INTERNATIONAL STANDARD

ISO/IEC 13249-5

First edition
2001-05-01

Information technology — Database languages — SQL multimedia and application packages —

Part 5: **Still Image**

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

Partie 5: Image fixe

Reference number
ISO/IEC 13249-5:2001(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents Page

Tables

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 13249 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 13249-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ISO/IEC 13249 consists of the following parts, under the general title *Information technology — Database languages — SQL multimedia and application packages*:

— *Part 1: Framework*

— *Part 2: Full-Text*

— *Part 3: Spatial*

— *Part 4: General purpose facilities*

— *Part 5: Still Image*

— *Part 6: Data Mining*

Annexes A and B of this part of ISO/IEC 13249 are for information only.

Introduction

The purpose of this International Standard is to define multimedia and application specific types and their associated routines using the user-defined features in ISO/IEC 9075.

This document is based on the content of ISO/IEC International Standard Database Language (SQL).

The organization of this part of ISO/IEC 13249 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 13249.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 13249, constitute provisions of this part of ISO/IEC 13249.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 13249.

4) Clause 4, "Concepts", presents concepts used in the definition of this part of ISO/IEC 13249.

5) Clause 5, "Still Image Types", defines the still image user-defined types and associated routines.

6) Clause 6, "Feature Types", defines the user-defined types provided for the manipulation of still image features.

7) Clause 7, "SQL/MM Still Image Information Schema" defines the SQL/MM Still Image Information Schema.

8) Clause 8, "SQL/MM Still Image Definition Schema" defines the SQL/MM Still Image Definition Schema.

9) Clause 9, "Status Codes", defines the SQLSTATE codes used in this part of ISO/IEC 13249.

10) Clause 10, "Conformance", defines the criteria for conformance to this part of ISO/IEC 13249.

11) Annex A, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 13249 states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementor shall provide in each case.

12) Annex B, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 13249 states explicitly that the meaning or effect on the database is implementation-dependent.

In the text of this part of ISO/IEC 13249, Clauses begin a new odd-numbered page, and in Clause 5, "Still Image Types", through Clause 10, "Conformance", Subclauses begin a new page. Any resulting blank space is not significant.

Information technology — Database languages — SQL multimedia and application packages —

Part 5: Still Image

1 Scope

This part of ISO/IEC 13249:

a) introduces the Still Image part of ISO/IEC 13249,

b) gives the references necessary for this part of this part of ISO/IEC 13249,

c) defines notations and conventions specific to this part of this part of ISO/IEC 13249,

d) defines concepts specific to this part of this part of ISO/IEC 13249,

e) defines the still image user-defined types and their associated routines.

The still image user-defined types defined in this part adhere to the following:

- A still image user-defined type is generic to image data handling. It addresses the need to store, manage and retrieve information based on aspects of image data such as height, width and format and based on image features such as average color, color histogram, positional color and texture.
- A still image user-defined type does not redefine the database language SQL directly or in combination with another still image data type.

The still image user-defined types are applicable to all different image formats. However, not all functionality can be used with all known still image formats.

An implementation of this part of ISO/IEC 13249 may exist in environments that also support information and content management, decision support, data mining, and data warehousing systems.

Application areas addressed by implementations of this part of ISO/IEC 13249 include, but are not restricted to, graphics, multimedia, scientific research, and medicine.

Blank page

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 13249. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 13249 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).*

ISO/IEC 9075-4:1999, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM).*

ISO/IEC 13249-1:2000, *Information technology — Database languages — SQL multimedia and application packages — Part 1: Framework.*

ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*.

Blank page

3 Definitions, notations, and conventions

3.1 Definitions

For the purposes of this part of ISO/IEC 13249, the following definitions apply.

3.1.1 Definitions provided in Part 1

This part of ISO/IEC 13249 makes use of all terms defined in Part 1 of ISO/IEC 13249.

3.1.2 Definitions provided in Part 5

This part of ISO/IEC 13249 defines the following terms:

3.1.2.1
basic image feature
a basic image feature is an image feature that is not a composite feature

3.1.2.2
color space
a set of conventions how to represent a color value

3.1.2.3
composite feature
an image feature which consists of basic image features and their associated weights

3.1.2.4
image format
a set of conventions for storing the image data of digital images in a specific compressed or uncompressed interchange format

3.1.2.5
image feature
characteristic (other than inherent image characteristics) of the image data

3.1.2.6
inherent image characteristics
image format and particular physical characteristics of a digital image

3.1.2.7
list of weighted features
see composite feature

3.1.2.8
picture element
see sample in Subclause 3.1.4, "Definitions taken from ISO/IEC 10918-1"

3.1.2.9
raw image
a binary string that represents a certain image

3.1.2.10
similarity of images
a numerical measure obtainable by the comparison of two images; the measure is based on image features

3.1.2.11
thumbnail
a raw image which was obtained from another raw image by downsizing

3.1.3 Definitions taken from ISO/IEC 9075

This part of ISO/IEC 13249 makes use of the following terms defined in ISO/IEC 9075:

3.1.4 Definitions taken from ISO/IEC 10918-1

This part of ISO/IEC 13249 makes use of the following terms defined in ISO/IEC 10918-1:

a) columns

NOTE 1 The use of "columns" here is as defined in the JPEG standard and not as defined in the SQL standard.

b) component

c) (digital) (still) image

NOTE 2 Parentheses around the text "digital" and "still" is a convention used by ISO/IEC 10918-1 to denote that the phases "digital image", "still image", and "image" are interchangeable.

d) image data

e) interchange format

f) (number of) lines

NOTE 3 Parentheses around the text "number of" is a convention used by ISO/IEC 10918-1 to denote that the phases "number of lines" and "lines" are interchangeable.

g) sample

3.2 Notations

3.2.1 Notations provided in Part 1

The notations used in this part of ISO/IEC 13249 are defined in Part 1 of ISO/IEC 13249.

3.2.2 Notations provided in Part 5

This part of ISO/IEC 13249 uses the prefix 'SI_' for user-defined types, attributes and SQL-invoked routine names.

3.3 Conventions

The conventions used in this part of ISO/IEC 13249 are defined in Part 1 of ISO/IEC 13249.

4 Concepts

4.1 Introduction

In the context of this part of ISO/IEC 13249, digital images are effectively 2-dimensional arrays of picture elements, or samples. The internal representation of a sample in the raw image itself is image format specific. The color value of a sample in the image data might be represented by an index in a color look up table, spread over multiple color planes, multiple binary strings which represent the single components of a color, or in any other image format specific way.

An image format is a set of conventions for storing digital images in an interchange format. An image consists of the representation and organization of data that constitute the picture elements, and prescriptions about auxiliary data that control the interpretation and processing of the digital image information according to that format.

A color space, which is used to represent the color values of the samples, is either defined by the image format or described in the header information of the raw image.

An image format is referenced by *format indications*. A format indication is a character string whose format and content is implementation-defined.

A binary string that adheres to a certain image format is called a raw image.

The inherent image characteristics of a raw image consist of:

- the format of the raw image;
- the width of the raw image is the number of columns of the image data;
- the height of the raw image is the number of lines of the image data.

An image format is a format supported by an implementation (for short: a *supported format*) if the implementation is able to derive the inherent image characteristics and features from the raw image.

NOTE 4 Features are described in Subclause 4.4, "Image features".

This part of ISO/IEC 13249 defines types and routines with provisions for storing and manipulating still images. This part of ISO/IEC 13249 consists of the following parts:

- The data type *SI_StillImage*, a value of which has the following structure:
  - a digital representation of a still image;
  - format conventions used for representing that still image;
  - physical characteristics of that still image (such as its height and width).
- Methods on the data type *SI_StillImage* for:
  - constructing *SI_StillImage* values;
  - obtaining the digital representation of an image or the inherent image characteristics of the image;
  - obtaining a thumbnail from an *SI_StillImage* value.
- Feature data types that abstract from certain characteristics of the pictorial information contained in images; these data types provide facilities for:

- deriving feature values from a given image;
- constructing feature values;
- deriving metric values that characterize the content of images with respect to feature values;
- Information Schema views that provide data describing certain capabilities of an implementation of this part of ISO/IEC 13249.

A conforming implementation of this part of ISO/IEC 13249 shall be based on SQL-implementations that supports Core SQL as defined by ISO/IEC 9075. A number of provisions are made for that purpose:

- No function name overloading is used for SQL-invoked regular functions that are intended for public use;
- For every method that is intended for public use, a corresponding SQL-invoked regular function is specified that provides the same services as the associated method;
- The lengths of the names of schemata, types and SQL-invoked regular functions that are intended for public use do not exceed 18 characters. If the name of a view of the Still Image Information Schema exceeds 18 characters, an equivalent view with a short identifier is also specified.

4.2 Concepts taken from ISO/IEC 9075

The following concepts defined in ISO/IEC 9075 are used in this part of ISO/IEC 13249.

a) binary string

b) EXECUTE privilege

c) function

d) SQL-invoked regular function

4.3 Types representing digital images

4.3.1 Attributes of the SI_StillImage type

The *SI_StillImage* type is an abstraction for digital images, using the following attributes:

- The attribute *SI_content* to represent the raw image;
- The attribute *SI_contentLength* to represent the length of the raw image;
- The attribute *SI_format* to represent a format indication; it identifies the image format of the raw image;
- The attribute *SI_width* to represent the width of the raw image;
- The attribute *SI_height* to represent the height of the raw image.

4.3.2 Methods of the SI_StillImage type

The type *SI_StillImage* provides the following methods for public use:

- *SI_StillImage*: constructs an *SI_StillImage* value from a raw image;
- *SI_StillImage*: constructs an *SI_StillImage* value from a raw image and a character string representing a format indication; this method allows for user-supplied format information when *SI_StillImage* values are to be constructed from a raw image whose format is not a supported one;
- *SI_setContent*: has the same effect as the invocation of the mutator function for the *SI_content* attribute, but additionally adjusts the values of the attributes that represent the inherent image characteristics;
- *SI_changeFormat*: has the same effect as the invocation of the mutator function for the *SI_format* attribute, but additionally adjusts the value of the attribute *SI_content* and the values of the attributes which represent the inherent image characteristics. The format conversion fails if the conversion between the source image format and the target image format is not supported;
- *SI_Thumbnail*: obtains a thumbnail from an *SI_StillImage* value;
- *SI_content*: returns the representation of the raw image;
- *SI_contentLength*: returns the length in bytes of the representation of the raw image;
- *SI_format*: returns the format indication of the image;
- *SI_height*: returns the number of lines of the image;
- *SI_width*: returns the number of columns of the image.

4.4 Image features

Image features (for short: *features*) are used to characterize the pictorial information of an image by means other than inherent image characteristics. This part of ISO/IEC 13249 supports four basic features and one composite feature. The *basic features* are:

- *Average color feature*: this feature characterizes an image by its average color;
- *Color histogram feature*: this feature characterizes an image by the relative frequencies of the colors exhibited by the samples of the raw image;
- *Positional color feature*: let an image be divided into n by m rectangles; the positional color feature characterizes that image by the n by m average colors of these rectangles;
- *Texture feature*: this feature characterizes an image by the size of repeating items (*coarseness*), brightness variations (*contrast*), and the predominant direction (*directionality*).

All basic features can be derived from images. In addition, the two basic features average color feature and color histogram feature can be constructed by means of numerical values.

The composite feature is a list of up to 4 basic features, each of a different feature type. All the basic features in the composite feature are associated with a feature weight.

Features are represented by feature types. Values of those types are used for obtaining a quantitative measure for the similarity between two images represented as *SI_StillImage* values, say, I_1 and I_2. If F_2 is some feature that characterizes the image I_2, then a similarity measure for the images I_1 and I_2 is obtained from F_2 by a method *SI_Score* that takes I_1 as its parameter. For a given pair of images, the obtained similarity depends on the kind of feature used for comparison; the exact relationship is implementation-dependent.

4.4.1 Feature types

The basic features are represented by the following feature types:

- Average color feature: *SI_AverageColor*;
- Color histogram feature: *SI_ColorHistogram*;
- Positional color feature: *SI_PositionalColor*;
- Texture feature: *SI_Texture*.

The composite feature is represented by the feature type *SI_FeatureList*.

For all basic features methods and functions are provided that derive the corresponding feature value from an *SI_StillImage* value. The functions are:

- Average color feature: *SI_fndAverageColor*;
- Color histogram feature: *SI_findColorHstgrm*;
- Positional color feature: *SI_findPositColor*;
- Texture feature: *SI_findTexture*.

SI_AverageColor and *SI_ColorHistogram* values can also be obtained by methods *SI_AverageColor* and *SI_ColorHistogram*, respectively. The parameter of the first method is a color value that is used to represent the intended average color. The second method takes a first color value and a first frequency and returns an initial color histogram. This initial color histogram can be extended using the method *SI_Append*.

SI_FeatureList values must be constructed from basic feature values and associated weights.

The observer and mutator functions of the basic feature types are not intended for public use. Thus, there are no GRANT statements granting EXECUTE privilege on these functions.

4.4.2 Assessing the similarity of images

Every feature type has a method *SI_Score*. This method can be used for obtaining numerical values that measure the similarity between two images. To that end, a distance function is used for this measurement. The returned numerical value for the distance indicates the difference between a given feature value and a still image value. Let F_1 be some feature value that characterizes an *SI_StillImage* values I_1. Then

```
F1.SI_Score(I2)
```

returns a measure for the similarity of the *SI_StillImage* value I_2 to the feature value F_1.

4.5 Complementary SQL-invoked regular functions

To ease conformance for implementation of this part of ISO/IEC 13249, each method intended for public use is complemented by an SQL-invoked regular function.

For each such method, the type of specified method, the method name, parameter types (if any), and the name of the corresponding SQL-invoked regular function is listed in Table 1 – Method and function name correspondences. Since the names of these functions are unique, their parameter types are not given.

Table 1 – Method and function name correspondences

Type Name	Method Name	Parameter Types (if any)	Function Name
SI_StillImage	SI_StillImage	BINARY LARGE OBJECT	SI_mkStillImage1
SI_StillImage	SI_StillImage	BINARY LARGE OBJECT, CHARACTER VARYING	SI_mkStillImage2
SI_StillImage	SI_setContent	BINARY LARGE OBJECT	SI_chgContent
SI_StillImage	SI_changeFormat	CHARACTER VARYING	SI_convertFormat
SI_StillImage	SI_content		SI_getContent
SI_StillImage	SI_contentLength		SI_getContentLngth
SI_StillImage	SI_format		SI_getFormat
SI_StillImage	SI_height		SI_getHeight
SI_StillImage	SI_width		SI_getWidth
SI_StillImage	SI_Thumbnail		SI_getThumbnail
SI_StillImage	SI_Thumbnail	INTEGER, INTEGER	SI_getSizedThmbnl
SI_AverageColor	SI_AverageColor	SI_StillImage	SI_fndAverageColor
SI_AverageColor	SI_AverageColor	SI_Color	SI_mkAverageColor
SI_AverageColor	SI_Score	SI_StillImage	SI_ScoreByAvrgClr
SI_ColorHistogram	SI_ColorHistogram	SI_StillImage	SI_findColorHstgrm
SI_ColorHistogram	SI_ColorHistogram	SI_Color, DOUBLE PRECISION	SI_mkColorHistogrm
SI_ColorHistogram	SI_ColorHistogram	SI_Color ARRAY, DOUBLE PRECISION ARRAY	SI_arrayClrHstgrm
SI_ColorHistogram	SI_Append	SI_Color, DOUBLE PRECISION	SI_appendClrHstgrm
SI_ColorHistogram	SI_Score	SI_StillImage	SI_ScoreByClrHstgr
SI_PositionalColor	SI_PositionalColor	SI_StillImage	SI_findPositColor
SI_PositionalColor	SI_Score	SI_StillImage	SI_ScoreByPositClr
SI_Texture	SI_Texture	SI_StillImage	SI_findTexture
SI_Texture	SI_Score	SI_StillImage	SI_ScoreByTexture

Type Name	Method Name	Parameter Types (if any)	Function Name
SI_FeatureList	SI_FeatureList	SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION	SI_mkFeatureList
SI_FeatureList	SI_setFeature	SI_AverageColor, DOUBLE PRECISION	SI_setAvgClrFtrW
SI_FeatureList	SI_setFeature	SI_ColorHistogram, DOUBLE PRECISION	SI_setClrHstgrFtrW
SI_FeatureList	SI_setFeature	SI_PositionalColor, DOUBLE PRECISION	SI_setPstnlClrFtrW
SI_FeatureList	SI_setFeature	SI_Texture, DOUBLE PRECISION	SI_setTextureFtrW
SI_FeatureList	SI_AvgClrFtr		SI_getAvgClrFtr
SI_FeatureList	SI_AvgClrFtrWght		SI_getAvgClrFtrW
SI_FeatureList	SI_ClrHstgrFtr		SI_getClrHstgrFtr
SI_FeatureList	SI_ClrHstgrFtrWght		SI_getClrHstgrFtrW
SI_FeatureList	SI_PstnlClrFtr		SI_getPstnlClrFtr
SI_FeatureList	SI_PstnlClrFtrWght		SI_getPstnlClrFtrW
SI_FeatureList	SI_TextureFtr		SI_getTextureFtr
SI_FeatureList	SI_TextureFtrWght		SI_getTextureFtrW
SI_FeatureList	SI_Score	SI_StillImage	SI_ScoreByFtrList
SI_Color	SI_RGBColor	INTEGER, INTEGER, INTEGER	SI_mkRGBColor

4.6 Auxiliary type SI_Color

Color values are encapsulated by the type *SI_Color*. The implementation shall provide constructor methods to obtain *SI_Color* values. Each constructor method is provided for a specific color space. Each function takes parameters that represent the intended color value in this color space. A constructor method *SI_RGBColor,* for the RGB color space, is in this part of ISO/IEC 13249. Constructor methods for other color spaces are implementation defined.

4.7 The Still Image Information Schema

This part of ISO/IEC 13249 prescribes an Information Schema called SI_INFORMTN_SCHEMA. It contains views for the following purposes:

- a view SI_IMAGE_FORMATS that lists the format indications of the supported image formats;
- a view SI_IMAGE_FORMAT_CONVERSIONS that lists pairs of format indications of image formats for which format conversions are supported;
- a view SI_IMAGE_FORMAT_FEATURES that lists pairs of format indications and feature indications; images of the indicated image format support the extraction of the indicated feature and scoring with respect to that feature;
- a view SI_THUMBNAIL_FORMATS that lists the format indications of image formats from which thumbnails can be derived;
- a view SI_VALUES that lists implementation-defined values.

5 Still Image Types

The types in this family provide for the storage and retrieval of still image values.

5.1 SI_StillImage Types and Routines

5.1.1 SI_StillImage Type

Purpose

The *SI_StillImage* type provides the definition of a still image type.

Definition

```
CREATE TYPE SI_StillImage
   AS (
      SI_content BINARY LARGE OBJECT(SI_MaxContentLength),
      SI_contentLength INTEGER,
      SI_format CHARACTER VARYING(SI_MaxFormatLength),
      SI_height INTEGER,
      SI_width INTEGER
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_StillImage
      (content BINARY LARGE OBJECT(SI_MaxContentLength))
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_StillImage
      (content BINARY LARGE OBJECT(SI_MaxContentLength),
       explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setContent
      (content BINARY LARGE OBJECT(SI_MaxContentLength))
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_changeFormat
      (targetFormat CHARACTER VARYING(SI_MaxFormatLength))
      RETURNS SI_StillImage
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,
```

5.1.1 SI_StillImage Type

```
METHOD SI_Thumbnail()
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD SI_Thumbnail
   (height INTEGER,
    width INTEGER)
   RETURNS SI_StillImage
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

Description

1) The *SI_StillImage* type provides for public use:

   a) a method *SI_StillImage(BINARY LARGE OBJECT)*,

   b) a method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING)*,

   c) a method *SI_setContent(BINARY LARGE OBJECT)*,

   d) a method *SI_changeFormat(CHARACTER VARYING)*,

   e) a method *SI_Thumbnail()*,

   f) a method *SI_Thumbnail(INTEGER, INTEGER)*,

   g) a function *SI_chgContent(SI_StillImage, BINARY LARGE OBJECT)*,

   h) a function *SI_convertFormat(SI_StillImage, CHARACTER VARYING)*,

   i) a function *SI_getThumbnail(SI_StillImage)*,

   j) a function *SI_getSizedThmbnl(SI_StillImage, INTEGER, INTEGER)*.

   k) a function *SI_mkStillImage1(BINARY LARGE OBJECT)*,

   l) a function *SI_mkStillImage2(BINARY LARGE OBJECT, CHARACTER VARYING)*,

2) The attributes *SI_content, SI_contentLength, SI_format, SI_height,* and *SI_width* are read-only. There are no GRANT statements granting EXECUTE privilege on the mutator functions for *SI_content, SI_contentLength, SI_format, SI_height*, and *SI_width*.

5.1.2 SI_StillImage Methods

Purpose

Return a specified *SI_StillImage* value.

Definition

```
CREATE CONSTRUCTOR METHOD SI_StillImage
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   FOR SI_StillImage
   RETURN SELF.
      SI_content(content).
      SI_contentLength(LENGTH(content)).
      SI_format(SI_format(content)).
      SI_height(SI_height(content)).
      SI_width(SI_width(content))

CREATE CONSTRUCTOR METHOD SI_StillImage
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE FormatError CONDITION FOR SQLSTATE '2FF10';
      DECLARE iFormat CHARACTER VARYING(SI_MaxFormatLength);
      DECLARE eFormat CHARACTER VARYING(SI_MaxFormatLength);

      SET iFormat = TRIM(BOTH ' ' FROM SI_format(content));
      SET eFormat = TRIM(BOTH ' ' FROM explicitFormat);
      IF eFormat IS NULL OR
         NOT (
            SI_supportedFormat(eFormat) = 1 AND
            iFormat = eFormat OR
            NOT SI_supportedFormat(eFormat) = 1 AND
            iFormat IS NULL) THEN
         SIGNAL FormatError
            SET MESSAGE_TEXT = 'illegal image format specification';
      END IF;
      RETURN SELF.
         SI_content(content).
         SI_contentLength(LENGTH(content)).
         SI_format(explicitFormat).
         SI_height(SI_height(content)).
         SI_width(SI_width(content));
   END
```

Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

Description

1) The method *SI_StillImage(BINARY LARGE OBJECT)* takes the following input parameter:

a) a BINARY LARGE OBJECT value *content*.

2) The method *SI_StillImage*(*BINARY LARGE OBJECT*, *CHARACTER VARYING*) takes the following input parameters:

a) a BINARY LARGE OBJECT value *content*,

b) a CHARACTER VARYING value *explicitFormat*.

3) If any of the following is true for an invocation of the method *SI_StillImage(BINARY LARGE OBJECT, CHARACTER VARYING)*, an exception condition is raised: *SQL/MM Still Image exception – illegal image format specification*.

a) *explicitFormat* is the null value.

b) At least one of the following is false:

i) *explicitFormat* indicates a supported image format, and *explicitFormat* is equivalent to the format derived from *content*.

ii) *explicitFormat* indicates an unsupported image format, and the image format derived from *content* is the null value (i.e. no supported image format can be derived from *content*).

5.1.3 SI_setContent Method

Purpose

Update the *SI_StillImage* content.

Definition

```
CREATE METHOD SI_setContent
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE NullInstance CONDITION FOR SQLSTATE '2202D';
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF01';

      IF SELF IS NULL THEN
         SIGNAL NullInstance
            SET MESSAGE_TEXT = 'null image value';
      END IF;
      IF SI_format(content) <> SELF.SI_format THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect image format';
      END IF;
      RETURN SELF.
         SI_content(content).
         SI_contentLength(LENGTH(content)).
         SI_height(SI_height(content)).
         SI_width(SI_width(content));
   END
```

Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

Description

1) The method *SI_setContent(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content*.

5.1.4 SI_changeFormat Method

Purpose

Convert the format of an *SI_StillImage* value and adjust affected attributes.

Definition

```
CREATE METHOD SI_changeFormat
   (targetFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE UnsupportedConversion CONDITION FOR SQLSTATE '2FF11';
      DECLARE localContent BINARY LARGE OBJECT(SI_MaxContentLength);

      IF NOT SI_supportedConversion (SELF.SI_format, targetFormat) = 1
      THEN
         SIGNAL UnsupportedConversion
            SET MESSAGE_TEXT =
               'unsupported image format conversion specified';
      END IF;
      IF TRIM(BOTH ' ' FROM targetFormat) =
            TRIM(BOTH ' ' FROM SELF.SI_format) THEN
         RETURN SELF;
      ELSE
         SET localContent =
            SI_convert(SELF.SI_content, SELF.SI_format, targetFormat);
         RETURN SELF.
            SI_content(localContent).
            SI_contentLength(LENGTH(localContent)).
            SI_format(targetFormat).
            SI_height(SI_height(localContent)).
            SI_width(SI_width(localContent));
      END IF;
   END
```

Definitional Rules

1) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

Description

1) The method *SI_changeFormat(CHARACTER VARYING)* takes the following input parameter:

   a) a CHARACTER VARYING value *targetFormat*.

2) Case:

   a) If the implementation does not support the conversion between the formats *SELF.SI_format* and *targetFormat*, then an exception condition is raised: *SQL/MM Still Image exception – unsupported image format conversion specified*.

   b) If *targetFormat* and *SELF.SI_format* are equivalent, *SELF* is returned unchanged.

   c) Otherwise, *SELF.SI_content* is converted to *targetFormat*, and the values of *SELF.SI_contentLength*, *SELF.SI_format*, *SELF.SI_height*, and *SELF.SI_width* are adjusted to reflect the new value of *SELF.SI_content*.

5.1.5 SI_Thumbnail Methods

Purpose

The *SI_Thumbnail* method derives a thumbnail from an *SI_StillImage* value.

Definition

```
CREATE METHOD SI_Thumbnail()
   RETURNS SI_StillImage
   FOR SI_StillImage
   RETURN NEW SI_Thumbnail(SI_ThumbnailHeight, SI_ThumbnailWidth)

CREATE METHOD SI_Thumbnail
   (height INTEGER,
    width INTEGER)
   RETURNS SI_StillImage
   FOR SI_StillImage
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF12';

      IF height > SELF.SI_height OR
         width > SELF.SI_height OR
         NOT SI_supportedThumbnail(SELF.SI_format) THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'illegal specification for thumbnail generation'
      END IF;
      RETURN SI_deriveThumbnail(SELF, height, width);
   END
```

Definitional Rules

1) *SI_ThumbnailHeight* is the implementation-dependent height for the resulting thumbnail.

2) *SI_ThumbnailWidth* is the implementation-dependent width for the resulting thumbnail.

Description

1) The method *SI_Thumbnail()*.

   a) This method takes no input parameters.

   b) The values for the attributes *SI_height* and *SI_width* for the resulting thumbnail are implementation-dependent.

2) The method *SI_Thumbnail(INTEGER, INTEGER)*.

   a) takes the following input parameters:

      i) an INTEGER value *height*,

      ii) an INTEGER value *width*.

   b) If at least one of the parameters *height* and *width* is larger than the values *SELF.SI_height* and *SELF.SI_width*, respectively, or if *SELF.SI_format* indicates that the derivation of a thumbnail from images with this format is not supported, then an exception condition is raised: *SQL/MM Still Image exception – illegal specification for thumbnail generation*.

5.1.6 Functions Complementing SI_StillImage Methods

Purpose

Return a specified *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_mkStillImage1
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN NEW SI_StillImage(content)

CREATE FUNCTION SI_mkStillImage2
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    explicitFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN NEW SI_StillImage(content, explicitFormat)
```

Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

Description

1) The function *SI_mkStillImage1*(*BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *content*.

2) The function *SI_mkStillImage2*(*BINARY LARGE OBJECT*, *CHARACTER VARYING*) takes the following input parameters:

   a) a BINARY LARGE OBJECT value *content*,

   b) a CHARACTER VARYING value *explicitFormat*,

5.1.7 SI_chgContent Function

Purpose

Update the *SI_StillImage* content.

Definition

```
CREATE FUNCTION SI_chgContent
   (image SI_StillImage,
    content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_setContent(content)
```

Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

Description

1) The function *SI_chgContent(SI_StillImage, BINARY LARGE OBJECT)* takes the following input parameters:

   a) an *SI_StillImage* value *image*,

   b) a BINARY LARGE OBJECT value *content*.

5.1.8 SI_convertFormat Function

Purpose

Convert the format of an *SI_StillImage* value and adjust affected attributes.

Definition

```
CREATE FUNCTION SI_convertFormat
   (image SI_StillImage,
    targetFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_changeFormat(targetFormat)
```

Definitional Rules

1) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

Description

1) The function *SI_convertFormat(SI_StillImage, CHARACTER VARYING)* takes the following input parameters:

   a) an *SI_StillImage* value *image,*

   b) a CHARACTER VARYING value *targetFormat*.

5.1.9 SI_getThumbnail Function

Purpose

The *SI_getThumbnail* derives a thumbnail from an *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_getThumbnail
   (image SI_StillImage)
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_Thumbnail()
```

Description

1) The function *SI_getThumbnail(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

5.1.10 SI_getSizedThmbnl Function

Purpose

The *SI_getSizedThmbnl* derives a thumbnail with user-supplied *height* and *width* values from an *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_getSizedThmbnl
   (image SI_StillImage,
    height INTEGER,
    width INTEGER)
   RETURNS SI_StillImage
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_Thumbnail(height, width)
```

Description

1) The function *SI_getSizedThmbnl(SI_StillImage, INTEGER, INTEGER)* takes the following input parameters:

a) an *SI_StillImage* value *image*,

b) an INTEGER value *height*,

c) an INTEGER value *width*.

5.1.11 Functions Complementing Observer Functions of Type SI_StillImage

Purpose

Obtain the value of the designated *SI_StillImage* attribute.

Definition

```
CREATE FUNCTION SI_getContent
   (image SI_StillImage)
   RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_content

CREATE FUNCTION SI_getContentLngth
   (image SI_StillImage)
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_contentLength

CREATE FUNCTION SI_getFormat
   (image SI_StillImage)
   RETURNS CHARACTER VARYING(SI_MaxFormatLength)
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_format

CREATE FUNCTION SI_getHeight
   (image SI_StillImage)
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_height

CREATE FUNCTION SI_getWidth
   (image SI_StillImage)
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN image.SI_width
```

Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

Description

1) The function *SI_getContent(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) *SI_getContent(SI_StillImage)* returns the binary representation of the attribute *SI_content* of the *SI_StillImage* value *image*.

3) The function *SI_getContentLngth(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

4) *SI_getContentLngth(SI_StillImage)* returns the length of the binary representation of the *SI_StillImage* value *image*.

5) The function *SI_getFormat(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

6) *SI_getFormat(SI_StillImage)* returns the character representation of the image format indication of the *SI_StillImage* value *image*.

7) The function *SI_getHeight(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

8) *SI_getHeight(SI_StillImage)* returns the height in samples of the *SI_StillImage* value *image*.

9) The function *SI_getWidth(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

10) *SI_getWidth(SI_StillImage)* returns the width in samples of the *SI_StillImage* value *image*.

5.1.12 Functions not intended for Public Use

Purpose

These functions are only intended to be used within the methods *SI_StillImage*, *SI_setContent*, *SI_changeFormat*, and *SI_Thumbnail*.

Definition

```
CREATE FUNCTION SI_format
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS CHARACTER VARYING(SI_MaxFormatLength)
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_height
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_width
   (content BINARY LARGE OBJECT(SI_MaxContentLength))
   RETURNS INTEGER
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END
```

5.1.12 Functions not intended for Public Use

```
CREATE FUNCTION SI_supportedFeature
   (featureName CHARACTER VARYING(SI_MaxFeatureNameLength),
    sourceImage SI_StillImage)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN
      CASE
         WHEN EXISTS(
            SELECT
               FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMAT_FEATURES
               WHERE TRIM(BOTH ' ' FROM SI_FEATURE_NAME) =
                     TRIM(BOTH ' ' FROM featureName) AND
                  TRIM(BOTH ' ' FROM SI_FORMAT) =
                     TRIM(BOTH ' ' FROM sourceImage.SI_format)
         ) THEN 1
         ELSE 0
      END

CREATE FUNCTION SI_convert
   (content BINARY LARGE OBJECT(SI_MaxContentLength),
    sourceFormat CHARACTER VARYING(SI_MaxFormatLength),
    targetFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS BINARY LARGE OBJECT(SI_MaxContentLength)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_supportedConversion
   (sourceFormat CHARACTER VARYING(SI_MaxFormatLength),
    targetFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN
      CASE
         WHEN EXISTS(
            SELECT *
               FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMAT_CONVERSIONS
               WHERE TRIM(BOTH ' ' FROM SI_SOURCE_FORMAT) =
                     TRIM(BOTH ' ' FROM sourceFormat) AND
                  TRIM(BOTH ' ' FROM SI_TARGET_FORMAT) =
                     TRIM(BOTH ' ' FROM targetFormat)
         ) THEN 1
         ELSE 0
      END
```

```
CREATE FUNCTION SI_supportedFormat
   (specifiedFormat CHARACTER VARYING(SI_MaxFormatLength))
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN
      CASE
         WHEN EXISTS(
            SELECT *
               FROM SI_INFORMTN_SCHEMA.SI_IMAGE_FORMATS
               WHERE TRIM(BOTH ' ' FROM SI_FORMAT) =
                     TRIM(BOTH ' ' FROM specifiedFormat)
         ) THEN 1
         ELSE 0
      END

CREATE FUNCTION SI_deriveThumbnail
   (image SI_StillImage,
    height INTEGER,
    width INTEGER)
   RETURNS SI_StillImage
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- !! See Description
      --
   END

CREATE FUNCTION SI_supportedThumbnail
   (sourceFormat CHARACTER VARYING[SI_MaxFormatLength])
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN
      CASE
         WHEN EXISTS(
            SELECT *
               FROM SI_INFORMTN_SCHEMA.SI_THUMBNAIL_FORMATS
               WHERE TRIM(BOTH ‘ ‘ FROM SI_FORMAT) =
                  TRIM(BOTH ‘ ‘ FROM sourceFormat)
         ) THEN 1
         ELSE 0
      END
```


Definitional Rules

1) *SI_MaxContentLength* is the implementation-defined maximum length for the binary representation of the *SI_StillImage* attribute *SI_content*.

2) *SI_MaxFeatureNameLength* is the implementation-defined maximum length for the character representation of a basic feature name.

3) *SI_MaxFormatLength* is the implementation-defined maximum length for the character representation of an image format indication.

5.1.12 Functions not intended for Public Use

Description

1) The function *SI_format(BINARY LARGE OBJECT)* determines the image format of its parameter. The result is the null value if the parameter does not contain an image format that is amongst the implementation-defined set of supported image formats. The determination of the image format includes a check of the correctness of the image data itself.

2) The function *SI_height(BINARY LARGE OBJECT)* determines and returns the height in pixels of an image.

3) The function *SI_width(BINARY LARGE OBJECT)* determines and returns the width in pixels of an image.

4) The function *SI_supportedFeature(CHARACTER VARYING, SI_StillImage)*:

   a) This function takes the following input parameters:

      i) a CHARACTER VARYING value *featureName*.

      ii) an *SI_StillImage* value *sourceImage*.

   b) If the view *SI_IMAGE_FORMAT_FEATURES* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to the value of *sourceImage.SI_format* and whose *SI_FEATURE_NAME* column value is equivalent to *featureName*, then the result of this function is 1 (one); i.e. the feature indicated by *featureName* is supported for *sourceImage*. Otherwise, the result is 0 (zero); i.e. the feature indicated by *featureName* is not supported for *sourceImage*.

5) The function *SI_convert(BINARY LARGE OBJECT, CHARACTER VARYING, CHARACTER VARYING)* converts the format of the image represented by the first parameter from the format indicated by the second parameter into the format indicated by the third parameter. The converted image is returned as a *BINARY LARGE OBJECT* value. If the format conversion fails, the function *SI_convert* raises an exception: *SQL/MM Still Image exception – fatal error during image format conversion*.

6) The function *SI_supportedConversion(CHARACTER VARYING, CHARACTER VARYING)*:

   a) This function takes the following input parameters:

      i) a CHARACTER VARYING value *sourceFormat*,

      ii) a CHARACTER VARYING value *targetFormat*.

   b) If the view *SI_IMAGE_FORMAT_CONVERSIONS* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_SOURCE_FORMAT* and *SI_TARGET_FORMAT* column values are equivalent to *sourceFormat* and *targetFormat*, respectively, then the result of this function is 1 (one); i.e., an *SI_StillImage* value whose *SI_format* value is *sourceFormat* can be converted to an *SI_StillImage* value whose *SI_format* value is *targetFormat*. Otherwise, the result is 0 (zero); i.e., that conversion is not supported.

7) The function SI_supportedFormat(CHARACTER VARYING):

   a) This function takes the following input parameter:

      i) a CHARACTER VARYING value *specifiedFormat*.

b) If the view *SI_IMAGE_FORMATS* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *specifiedFormat*, then the result of this function is 1 (one); i.e. *specifiedFormat* is a supported image format. Otherwise, the result is 0 (zero); i.e. *specifiedFormat* is not a supported image format.

8) The private function *SI_deriveThumbnail(SI_StillImage)*:

a) This function takes the following input parameters:

i) an *SI_StillImage* value *image*,

ii) an INTEGER value *height*,

iii) an INTEGER value *width*.

b) This function derives and returns a thumbnail of its parameter *image*. If a thumbnail can not be derived from *image* then the result *ret* is the NULL value. Otherwise, *ret* returns a thumbnail with the specified *height* and *width*. The format of the thumbnail *ret.SI_format* is the same as the format of the still image *image.*

9) The private function *SI_supportedThumbnail(CHARACTER VARYING)*:

a) This function takes the following input parameter:

i) a CHARACTER VARYING value *sourceFormat*.

b) If the view *SI_THUMBNAIL_FORMATS* of the information schema *SI_INFORMTN_SCHEMA* contains a row whose *SI_FORMAT* column value is equivalent to *sourceFormat*, then the result of this function is 1 (one); i.e. a thumbnail can be derived from an image whose format indication equals *sourceFormat*. Otherwise, the result is 0 (zero); i.e. a thumbnail can not be derived from an image whose format indication equals *sourceFormat*.

Blank page

6 Feature Types

The types in this family provide for the manipulation of still image features.

6.1 SI_AverageColor Type and Routines

6.1.1 SI_AverageColor Type

Purpose

Provide the definition of the feature type *SI_AverageColor* and facilities for scoring *SI_StillImage* values using values of the *SI_AverageColor* type.

Definition

```
CREATE TYPE SI_AverageColor
   AS (
      SI_AverageColorSpec SI_Color
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_AverageColor
      (sourceImage SI_StillImage)
      RETURNS SI_AverageColor
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_AverageColor
      (averageColor SI_Color)
      RETURNS SI_AverageColor
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT
```

Description

1) The *SI_AverageColor* type provides for public use:

   a) a method *SI_AverageColor(SI_StillImage)*,

   b) a method *SI_AverageColor(SI_Color)*,

   c) a method *SI_Score(SI_StillImage)*,

   d) a function *SI_fndAverageColor(SI_StillImage)*.

e) a function *SI_mkAverageColor(SI_Color)*,

f) a function *SI_ScoreByAvrgClr(SI_AverageColor, SI_StillImage)*.

2) The *SI_AverageColor* type represents color values that are interpreted as average color values using the attribute:

a) an *SI_Color value SI_AverageColorSpec*.

3) The attribute *SI_AverageColorSpec* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI_AverageColorSpec*.

6.1.2 SI_AverageColor Methods

Purpose

Return a specified *SI_AverageColor* value.

Definition

```
CREATE CONSTRUCTOR METHOD SI_AverageColor
   (sourceImage SI_StillImage)
   RETURNS SI_AverageColor
   FOR SI_AverageColor
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF06';

      IF sourceImage IS NULL OR
         sourceImage.SI_content IS NULL OR
         NOT SI_supportedFeature('SI_AverageColor', sourceImage) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'bad input image; average color feature cannot be
                  determined';
      END IF;
      --
      -- !! See Description
      --
   END

CREATE CONSTRUCTOR METHOD SI_AverageColor
   (averageColor SI_Color)
   RETURNS SI_AverageColor
   FOR SI_AverageColor
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF02';

      IF averageColor IS NULL THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'incorrect average color feature specification';
      END IF;
      RETURN SELF.
            SI_AverageColorSpec(averageColor);
   END
```

Description

1) The method *SI_AverageColor(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If the parameter *sourceImage* of the method *SI_AverageColor(SI_StillImage)* or its attribute *sourceImage.SI_content* is the null value, or if the *SI_AverageColor* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; average color feature cannot be determined*.

3) The method *SI_AverageColor(SI_StillImage)* derives an *SI_AverageColor* value from the parameter *sourceImage*. To that end, each component of all samples is summed separately and the sum of each component is divided by the number of samples to give the values of the components of the resulting color value in the feature value.

4) The method *SI_AverageColor(SI_Color)* takes the following input parameter:

   a) an *SI_Color* value *averageColor*.

5) If the parameters of the method *SI_AverageColor(SI_Color)* is the null value, then an exception condition is raised: *SQL/MM Still Image exception – incorrect average color feature specification*.

6.1.3 SI_Score Method

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_AverageColor*.

Definition

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_AverageColor
   BEGIN
      --
      -- !! See Description
      --
   END
```

Description

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the average color of *image* is characterized by the average color represented by the *SI_AverageColor* value used for scoring *image*.

   Case:

   a) If SELF or *image* or *image.SI_content* is the null value, or if the average color feature is not supported for *image,* then the null value is returned.

   b) Otherwise, the exact relationship between the values of *SI_AverageColor*, *SI_StillImage* and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

6.1.4 SI_fndAverageColor Function

Purpose

Return the *SI_AverageColor* value from an *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_fndAverageColor
   (sourceImage SI_StillImage)
   RETURNS SI_AverageColor
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH

   RETURN NEW SI_AverageColor(sourceImage)
```

Description

1) The function *SI_fndAverageColor(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

6.1.5 SI_mkAverageColor Function

Purpose

Return a specified *SI_AverageColor* value.

Definition

```
CREATE FUNCTION SI_mkAverageColor
   (averageColor SI_Color)
   RETURNS SI_AverageColor
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_AverageColor(averageColor)
```

Description

1) The function *SI_mkAverageColor(SI_Color)* takes the following input parameter:

   a) an *SI_Color* value *averageColor*.

6.1.6 SI_ScoreByAvrgClr Function

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_AverageColor* value.

Definition

```
CREATE FUNCTION SI_ScoreByAvrgClr
   (feature SI_AverageColor,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

Description

1) The function *SI_ScoreByAvrgClr(SI_AverageColor, SI_StillImage)* takes the following input parameters:

   a) an *SI_AverageColor* value *feature*,

   b) an *SI_StillImage* value *image*.

6.2 SI_ColorHistogram Type and Routines

6.2.1 SI_ColorHistogram Type

Purpose

Provide the definition of the feature type *SI_ColorHistogram* and facilities for scoring *SI_StillImage* values using values of the *SI_ColorHistogram* type.

Definition

```
CREATE TYPE SI_ColorHistogram
   AS (
      SI_ColorsList SI_Color ARRAY[SI_MaxHistogramLength],
      SI_FrequenciesList
         DOUBLE PRECISION ARRAY[SI_MaxHistogramLength]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_ColorHistogram
      (sourceImage SI_StillImage)
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_ColorHistogram
      (firstColor SI_Color,
       frequency DOUBLE PRECISION)
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   CONSTRUCTOR METHOD SI_ColorHistogram
      (colors SI_Color ARRAY[SI_MaxHistogramLength],
       frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength])
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Append
      (color SI_Color,
       frequency DOUBLE PRECISION)
      RETURNS SI_ColorHistogram
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,
```

6.2.1 SI_ColorHistogram Type

```
METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) The *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

Description

1) The *SI_ColorHistogram* type provides for public use:

   a) a method *SI_ColorHistogram(SI_StillImage)*,

   b) a method *SI_ColorHistogram(SI_Color, DOUBLE PRECISION)*,

   c) a method *SI_ColorHistogram(SI_Color ARRAY, DOUBLE PRECISION ARRAY)*,

   d) a method *SI_Append(SI_Color, DOUBLE PRECISION)*,

   e) a method *SI_Score(SI_StillImage),*

   f) a function *SI_findColorHstgrm(SI_StillImage)*,

   g) a function *SI_mkColorHistogrm(SI_Color, DOUBLE PRECISION)*,

   h) a function *SI_arrayClrHstgrm(SI_Color ARRAY, DOUBLE PRECISION ARRAY)*,

   i) a function *SI_appendClrHstgrm(SI_ColorHistogram, SI_Color, DOUBLE PRECISION)*,

   j) a function *SI_ScoreByClrHstgr(SI_ColorHistogram, SI_StillImage)*.

2) The *SI_ColorHistogram* type represents a sequence of color/frequency pairs using the attributes:

   a) an *SI_Color* ARRAY value *SI_ColorsList*,

   b) a DOUBLE PRECISION ARRAY value *SI_FrequenciesList*; the DOUBLE PRECISION values of the array range from 0 (zero) to 100. The *i*-th value in this array is the frequency of the *i*-th color value in *SI_ColorsList*. The arrays *SI_ColorsList* and *SI_FrequenciesList* have the same number of elements.

3) The attributes *SI_ColorsList* and *SI_FrequenciesList* are not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attributes *SI_ColorsList* and *SI_FrequenciesList*.

6.2.2 SI_ColorHistogram Methods

Purpose

Return a specified *SI_ColorHistogram* value.

Definition

```
CREATE CONSTRUCTOR METHOD SI_ColorHistogram
   (sourceImage SI_StillImage)
   RETURNS SI_ColorHistogram
   FOR SI_ColorHistogram
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF08';

      IF sourceImage IS NULL OR
         sourceImage.SI_content IS NULL OR
         NOT SI_supportedFeature('SI_ColorHistogram', sourceImage) = 1
      THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'bad input image; color histogram feature cannot be
                  determined';
      END IF;
      --
      -- !! See Description
      --
   END

CREATE CONSTRUCTOR METHOD SI_ColorHistogram
   (firstColor SI_Color,
    frequency DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   FOR SI_ColorHistogram

   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF03';

      IF firstColor IS NULL OR
         frequency IS NULL OR
         frequency < 0.0 OR frequency > 100.0 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'incorrect color histogram feature specification';
      END IF;
      RETURN SELF.
         SI_ColorsList(ARRAY[firstColor]).
         SI_FrequenciesList(ARRAY[frequency]);
   END
```

```
CREATE CONSTRUCTOR METHOD SI_ColorHistogram
   (colors SI_Color ARRAY[SI_MaxHistogramLength],
    frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength]
   )
   RETURNS SI_ColorHistogram
   FOR SI_ColorHistogram
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF03';
      DECLARE i INTEGER;

      IF colors IS NULL OR
         frequencies IS NULL OR
         CARDINALITY(colors) <> CARDINALITY(frequencies) THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'incorrect color histogram feature specification';
      END IF;
      SET i = 1;
      WHILE i <= CARDINALITY(frequencies) DO
         IF frequencies[i] < 0.0 OR frequencies[i] > 100.0 THEN
            SIGNAL InvalidInput
               SET MESSAGE_TEXT =
                  'incorrect color histogram feature specification';
         END IF;
         SET i = i + 1;
      END WHILE;
      RETURN SELF.
         SI_ColorsList(colors).
         SI_FrequenciesList(frequencies);
   END
```

Definitional Rules

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

Description

1) The method *SI_ColorHistogram(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If the parameter *sourceImage* of the method *SI_ColorHistogram(SI_StillImage)* or its attribute *sourceImage.SI_content* is the null value, or if the *SI_ColorHistogram* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; color histogram feature cannot be determined*.

3) The method *SI_ColorHistogram(SI_StillImage)* derives an *SI_ColorHistogram* value from the parameter *sourceImage*. To that end, the color space is divided into an implementation-dependent number N of areas A_i, each of which is represented by some color C_i. For i ranging from 1 (one) to N, let F_i be frequency values that are initially all 0 (zero). For each sample of *sourceImage*, F_i is increased by 1 (one) if the color of that sample is one of the colors of A_i. Let F be the value that is obtained by summing up all F_i values. The final result is effectively a sequence of pairs (C_i , F_i / F).

4) The method *SI_ColorHistogram(SI_Color, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_Color* value *firstColor*,

b) a DOUBLE PRECISION value *frequency*.

5) If any of the parameters of the method *SI_ColorHistogram(SI_Color, DOUBLE PRECISION)* is the null value, or if the frequency value is less than 0 (zero) or greater than 100, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color histogram feature specification*.

6) The method *SI_ColorHistogram(SI_Color ARRAY, DOUBLE PRECISION ARRAY)* takes the following input parameters:

a) an *SI_Color* ARRAY value *colors*,

b) a DOUBLE PRECISION ARRAY value *frequencies*.

7) If any of the parameters of the method *SI_ColorHistogram*(*SI_Color ARRAY*, *DOUBLE PRECISION ARRAY*) is the null value, or if a frequency value is less than 0 (zero) or greater than 100, or if the cardinalities of the arrays *colors* and *frequencies* are different, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color histogram feature specification*.

6.2.3 SI_Append Method

Purpose

Extend an *SI_ColorHistogram* value by another (*SI_Color*, DOUBLE PRECISION) pair.

Definition

```
CREATE METHOD SI_Append
   (color SI_Color,
    frequency DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   FOR SI_ColorHistogram
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF03';

      IF CARDINALITY(SELF.SI_ColorsList) = SI_MaxHistogramLength OR
         firstColor IS NULL OR
         frequency IS NULL OR
         frequency < 0.0 OR frequency > 100.0 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'incorrect color histogram feature specification';
      END IF;
      SET SELF.SI_ColorsList =
         SELF.SI_ColorsList || ARRAY[color];
      SET SELF.SI_FrequenciesList =
         SELF.SI_FrequenciesList || ARRAY[frequency];
      RETURN SELF;
   END
```

Definitional Rules

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

Description

1) The method *SI_Append(SI_Color, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_Color* value *color*,

   b) a DOUBLE PRECISION value *frequency*.

2) If the maximum number of color/frequency pairs is already reached, or if any of the parameters is the null value, or if the frequency value is less than 0 (zero) or greater than 100, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color histogram feature specification*.

6.2.4 SI_Score Method

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_ColorHistogram* value.

Definition

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_ColorHistogram
   BEGIN
      --
      -- !! See Description
      --
   END
```

Description

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the color histogram of *image* is characterized by the color histogram represented by the *SI_ColorHistogram* value used for scoring *image*.

   Case:

   a) If SELF or *image* or *image.SI_content* is the null value, or if the color histogram feature is not supported for *image*, then the null value is returned.

   b) Otherwise, the exact relationship between the values of *SI_ColorHistogram*, *SI_StillImage*, and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

6.2.5 SI_findColorHstgrm Function

Purpose

Return the *SI_ColorHistogram* value from an *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_findColorHstgrm
   (sourceImage SI_StillImage)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_ColorHistogram(sourceImage)
```

Description

1) The function *SI_findColorHstgrm(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

6.2.6 SI_mkColorHistogrm Function

Purpose

Return a specified *SI_ColorHistogram* value.

Definition

```
CREATE FUNCTION SI_mkColorHistogrm
   (firstColor SI_Color,
    frequency DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_ColorHistogram(firstColor, frequency)
```

Description

1) The function *SI_mkColorHistogrm(SI_Color, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_Color* value *firstColor*,

   b) a DOUBLE PRECISION value *frequency*.

6.2.7 SI_arrayClrHstgrm Function

Purpose

Return a specified *SI_ColorHistogram* value.

Definition

```
CREATE FUNCTION SI_arrayClrHstgrm
   (colors SI_Color ARRAY[SI_MaxHistogramLength],
    frequencies DOUBLE PRECISION ARRAY[SI_MaxHistogramLength])
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_ColorHistogram(colors, frequencies)
```

Definitional Rules

1) *SI_MaxHistogramLength* is the implementation-defined maximum number of color/frequency pairs that are admissible in an *SI_ColorHistogram* feature value.

Description

1) The function *SI_arrayClrHstgrm(SI_Color ARRAY, DOUBLE PRECISION ARRAY)* takes the following input parameters:

   a) an *SI_Color* ARRAY value *colors*,

   b) a DOUBLE PRECISION ARRAY value *frequencies*.

6.2.8 SI_appendClrHstgrm Function

Purpose

Extend an *SI_ColorHistogram* value by another (*SI_Color*, DOUBLE PRECISION) pair.

Definition

```
CREATE FUNCTION SI_appendClrHstgrm
   (colorHistogram SI_ColorHistogram,
    color SI_Color,
    frequency  DOUBLE PRECISION)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN colorHistgram.SI_Append(color, frequency)
```

Description

1) The function *SI_appendClrHstgrm(SI_ColorHistogram, SI_Color, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_ColorHistogram* value *colorHistogram*,

   b) an *SI_Color* value *color*,

   c) a DOUBLE PRECISION value *frequency*.

6.2.9 SI_ScoreByClrHstgr Function

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_ColorHistogram* value.

Definition

```
CREATE FUNCTION SI_ScoreByClrHstgr
   (feature SI_ColorHistogram,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

Description

1) The function *SI_ScoreByClrHstgr(SI_ColorHistogram, SI_StillImage)* takes the following input parameters:

   a) an *SI_ColorHistogram* value *feature*,

   b) an *SI_StillImage* value *image*.

6.3 SI_PositionalColor Type and Routines

6.3.1 SI_PositionalColor Type

Purpose

Provide the definition of the feature type *SI_PositionalColor* and facilities for scoring *SI_StillImage* values using values of the *SI_PositionalColor* type.

Definition

```
CREATE TYPE SI_PositionalColor
   AS (
      SI_AverageColorPositions
         SI_Color ARRAY[SI_NumberSections]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_PositionalColor
      (sourceImage SI_StillImage)
      RETURNS SI_PositionalColor
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) The *SI_NumberSections* is the implementation-defined number of *SI_AverageColor* values that are represented by an *SI_PositionalColor* value.

Description

1) The *SI_PositionalColor* type provides for public use:

   a) a method *SI_PositionalColor(SI_StillImage)*,

   b) a method *SI_Score(SI_StillImage)*,

   c) a function *SI_findPositColor(SI_StillImage)*,

   d) a function *SI_ScoreByPositClr(SI_PositionalColor, SI_StillImage)*.

2) The *SI_PositionalColor* type represents lists of *SI_Color* values using the attribute:

   a) an *SI_Color ARRAY value SI_AverageColorPositions*.

3) The attribute SI_AverageColorPositions is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI_AverageColorPositions*.

6.3.2 SI_PositionalColor Method

Purpose

Return the *SI_PositionalColor* value from an *SI_StillImage* value.

Definition

```
CREATE CONSTRUCTOR METHOD SI_PositionalColor
   (sourceImage SI_StillImage)
   RETURNS SI_PositionalColor
   FOR SI_PositionalColor
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF07';

      IF sourceImage IS NULL OR
         sourceImage.SI_content IS NULL OR
         NOT
            SI_supportedFeature('SI_PositionalColor', sourceImage) = 1
            THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'bad input image; positional color feature cannot be
                  determined';
      END IF;
      --
      -- !! See Description
      --
   END
```

Description

1) The method *SI_PositionalColor(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If the parameter *sourceImage* of the method *SI_PositionalColor(SI_StillImage)* or its attribute *sourceImage.SI_content* is the null value, or if the *SI_PositionalColor* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; positional color feature cannot be determined*.

3) The method *SI_PositionalColor(SI_StillImage)* derives an *SI_PositionalColor* value from the parameter *sourceImage*. To that end, *sourceImage* is effectively divided into *n* by *m* rectangles, and for each rectangle, the average color value is determined. The array, thus computed, of color values which represent average colors is the *SI_AverageColorPositions* value of the returned *SI_PositionalColor* value. Further details on the relationship between *sourceImage* and the resulting *SI_PositionalColor* value, such as the values *n* and *m*, are implementation-dependent.

   NOTE 5 The color values representing the average color for each area are determined as described in Description Rule 3) in Subclause 6.1.2, "SI_AverageColor Methods", for the method SI_AverageColor(SI_StillImage).

6.3.3 SI_Score Method

Purpose

Determine and return the score of an SI_StillImage value to a given *SI_PositionalColor* value.

Definition

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_PositionalColor
   BEGIN
      --
      -- !! See Description
      --
   END
```

Description

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). For scoring an image, that image is effectively divided into *n* by *m* rectangles, such that the product of *n* and *m* equals *SI_NumberSections*. The lower the returned value, the better the *n* by *m* average colors of *image* are characterized by the average colors represented by the *SI_PositionalColor* value used for scoring *image*.

   NOTE 6 The way in which *image* is divided into *SI_NumberSections* of rectangles is implementation-dependent, as well *n* and *m* itself. However, the division shall be performed in the same fashion for the *SI_Score* method and the method *SI_PositionalColor*(*SI_StillImage*).

   Case:

   a) If SELF or *image* or *image.SI_content* is the null value, or if the positional color feature is not supported for *image*, then the null value is returned.

   b) Otherwise, the exact relationship between the values of *SI_PositionalColor*, *SI_StillImage* and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

6.3.4 SI_findPositColor Function

Purpose

Return the *SI_PositionalColor* value from an *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_findPositColor
   (sourceImage SI_StillImage)
   RETURNS SI_PositionalColor
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_PositionalColor(sourceImage)
```

Description

1) The function *SI_findPositColor(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

6.3.5 SI_ScoreByPositClr Function

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_PositionalColor* value.

Definition

```
CREATE FUNCTION SI_ScoreByPositClr
   (feature SI_PositionalColor,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

Description

1) The function *SI_ScoreByPositClr(SI_PositionalColor, SI_StillImage)* takes the following input parameters:

   a) an SI_PositionalColor value feature,

   b) an *SI_StillImage* value *image*.

6.4 SI_Texture Type and Routines

6.4.1 SI_Texture Type

Purpose

Provide the definition of the feature type *SI_Texture* and facilities for scoring *SI_StillImage* values using values of the *SI_Texture* type.

Definition

```
CREATE TYPE SI_Texture
   AS (
      SI_TextureEncoding CHARACTER VARYING(SI_MaxTextureLength)
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_Texture
      (sourceImage SI_StillImage)
      RETURNS SI_Texture
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_Score
      (image SI_StillImage)
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) The *SI_MaxTextureLength* is the implementation-defined number of bytes needed for the implementation-defined encoded representation of an *SI_Texture*.

Description

1) The *SI_Texture* type provides for public use:

   a) a method *SI_Texture(SI_StillImage)*,

   b) a method *SI_Score(SI_StillImage)*,

   c) a function *SI_findTexture(SI_StillImage)*,

   d) a function *SI_ScoreByTexture(feature SI_Texture, image SI_StillImage)*.

2) The *SI_Texture* type provides for the representation of image textures using the attribute:

   a) a CHARACTER VARYING value *SI_TextureEncoding*, the values of which represents image texture characteristics such as coarseness, contrast, and directionality in an implementation-dependent fashion.

3) The attribute *SI_TextureEncoding* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the attribute *SI_TextureEncoding*.

6.4.2 SI_Texture Method

Purpose

Return the *SI_Texture* value from an *SI_StillImage* value.

Definition

```
CREATE CONSTRUCTOR METHOD SI_Texture
   (sourceImage SI_StillImage)
   RETURNS SI_Texture
   FOR SI_Texture
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF09';

      IF sourceImage IS NULL OR
         sourceImage.SI_content IS NULL OR
         NOT SI_supportedFeature('SI_Texture', sourceImage) = 1 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT =
               'bad input image; texture feature cannot be determined';
      END IF;
      --
      -- !! See Description
      --
   END
```

Definition

1) The method *SI_Texture(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

2) If the parameter *sourceImage* of the method *SI_Texture(SI_StillImage)* or its attribute *sourceImage.SI_content* is the null value, or if the *SI_Texture* feature is not defined for *sourceImage*, then an exception condition is raised: *SQL/MM Still Image exception – bad input image; texture feature cannot be determined*.

3) The method *SI_Texture(SI_StillImage)* derives an *SI_Texture* value from the parameter *sourceImage*. The relationship between *sourceImage* and the resulting *SI_Texture* value is implementation-dependent.

6.4.3 SI_Score Method

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_Texture* value.

Definition

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_Texture
   BEGIN
      --
      -- !! See Description
      --
   END
```

Description

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better the texture of *image* is characterized by the *SI_Texture* value used for scoring that *image*.

   Case:

   a) If SELF or *image* or *image.SI_content* is the null value, or if the texture feature is not supported for *image*, then the null value is returned.

   b) Otherwise, the exact relationship between the values of *SI_Texture*, *SI_StillImage* and the result of *SI_Score(SI_StillImage)* is implementation-dependent.

6.4.4 SI_findTexture Function

Purpose

Return the *SI_Texture* value from an *SI_StillImage* value.

Definition

```
CREATE FUNCTION SI_findTexture
   (sourceImage SI_StillImage)
   RETURNS SI_Texture
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_Texture(sourceImage)
```

Description

1) The function *SI_findTexture(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *sourceImage*.

6.4.5 SI_ScoreByTexture Function

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_Texture* value.

Definition

```
CREATE FUNCTION SI_ScoreByTexture
   (feature SI_Texture,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

Description

1) The function *SI_ScoreByTexture(SI_Texture, SI_StillImage)* takes the following input parameters:

   a) an *SI_Texture* value *feature*,

   b) an *SI_StillImage* value *image*.

6.5 SI_FeatureList Type and Routines

6.5.1 SI_FeatureList Type

Purpose

Provide the definition of the type *SI_FeatureList* and facilities for scoring *SI_StillImage* values using values of the *SI_FeatureList* type.

Definition

```
CREATE TYPE SI_FeatureList
   AS (
      SI_AvgClrFtr SI_AverageColor,
      SI_AvgClrFtrWght DOUBLE PRECISION DEFAULT 0.0,
      SI_ClrHstgrFtr SI_ColorHistogram,
      SI_ClrHstgrFtrWght DOUBLE PRECISION DEFAULT 0.0,
      SI_PstnlClrFtr SI_PositionalColor,
      SI_PstnlClrFtrWght DOUBLE PRECISION DEFAULT 0.0,
      SI_TextureFtr SI_Texture,
      SI_TextureFtrWght DOUBLE PRECISION DEFAULT 0.0
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD SI_FeatureList
      (averageColorFeature SI_AverageColor,
       averageColorFeatureWeight DOUBLE PRECISION,
       colorHistogramFeature SI_ColorHistogram,
       colorHistogramFeature DOUBLE PRECISION,
       positionalColorFeature SI_PositionalColor,
       positionalColorFeatureWeight DOUBLE PRECISION,
       textureFeature SI_Texture,
       textureFeatureWeight DOUBLE PRECISION)
      RETURNS SI_FeatureList
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setFeature
      (averageColorFeature SI_AverageColor,
       averageColorFeatureWeight DOUBLE PRECISION)
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,

   METHOD SI_setFeature
      (colorHistogramFeature SI_ColorHistogram,
       colorHistogramFeatureWeight DOUBLE PRECISION)
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,
```

```
METHOD SI_setFeature
   (positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION)
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD SI_setFeature
   (textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
```

Description

1) The *SI_FeatureList* type provides for public use:

   a) a method *SI_FeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)*,

   b) a method *SI_setFeature(SI_AverageColor, DOUBLE PRECISION)*,

   c) a method *SI_setFeature(SI_ColorHistogram, DOUBLE PRECISION)*,

   d) a method *SI_setFeature(SI_PositionalColor, DOUBLE PRECISION)*,

   e) a method *SI_setFeature(SI_Texture, DOUBLE PRECISION)*,

   f) a method *SI_Score(SI_StillImage)*,

   g) a function *SI_mkFeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)*,

   h) a function *SI_setAvgClrFtrW(SI_FeatureList, SI_AverageColor, DOUBLE PRECISION)*,

   i) a function *SI_setClrHstgrFtrW(SI_FeatureList, SI_ColorHistogram, DOUBLE PRECISION)*,

   j) a function *SI_setPstnlClrFtrW(SI_FeatureList, SI_PositionalColor, DOUBLE PRECISION)*,

   k) a function *SI_setTextureFtrW(SI_FeatureList, SI_Texture, DOUBLE PRECISION)*,

   l) a function *SI_ScoreByFtrList(SI_FeatureList, SI_StillImage)*.

2) The *SI_FeatureList* type represents a list of weighted features using the attributes:

   a) an *SI_AverageColor* value *SI_AvgClrFtr*,

   b) a DOUBLE PRECISION value *SI_AvgClrFtrWght*,

   c) an *SI_ColorHistogram* value *SI_ClrHstgrFtr*,

   d) a DOUBLE PRECISION value *SI_ClrHstgrFtrWght*,

   e) an *SI_PositionalColor* value *SI_PstnlClrFtr*,

   f) a DOUBLE PRECISION value *SI_PstnlClrFtrWght*,

   g) an *SI_Texture* value *SI_TextureFtr*,

   h) a DOUBLE PRECISION value *SI_TextureFtrWght*.

3) There are no GRANT statements granting EXECUTE privilege on the mutator functions for the attributes *SI_AvgClrFtr*, *SI_AvgClrFtrWght*, *SI_ClrHstgrFtr*, *SI_ClrHstgrFtrWght*, *SI_PstnlClrFtr*, *SI_PstnlClrFtrWght*, *SI_TextureFtr*, and *SI_TextureFtrWght*.

6.5.2 SI_FeatureList Method

Purpose

Return a specified *SI_FeatureList* value.

Definition

```
CREATE CONSTRUCTOR METHOD SI_FeatureList
   (averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION,
    colorHistogramFeature SI_ColorHistogram,
    colorHistogramFeatureWeight DOUBLE PRECISION,
    positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION,
    textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF (averageColorFeatureWeight IS NULL OR
          averageColorFeatureWeight < 0.0) OR
         (colorHistogramFeatureWeight IS NULL OR
          colorHistogramFeatureWeight < 0.0) OR
         (positionalColorFeatureWeight IS NULL OR
          positionalColorFeatureWeight < 0.0) OR
         (textureFeatureWeight IS NULL OR
          textureFeatureWeight < 0.0) THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_AvgClrFtr(averageColorFeature).
         SI_AvgClrFtrWght(
            CASE
               WHEN averageColorFeature IS NULL THEN 0.0
               ELSE averageColorFeatureWeight
            END).
         SI_ClrHstgrFtr(colorHistogramFeature).
         SI_ClrHstgrFtrWght(
            CASE
               WHEN colorHistogramFeature IS NULL THEN 0.0
               ELSE colorHistogramFeatureWeight
            END).
         SI_PstnlClrFtr(positionalColorFeature).
         SI_PstnlClrFtrWght(
            CASE
               WHEN positionalColorFeature IS NULL THEN 0.0
               ELSE positionalColorFeatureWeight
            END).
         SI_TextureFtr(textureFeature).
         SI_TextureFtrWght(
            CASE
               WHEN textureFeature IS NULL THEN 0.0
               ELSE textureFeatureWeight
            END);
   END
```


Description

1) The method *SI_FeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_AverageColor* value *averageColorFeature*,

   b) a DOUBLE PRECISION value *averageColorFeatureWeight*,

   c) an *SI_ColorHistogram* value *colorHistogramFeature*,

   d) a DOUBLE PRECISION value *colorHistogramFeatureWeight*,

   e) an *SI_PositionalColor* value *positionalColorFeature*,

   f) a DOUBLE PRECISION value *positionalColorFeatureWeight*,

   g) an *SI_Texture* value *textureFeature*,

   h) a DOUBLE PRECISION value *textureFeatureWeight*.

2) If any of the parameters *averageColorFeatureWeight, colorHistogramFeatureWeight, positionalColorFeatureWeight* or *textureFeatureWeight* is the null value or less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

3) If any of the parameters *averageColorFeature*, *colorHistogramFeature*, *positionalColorFeature*, or *textureFeature* is the null value, then the attribute value of *SI_AvgClrFtrWght*, *SI_ClrHstgrFtrWght*, *SI_PstnlClrFtrWght*, or *SI_TextureFtrWght*, respectively, is set to 0 (zero), disregarding the value of the respective parameter *averageColorFeatureWeight*, *colorHistogramFeatureWeight*, *positionalColorFeatureWeight*, or *textureFeatureWeight*.

6.5.3 SI_setFeature Methods

Purpose

Modify a designated feature attribute and the corresponding weight attribute of an *SI_FeatureList* value.

Definition

```
CREATE METHOD SI_setFeature
   (averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF averageColorFeatureWeight IS NULL OR
         averageColorFeatureWeight < 0.0 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_AvgClrFtr(averageColorFeature).
         SI_AvgClrFtrWght(
            CASE
               WHEN averageColorFeature IS NULL THEN 0.0
               ELSE averageColorFeatureWeight
            END);
   END

CREATE METHOD SI_setFeature
   (colorHistogramFeature SI_ColorHistogram,
    colorHistogramFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF colorHistogramFeatureWeight IS NULL OR
         colorHistogramFeatureWeight < 0.0 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_ClrHstgrFtr(colorHistogramFeature).
         SI_ClrHstgrFtrWght(
            CASE
               WHEN colorHistogramFeature IS NULL THEN 0.0
               ELSE colorHistogramFeatureWeight
            END);
   END
```



```
CREATE METHOD SI_setFeature
   (positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF positionalColorFeatureWeight IS NULL OR
         positionalColorFeatureWeight < 0.0 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_ClrHstgrFtr(positionalColorFeature).
         SI_ClrHstgrFtrWght(
            CASE
               WHEN positionalColorFeature IS NULL THEN 0.0
               ELSE positionalColorFeatureWeight
            END);
   END

CREATE METHOD SI_setFeature
   (textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   FOR SI_FeatureList
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF04';

      IF textureFeatureWeight IS NULL OR
         textureFeatureWeight < 0.0 THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect feature list specification';
      END IF;
      RETURN SELF.
         SI_ClrHstgrFtr(textureFeature).
         SI_ClrHstgrFtrWght(
         CASE
            WHEN textureFeature IS NULL THEN 0.0
            ELSE textureFeatureWeight
         END);
   END
```

Description

1) The method *SI_setFeature*(*SI_AverageColor*, *DOUBLE PRECISION*) takes the following input parameters:

   a) an *SI_AverageColor* value *averageColorFeature*,

   b) a DOUBLE PRECISION value *averageColorFeatureWeight*.

2) If the parameter *averageColorFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

3) If the parameter *averageColorFeature* is the null value, then the attribute value of *SI_AvgClrFtrWght* is set to 0 (zero), disregarding the value of the parameter *averageColorFeatureWeight*.

4) The method *SI_setFeature*(*SI_ColorHistogram*, *DOUBLE PRECISION*) takes the following input parameters:

   a) an *SI_ColorHistogram* value *colorHistogramFeature*,

   b) a DOUBLE PRECISION value *colorHistogramFeatureWeight*.

5) If the parameter *colorHistogramFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

6) If the parameter *colorHistogramFeature* is the null value, then the attribute value of *SI_ClrHstgrFtrWght* is set to 0 (zero), disregarding the value of the parameter *colorHistogramFeatureWeight*.

7) The method *SI_setFeature*(*SI_PositionalColor*, *DOUBLE PRECISION*) takes the following input parameters:

   a) an *SI_PositionalColor* value *positionalColorFeature*,

   b) a DOUBLE PRECISION value *positionalColorFeatureWeight*.

8) If the parameter *positionalColorFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

9) If the parameter *positionalColorFeature* is the null value, then the attribute value of *SI_PstnlClrFtrWght* is set to 0 (zero), disregarding the value of the parameter *positionalColorFeatureWeight*.

10) The method *SI_setFeature*(*SI_Texture*, *DOUBLE PRECISION*) takes the following input parameters:

    a) an *SI_Texture* value *textureFeature*,

    b) a DOUBLE PRECISION value *textureFeatureWeight*.

11) If the parameter *textureFeatureWeight* is the null value, or if it is less than 0 (zero), then an exception condition is raised: *SQL/MM Still Image exception – incorrect feature list specification*.

12) If the parameter *textureFeature* is the null value, then the attribute value of *SI_TextureFtrWght* is set to 0 (zero), disregarding the value of the parameter *textureFeatureWeight*.

6.5.4 SI_Score Method

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_FeatureList* value.

Definition

```
CREATE METHOD SI_Score
   (image SI_StillImage)
   RETURNS DOUBLE PRECISION
   FOR SI_FeatureList
   BEGIN
      DECLARE totalWeight DOUBLE PRECISION;
      DECLARE result DOUBLE PRECISION;

      IF SELF IS NULL OR image IS NULL THEN
         RETURN NULL;
      END IF;
      IF SELF.SI_AvgClrFtr IS NULL AND
            SI_ClrHstgrFtr IS NULL AND
            SI_PstnlClrFtr IS NULL AND
            SI_TextureFtr IS NULL THEN
         RETURN NULL;
      END IF;
      SET totalWeight = SI_AvgClrFtrWght + SI_ClrHstgrFtrWght +
         SI_PstnlClrFtrWght + SI_TextureFtrWght;
      IF totalWeight = 0.0 THEN
         RETURN NULL;
      END IF;
      SET result = 0.0;
      IF SELF.SI_AvgClrFtr IS NOT NULL THEN
         SET result = result +
            SELF.SI_AvgClrFtr.SI_Score(image) * SI_AvgClrFtrWght;
      END IF;
      IF SELF.SI_ClrHstgrFtr IS NOT NULL THEN
         SET result = result +
            SELF.ClrHstgrFtr.SI_Score(image) * SI_ClrHstgrFtrWght;
      END IF;
      IF SELF.SI_PstnlClrFtr IS NOT NULL THEN
         SET result = result +
            SELF.PstnlClrFtr.SI_Score(image) * SI_PstnlClrFtrWght;
      END IF;
      IF SELF.SI_TextureFtr IS NOT NULL THEN
         SET result = result +
            SELF.TextureFtr.SI_Score(image) * SI_TextureFtrWght;
      END IF;
      RETURN result/totalWeight;
   END
```


Description

1) The method *SI_Score(SI_StillImage)* takes the following input parameter:

   a) an *SI_StillImage* value *image*.

2) The method *SI_Score(SI_StillImage)* returns the null value if one of the following is true:

   a) SELF is the null value.

b) *image* is the null value.

c) The values *SELF.SI_AvgClrFtr*, *SELF.SI_ClrHstgrFtr*, *SELF.SI_PstnlClrFtr*, and *SELF.SI_TextureFtr* are all the null value.

d) The sum of *SELF.SI_AvgClrFtrWght*, *SELF.SI_ClrHstgrFtrWght*, *SELF.SI_PstnlClrFtrWght*, and *SELF.SI_TextureFtrWght* is 0 (zero).

3) The method *SI_Score(SI_StillImage)* returns a value greater than or equal to 0 (zero). The lower the returned value, the better *image* is characterized by the *SI_FeatureList* value used for scoring *image*. Let N be the number of feature attributes of SELF that are not the null value. For i ranging from 1 (one) to N, let F_i be the value of that attribute, and W_i the value of the corresponding weight attribute. Then the result:

$$\frac{\sum_{i=1}^{N} F_i.SI\_Score(image) \cdot W_i}{\sum_{i=1}^{N} W_i}$$

6.5.5 SI_mkFeatureList Function

Purpose

Return a specified *SI_FeatureList* value.

Definition

```
CREATE FUNCTION SI_mkFeatureList
   (averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION,
    colorHistogramFeature SI_ColorHistogram,
    colorHistogramFeatureWeight DOUBLE PRECISION,
    positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION,
    textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN NEW SI_FeatureList
      (averageColorFeature, averageColorFeatureWeight,
       colorHistogramFeature, colorHistogramFeatureWeight,
       positionalColorFeature, positionalColorFeatureWeight,
       textureFeature, textureFeatureWeight)
```

Description

1) The function *SI_mkFeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_AverageColor* value *averageColorFeature*,

   b) a DOUBLE PRECISION value *averageColorFeatureWeight*,

   c) an *SI_ColorHistogram* value *colorHistogramFeature*,

   d) a DOUBLE PRECISION value *colorHistogramFeatureWeight*,

   e) an *SI_PositionalColor* value *positionalColorFeature*,

   f) a DOUBLE PRECISION value *positionalColorFeatureWeight*,

   g) an *SI_Texture* value *textureFeature*,

   h) a DOUBLE PRECISION value *textureFeatureWeight*.

6.5.6 SI_ScoreByFtrList Function

Purpose

Determine and return the score of an *SI_StillImage* value to a given *SI_FeatureList* value.

Definition

```
CREATE FUNCTION SI_ScoreByFtrList
   (feature SI_FeatureList,
    image SI_StillImage)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN feature.SI_Score(image)
```

Description

1) The function *SI_ScoreByFtrList(SI_FeatureList, SI_StillImage)* takes the following input parameters:

   a) an *SI_FeatureList* value *feature*,

   b) an *SI_StillImage* value *image*.

6.5.7 Regular Functions Complementing SI_setFeature Methods

Purpose

Modify a designated feature attribute and the corresponding weight attribute of an *SI_FeatureList* value.

Definition

```
CREATE FUNCTION SI_setAvgClrFtrW
   (featureList SI_FeatureList,
    averageColorFeature SI_AverageColor,
    averageColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(averageColorFeature, averageColorFeatureWeight)

CREATE FUNCTION SI_setClrHstgrFtrW
   (featureList SI_FeatureList,
    colorHistogramFeature SI_ColorHistogram,
    colorHistogramFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(colorHistogramFeature, colorHistogramFeatureWeight)

CREATE FUNCTION SI_setPstnlClrFtrW
   (featureList SI_FeatureList,
    positionalColorFeature SI_PositionalColor,
    positionalColorFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(positionalColorFeature, positionalColorFeatureWeight)

CREATE FUNCTION SI_setTextureFtrW
   (featureList SI_FeatureList,
    textureFeature SI_Texture,
    textureFeatureWeight DOUBLE PRECISION)
   RETURNS SI_FeatureList
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.
      SI_setFeature(textureFeature, textureFeatureWeight)
```

6.5.7 Regular Functions Complementing SI_setFeature Methods

Description

1) The function *SI_setAvgClrFtrW (SI_FeatureList, SI_AverageColor, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_FeatureList* value *featureList*,

   b) an *SI_AverageColor* value *averageColorFeature*,

   c) a DOUBLE PRECISION value *averageColorFeatureWeight*.

2) The function *SI_setClrHstgrFtrW(SI_FeatureList, SI_ColorHistogram, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_FeatureList* value *featureList*,

   b) an *SI_ColorHistogram* value *colorHistogramFeature*,

   c) a DOUBLE PRECISION value *colorHistogramFeatureWeight*.

3) The function *SI_setPstnlClrW(SI_FeatureList, SI_PositionalColor, DOUBLE PRECISION)* takes the following input parameters:

   a) an *SI_FeatureList* value *featureList*,

   b) an *SI_PositionalColor* value *positionalColorFeature*,

   c) a DOUBLE PRECISION value *positionalColorFeatureWeight*.

4) The function *SI_setTextureW*(*SI_FeatureList, SI_Texture*, *DOUBLE PRECISION*) takes the following input parameters:

   a) an *SI_FeatureList* value *featureList*,

   b) an *SI_Texture* value *textureFeature*,

   c) a DOUBLE PRECISION value *textureFeatureWeight*.

6.5.8 Regular Functions Complementing Observer Functions of type SI_FeatureList

Purpose

Obtain the value of a designated attribute from an *SI_FeatureList* value.

Definition

```
CREATE FUNCTION SI_getAvgClrFtr
   (featureList SI_FeatureList)
   RETURNS SI_AverageColor
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_AvgClrFtr

CREATE FUNCTION SI_getAvgClrFtrW
   (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_AvgClrFtrWght

CREATE FUNCTION SI_getClrHstgrFtr
   (featureList SI_FeatureList)
   RETURNS SI_ColorHistogram
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_ClrHstgrFtr

CREATE FUNCTION SI_getClrHstgrFtrW
   (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_ClrHstgrFtrWght

CREATE FUNCTION SI_getPstnlClrFtr
   (featureList SI_FeatureList)
   RETURNS SI_PositionalColor
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN featureList.SI_PstnlClrFtr

CREATE FUNCTION SI_getPstnlClrFtrW
   (featureList SI_FeatureList)
   RETURNS DOUBLE PRECISION
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

```
      STATIC DISPATCH
      RETURN featureList.SI_PstnlClrFtrWght


   CREATE FUNCTION SI_getTextureFtr
      (featureList SI_FeatureList)
      RETURNS SI_Texture
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT
      STATIC DISPATCH
      RETURN featureList.SI_TextureFtr

   CREATE FUNCTION SI_getTextureFtrW
      (featureList SI_FeatureList)
      RETURNS DOUBLE PRECISION
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT
      STATIC DISPATCH
      RETURN featureList.SI_TextureFtrWght
```

Description

1) The function *SI_getAvgClrFtr(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

2) The function *SI_getAvgClrFtrW(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

3) The function *SI_getClrHstgrFtr(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

4) The function *SI_getClrHstgrFtrW(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

5) The function *SI_getPstnlClrFtr(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

6) The function *SI_getPstnlClrFtrW(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

7) The function *SI_getTextureFtr(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

8) The function *SI_getTextureFtrW(SI_FeatureList)* takes the following input parameter:

   a) an *SI_FeatureList* value *featureList*.

6.6 Auxiliary Types and Routines

6.6.1 SI_Color Type

Purpose

Provide the definition of the type SI_Color and facilities for constructing values of this type.

Definition

```
CREATE TYPE SI_Color
   AS (
      --
      -- !! See Description
      --
   )
   INSTANTIABLE
   NOT FINAL

   METHOD SI_RGBColor
      (redValue INTEGER,
       greenValue INTEGER,
       blueValue INTEGER)
      RETURNS SI_Color
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT
```

Description

1) The *SI_Color* type provides for public use:

   a) a method *SI_RGBColor(INTEGER, INTEGER, INTEGER)*,

   b) a function *SI_mkRGBColor(INTEGER, INTEGER, INTEGER)*.

   NOTE 7 An implementation may provide additional methods or functions to construct *SI_Color* values using the representation of color values in other color spaces beside RGB.

2) The *SI_Color* type represents color values using an implementation-dependent set of attributes.

3) The implementation-dependent attributes are not for public use. There are no GRANT statements granting EXECUTE privilege on the observer and mutator functions for the implementation-dependent set of attributes.

4) Values of the *SI_Color* type represent color values using an implementation-dependent color space.

6.6.2 SI_RGBColor Method

Purpose

Construct a specified *SI_Color* value using the representation of colors in the RGB color space.

Definition

```
CREATE METHOD SI_RGBColor
   (redValue INTEGER,
    greenValue INTEGER,
    blueValue INTEGER)
   RETURNS SI_Color
   FOR SI_Color
   BEGIN
      DECLARE InvalidInput CONDITION FOR SQLSTATE '2FF05';

      IF redValue IS NULL OR
         greenValue IS NULL OR
         blueValue IS NULL OR
         redValue < 0 OR redValue > SI_MaxRGBColor OR
         greenValue < 0 OR greenValue > SI_MaxRGBColor OR
         blueValue < 0 OR blueValue > SI_MaxRGBColor THEN
         SIGNAL InvalidInput
            SET MESSAGE_TEXT = 'incorrect color specification';
      END IF;
      RETURN SELF.
         --
         -- !! See Description
         --
   END
```

Definitional Rules

1) *SI_MaxRGBColor* is the implementation-defined maximum value for each component of a color value that is represented in the RGB color space.

Description

1) The method *SI_RGBColor(INTEGER, INTEGER, INTEGER)* takes the following input parameters:

   a) an INTEGER value *redValue*,

   b) an INTEGER value *greenValue*,

   c) an INTEGER value *blueValue*.

2) If any of the parameters is the null value, is if any of the input values are less than 0 (zero) or greater than *SI_MaxRGBColor*, then an exception condition is raised: *SQL/MM Still Image exception – incorrect color specification*.

3) It is implementation-dependent how the color value in the RGB color spaces, specified by its red, green, and blue components, is represented by the implementation-dependent set of attributes of the *SI_Color* type.

6.6.3 SI_mkRGBColor Function

Purpose

Construct a specified *SI_Color* value using the representation of colors in the RGB color space.

Definition

```
CREATE FUNCTION SI_mkRGBColor
   (redValue INTEGER,
    greenValue INTEGER,
    blueValue INTEGER)
   RETURNS SI_Color
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   RETURN NEW SI_RGBColor(redValue, greenValue, blueValue)
```

Description

The function *SI_mkRGBColor(INTEGER, INTEGER, INTEGER)* takes the following input parameters:

a) an INTEGER value *redValue*,

b) an INTEGER value *greenValue*,

c) an INTEGER value *blueValue*.

Blank page

7 SQL/MM Still Image Information Schema

7.1 Introduction

The SQL/MM Still Image Information Schema views are defined as being in a schema named *SI_INFORMTN_SCHEMA* enabling these views to be accessed in the same way as any other tables in any other schema. SELECT privilege on all of these views is granted to PUBLIC WITH GRANT OPTION so that they can be queried by any user and so that SELECT privilege can be further granted on views that reference these Information Schema views. No other privilege is granted on them so they cannot be updated.

In order to provide access to the same information that is available via the *SI_INFORMTN_SCHEMA* to an SQL-Agent in an SQL-environment where the SQL-implementation does not support Feature F391, ''Long identifiers'' of Part 2 of ISO/IEC 9075, alternative views are provided that use only short identifiers.

An implementation may define objects that are associated with *SI_INFORMTN_SCHEMA* that are not defined in this Clause. An implementation may also add columns to tables that are defined in this Clause.

7.2 SI_IMAGE_FORMATS view

Function

Identify the supported image formats.

Definition

```
CREATE VIEW SI_IMAGE_FORMATS AS
   SELECT SI_FORMAT
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMATS
```


7.3 SI_IMAGE_FORMAT_CONVERSIONS view

Function

Identify the source and target image formats for which an image format conversion is supported.

Definition

```
CREATE VIEW SI_IMAGE_FORMAT_CONVERSIONS AS
   SELECT SI_SOURCE_FORMAT, SI_TARGET_FORMAT
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_CONVERSIONS
```

7.4 SI_IMAGE_FORMAT_FEATURES view

Purpose

Identify the image formats for which a certain basic feature is supported.

Definition

```
CREATE VIEW SI_IMAGE_FORMAT_FEATURES AS
   SELECT SI_FORMAT, SI_FEATURE_NAME
      FROM SI_DEFINITION_SCHEMA.SI_IMAGE_FORMAT_FEATURES
```


