.1.19 ST_Distance Method

1. Rationale: Correct the returned value description.

Replace Description 3) with:

3) Case:

   a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Distance(ST_Geometry)* is in the linear unit of measure identified by <linear unit>.

   b) Otherwise, the value returned by *ST_Distance(ST_Geometry)* is in an implementation-defined unit of measure.

6.1.1 ST_Point Type

1. *Rationale: Correct constructor method definitions.*

Replace the two <original method specification>s that begins with "METHOD ST_Point" in the Definition with:

```
CONSTRUCTOR METHOD ST_Point
    (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION)
    RETURNS ST_Point
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Point
    (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION, asrid INTEGER)
    RETURNS ST_Point
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```

6.1.2 ST_Point Methods

1. *Rationale: Correct constructor method definitions.*

Replace the Definition with:


```
CREATE CONSTRUCTOR METHOD ST_Point
    (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION)
    RETURNS ST_Point
    FOR ST_Point
    RETURN SELF.                        -- Return an ST_Point value with
        ST_PrivateDimension(0).             -- dimension = 0,
        ST_PrivateCoordinateDimension(2). -- coordinate dimension = 2,
        ST_SRID(0).                         -- SRID = asrid,
        ST_X(xcoord).                       -- ST_X = xcoord,
        ST_Y(ycoord)                        -- ST_Y = ycoord

CREATE CONSTRUCTOR METHOD ST_Point
    (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION, asrid INTEGER)
    RETURNS ST_Point
    FOR ST_Point
    RETURN SELF.                        -- Return an ST_Point value with
        ST_PrivateDimension(0).             -- dimension = 0,
        ST_PrivateCoordinateDimension(2). -- coordinate dimension = 2,
        ST_SRID(asrid).                     -- SRID = asrid,
        ST_X(xcoord).                       -- ST_X = xcoord,
        ST_Y(ycoord)                        -- ST_Y = ycoord
```

Replace the first sentence of Description 2) with:

2) The null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION)* returns an *ST_Point* value with:

Replace the first sentence of Description 4) with:

4) The null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* returns an *ST_Point* value with:

7.1.2 ST_Length Method

1. Rationale: Correct the returned value description.

Replace Description 3) with:

3) Case:

   a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Length*() is in the linear unit of measure identified by <linear unit>.

   b) Otherwise, the value returned by *ST_Length*() is in an implementation-defined unit of measure.

7.2.1 ST_LineString Type

1. Rationale: Correct constructor method definitions.

Replace the two <original method specification>s that begins with "METHOD ST_LineString" in the Definition with:

```
CONSTRUCTOR METHOD ST_LineString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
    RETURNS ST_LineString
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_LineString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
     asrid INTEGER)
    RETURNS ST_LineString
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```


7.2.2 ST_LineString Methods

1. Rationale: Correct constructor method definitions.

Replace the Definition with:

```
CREATE CONSTRUCTOR METHOD ST_LineString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
    RETURNS ST_LineString
    FOR ST_LineString
    RETURN SELF.ST_SRID(0).ST_Points(apointarray)

CREATE CONSTRUCTOR METHOD ST_LineString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
     asrid INTEGER)
    RETURNS ST_LineString
    FOR ST_LineString
    RETURN SELF.ST_SRID(asrid).ST_Points(apointarray)
```

Replace the first sentence of Description 2) with:

2) The null-call type preserving SQL-invoked constructor method *ST_LineString(ST_Point ARRAY)* returns an *ST_LineString* value with:

Replace the first sentence of Description 4) with:

4) The null-call type preserving SQL-invoked constructor method *ST_LineString(ST_Point ARRAY, INTEGER)* returns an *ST_LineString* value with:

7.2.5 ST_PointN Method

1. Rationale: Use a return value for such completion codes.

Replace the Definition with the following:

```
CREATE METHOD ST_PointN(aposition INTEGER)
    RETURNS ST_Point
    FOR ST_LineString
    BEGIN
        IF SELF.ST_NumPoints = 0 THEN
          BEGIN
             SIGNAL SQLSTATE '01F06'
                SET MESSAGE_TEXT = 'empty geometry';
             RETURN NULL;
          END;
        END IF;
        IF aposition < 1 OR
          aposition > SELF.ST_NumPoints THEN
          BEGIN
             SIGNAL SQLSTATE '01F01'
                SET MESSAGE_TEXT = 'invalid position';
          RETURN NULL;
          END;
        END IF;
        RETURN SELF.ST_PrivatePoints[aposition];
    END
```

In the Description section, replace Description 2) with:

2) For the null-call method *ST_PointN(INTEGER)*:

   Case:

   a) If the cardinality of the *ST_PrivatePoints* attribute is equal to 0 (zero), then:

      i) A completion condition is raised: SQL/MM Spatial warning – empty geometry.

      ii) Return the null value.

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivatePoints* attribute, then:

      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.

      ii) Return the null value.

   c) Otherwise, return an *ST_Point* value at element *aposition* in the *ST_PrivatePoints* attribute of SELF.

7.3.1 ST_CircularString Type

1. Rationale: Correct constructor method definitions.

Replace the two <original method specification>s that begins with "METHOD ST_CircularString" in the Definition with:

```
CONSTRUCTOR METHOD ST_CircularString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
    RETURNS ST_CircularString
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```

```
CONSTRUCTOR METHOD ST_CircularString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
     asrid INTEGER)
    RETURNS ST_CircularString
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```

7.3.2 ST_CircularString Methods

1. Rationale: Correct constructor method definitions.

Replace the Definition with:

```
CREATE CONSTRUCTOR METHOD ST_CircularString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
    RETURNS ST_CircularString
    FOR ST_CircularString
    RETURN SELF.ST_SRID(0).ST_Points(apointarray)

CREATE CONSTRUCTOR METHOD ST_CircularString
    (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
     asrid INTEGER)
    RETURNS ST_CircularString
    FOR ST_CircularString
    RETURN SELF.ST_SRID(asrid).ST_Points(apointarray)
```


Replace the first sentence of Description 2) with:

2) The null call type preserving SQL-invoked constructor method *ST_CircularString(ST_Point ARRAY)* returns an *ST_CircularString* value with:

Replace the first sentence of Description 4) with:

4) The null call type preserving SQL-invoked constructor method *ST_CircularString(ST_Point ARRAY, INTEGER)* returns an *ST_CircularString* value with:

7.3.5 ST_PointN Method

1. Rationale: Use a return value for such completion codes.

Replace the Definition with the following:

```
CREATE METHOD ST_PointN(aposition INTEGER)
    RETURNS ST_Point
    FOR ST_CircularString
    BEGIN
        IF SELF.ST_NumPoints = 0 THEN
          BEGIN
            SIGNAL SQLSTATE '01F06'
                SET MESSAGE_TEXT = 'empty geometry';
            RETURN NULL;
          END;
        END IF;
        IF aposition < 1 OR
          aposition > SELF.ST_NumPoints THEN
          BEGIN
            SIGNAL SQLSTATE '01F01'
                SET MESSAGE_TEXT = 'invalid position';
            RETURN NULL;
          END;
        END IF;
        RETURN SELF.ST_PrivatePoints[aposition];
    END
```

In the Description section, replace Description 2) with:

2) For the null-call method *ST_PointN(INTEGER)*:

   Case:

   a) If the cardinality of the attribute *ST_PrivatePoints* is equal to 0 (zero), then:

      i) A completion condition is raised: *SQL/MM Spatial warning – empty geometry*.

      ii) Return the null value.

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the attribute *ST_PrivatePoints*, then:

      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.

      ii) Return the null value.

   c) Otherwise, return an *ST_Point* value at element *aposition* in the attribute *ST_PrivatePoints* of SELF.

7.4.1 ST_CompoundCurve Type

1. Rationale: Correct constructor method definitions.

Replace the four <original method specification>s that begins with "METHOD ST_CompoundCurve" in the Definition with:


```
CONSTRUCTOR METHOD ST_CompoundCurve(acurve ST_Curve)
    RETURNS ST_CompoundCurve
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```



```
CONSTRUCTOR METHOD ST_CompoundCurve
    (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
    RETURNS ST_CompoundCurve
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
    (acurve ST_Curve,
     asrid INTEGER)
    RETURNS ST_CompoundCurve
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
    (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
     asrid INTEGER)
    RETURNS ST_CompoundCurve
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```

7.4.2 ST_CompoundCurve Methods

1. Rationale: Correct constructor method definitions.

Replace the Definition with:

```
CREATE CONSTRUCTOR METHOD ST_CompoundCurve
    (acurve ST_Curve)
    RETURNS ST_CompoundCurve
    FOR ST_CompoundCurve
    RETURN SELF.ST_SRID(0).ST_Curves(ARRAY[acurve])

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
    (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
    RETURNS ST_CompoundCurve
    FOR ST_CompoundCurve
    RETURN SELF.ST_SRID(0).ST_Curves(acurvearray)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
    (acurve ST_Curve,
     asrid INTEGER)
    RETURNS ST_CompoundCurve
    FOR ST_CompoundCurve
    RETURN SELF.ST_SRID(asrid).ST_Curves(ARRAY[acurve])

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
    (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
     asrid INTEGER)
    RETURNS ST_CompoundCurve
    FOR ST_CompoundCurve
    RETURN SELF.ST_SRID(asrid).ST_Curves(acurvearray)
```


Replace the first sentence of Description 2) with:

2) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve)* returns an *ST_CompoundCurve* value with:

Replace the first sentence of Description 4) with:

4) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve ARRAY)* returns an *ST_CompoundCurve* value with:

Replace the first sentence of Description 6) with:

6) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve, INTEGER)* returns an *ST_CompoundCurve* value with:

Replace the first sentence of Description 8) with:

8) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)* returns an *ST_CompoundCurve* value with:

7.4.5 ST_CurveN Method

1. Rationale: Use a return value for such completion codes.

Replace the Definition with the following:

```
CREATE METHOD ST_CurveN(aposition INTEGER)
    RETURNS ST_Curve
    FOR ST_CompoundCurve
    BEGIN
        IF CARDINALITY(SELF.ST_PrivateCurves) = 0 THEN
          BEGIN
             SIGNAL SQLSTATE '01F06'
                SET MESSAGE_TEXT = 'empty geometry';
             RETURN NULL;
          END;
        END IF;
```

```
    IF aposition < 1 OR
      aposition > CARDINALITY(SELF.ST_PrivateCurves) THEN
      BEGIN
         SIGNAL SQLSTATE '01F01'
            SET MESSAGE_TEXT = 'invalid position';
         RETURN NULL;
      END;
    END IF;
    RETURN SELF.ST_PrivateCurves[aposition];
END
```

In the Description section, replace Description 2) with:

2) For the null-call method *ST_CurveN(INTEGER)*:

   Case:

   a) If the cardinality of the *ST_PrivateCurves* attribute is equal to 0 (zero), then:

      i) A completion condition is raised: *SQL/MM Spatial warning – geometry*.

      ii) Return the null value.

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivateCurves* attribute, then:

      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.

      ii) Return the null value.

   c) Otherwise, return an *ST_Curve* value at element *aposition* in the *ST_PrivateCurves* attribute of SELF.

8.1.2 ST_Area Method

1. Rationale: Correct the returned value description.

Replace Description 3) with:

3) Case:

   a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Area()* is in the linear unit of measure identified by <linear unit> squared.

   b) Otherwise, the value returned by *ST_Area()* is in an implementation-defined unit of measure.

8.1.3 ST_Perimeter Method

1. Rationale: Correct the returned value description.

Replace Description 3) with:

3) Case:

   a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Perimeter*() is in the linear unit of measure identified by <linear unit> squared.

   b) Otherwise, the value returned by *ST_Perimeter*() is in an implementation-defined unit of measure.

8.2.1 ST_CurvePolygon Type

1. Rationale: Correct constructor method definitions.

Replace the four <original method specification>s that begins with "METHOD ST_CurvePolygon" in the Definition with:

```
CONSTRUCTOR METHOD ST_CurvePolygon
    (acurve ST_Curve)
    RETURNS ST_CurvePolygon
    SELF AS RESULT
```