Seminar:
Multimediale Informationssysteme

SQL/MM

SQL MultiMedia und Application Packages

Christian Müller

Sommersemester 2002

28.06.2002

Inhaltsverzeichnis

1 Einleitung

Multimediale Daten werden heute immer wichtiger. Um diese Daten besser in Datenbanken speichern und verarbeiten zu können, wurde SQL/MM, mit dem Ziel die verschiedenen vorhandenen Ansätze zur Speicherung zu vereinheitlichen, entwickelt [8]. Der Standard definiert dazu für jeden Datentyp auf Basis der Typ-Erweiterbarkeit von SQL:1999 neue benutzerdefinierte Typen (UDT) mit eigenen Funktionen und Methoden.

Durch Typ-Erweiterbarkeit ist es nun beispielsweise möglich einen neuen Typ Adresse zu erzeugen:

```
CREATE TYPE Adresse AS
 (Straße    CHAR(40),
  Stadt     CHAR(30),
  Plz       INTEGER);
```

Diesen Typ Adresse kann man nun beim Erzeugen einer neuen Tabelle als Spaltentyp nutzen.

```
CREATE TABLE Mitarbeiter
 (id        INTEGER PRIMARY KEY,
  Name      CHAR(40),
  Ort       Adresse);
```

Auf die Spalte Ort vom Typ Adresse kann man dann mittels automatisch erzeugter Methoden und Funktionen zugreifen. So gibt es eine Konstruktor-Methode (welche immer den Namen des definierten Typs hat) zum Erzeugen einer neuen Instanz, sowie Observer- und Mutator-Methoden (mit dem Namen des jeweiligen Attributs) für jedes einzelne Attribut.

```
UPDATE Mitarbeiter
SET Ort = Ort.Stadt('KL')
WHERE Ort.Plz() = 67663;
```

Auf Basis dieser Standardmethoden können noch erweiterte Methoden oder Funktionen mittels CREATE METHOD bzw. CREATE FUNCTION definiert werden.

In SQL/MM werden vier Anwendungsgebiete unterschieden, für welche neue Typen definiert werden. Hinzu kommt dann noch ein einleitender Teil zu SQL/MM, wodurch der Standard in fünf Teile (Framework, Full-Text, Spatial, Still Image, Data Mining) eingeteilt wird, welche hier erst kurz vorgestellt werden, um dann auf die drei Teile Full-Text, Spatial und Still Image näher einzugehen. Die Speicherung von Videos wie auch von Audiodaten werden im SQL/MM Standard nicht spezifiziert, werden in der Zukunft allerdings vielleicht noch aufgenommen.

2 Teile von SQL/MM

2.1 Framework

Im ersten Teil Framework von SQL/MM [1] werden Definitionen, Notationen und Konzepte vorgestellt, welche von mindestens zwei anderen Teilen genutzt werden. Hauptsächlich wird dabei auf den SQL:1999 Standard verwiesen und noch einige neue Definitionen gemacht.

2.2 Full-Text

Im Teil Full-Text von SQL/MM [2] werden Typen für lange Texte und für Suchpattern auf diesen Texten spezifiziert. Full-Text unterscheidet sich von normalen Strings nicht nur durch die Länge, sondern auch durch spezielle Datenbankoperationen. So unterstützt Full-Text zum Beispiel die Suche nach Pluralformen von Nomen oder nach konjugierten Formen von Verben.

2.3 Spatial

SQL/MM Spatial [3] definiert Typen und Methoden, um geometrische Objekte in Datenbanken zu speichern und zu verwenden. Diese Daten werden oft für Verkehrsplanung, Stadtplanung oder auch im militärischen Bereich genutzt. Spatial unterstützt allerdings nur zwei-dimensionale Daten, drei-dimensionale oder höherdimensionale Daten sind erst für zukünftige Versionen vorgesehen.

2.4 Still Image

Bei Still Image [4] werden Typen für zwei-dimensionale unbewegte Bilder definiert und auch Methoden zum Rotieren, Vergrößern oder um die Bilder auf andere Weise zu verändern. Weiterhin werden noch andere Merkmale, wie z.B. durchschnittliche Farbe und ihre Speicherung spezifiziert.

2.5 Data Mining

Im Data Mining Teil [5] wird ein Interface zu Data Mining-Algorithmen definiert. Diese Algorithmen sollen in großen Datenmengen bisher unbekannte Informationen finden. Dazu werden vom Standard vier verschiedene Techniken unterstützt. Als erste Technik wäre hier das sogenannte "rule model" zu erwähnen, welches nach Zusammenhängen in den verschiedenen Datenteilen sucht. Beim "clustering model" werden Daten mit ähnlichen Eigenschaften gruppiert, um die wichtigsten Eigenschaften zu ermitteln. Durch die Analyse der bestehenden Daten, wird beim "regression model" eine Vorhersage für die Bewertung neuer Daten getroffen. Die letzte Technik, das "classification model" ist dem "regression model" sehr ähnlich, ordnet die neuen Daten jedoch in eine schon vorhandene Gruppe ein.

Der Teil Data Mining wird hier nicht weiter behandelt, da kein Bezug zum Thema Multimediale Informationssysteme besteht.

3 Full-Text

Zur Speicherung von Texten wird in SQL/MM der Teil Full-Text spezifiziert. Durch den Full-Text Standard werden die Texte nicht nur als String erfaßt, den Texten werden Sprachen zugeordnet, wodurch sich die Texte in einzelne Wörter, Sätze oder Absätze einteilen lassen.

```
CREATE TABLE Mitarbeiter
 (id             INTEGER,
  name           CHAR(30),
  beschreibung   FULLTEXT);
```

erzeugt eine neue Tabelle mit dem Namen Mitarbeiter, wobei das Attribut *beschreibung* vom Typ FullText ist. Zum Einfügen eines neuen Mitarbeiters, dessen Beschreibung in der Variable *text* liegt, kann folgende Anweisung genutzt werden:

```
INSERT INTO Mitarbeiter
  VALUES (1, 'Mayer', new FullText(:text));
```

Bei der Suche in Texten (wie z.B. in *beschreibung*) kann mit Hilfe verschiedener Methoden (Contains, Score, NumberOfMatches) gesucht werden. Die Einfachste dieser Methoden ist die Contains-Methode, welche nur nach der Existenz des spezifizierten Pattern (Suchmuster) sucht:

```
SELECT M.id, M.name
FROM Mitarbeiter M
WHERE M.beschreibung.Contains (Pattern)
```

Im folgenden werden die verschiedenen Arten der Pattern vorgestellt, um danach noch auf Bewertungsfunktionen und Sprachaspekte einzugehen.

3.1 Wortsuche

Die einfachste Form der Suchpattern sind die single word patterns. Sie bestehen aus einer Folge von Buchstaben, also im allgemeinen aus genau einem Wort. So würde der Test

```
BeispielText.Contains(' "Standard" ');
```

erfolgreich sein, wenn *Standard* in *BeispielText* vorkommt. An diesem Beispiel kann man gleich die Syntax der Anfragen sehen, so muss das Suchwort von Anführungszeichen umschlossen sein. Die Leerzeichen zwischen einfachen und doppelten Anführungszeichen sind nicht relevant, sondern werden hier nur aufgrund der besseren Lesbarkeit genutzt. *BeispielText* steht hier, wie im folgenden auch für einen FullText-Typ.

3.2 Phrasensuche

Single phrase patterns stellen eine Sequenz von Wörtern dar, wobei die Wörter durch ein Trennzeichen (hier Leerzeichen) getrennt werden. Auch die single phrase patterns werden von doppelten Anführungszeichen umschlossen. Der Test

```
BeispielText.Contains(' "internationaler Standard" ')
```

wäre erfolgreich, wenn *internationaler Standard* in genau dieser Reihenfolge mindestens einmal im zugrunde liegenden Text vorkommt.

3.3 Suche nach mehreren Wörtern bzw. Wortvariationen

Pattern, die eine Menge von Wörtern repräsentieren, können auf verschiedene Arten angegeben werden.

Durch die Nutzung eines Unterstrichs oder des Prozentzeichens an einer beliebigen Stelle eines Wortes, kann man mehrere einzelne Wörter spezifizieren. Der Test

```
BeispielText.Contains(' "Standard_" ')
```

liefert true zurück, wenn im *BeispielText* das Wort *Standard* gefolgt von einem einzelnen Buchstaben (z.B. "s") vorkommt. Der Unterstrich steht also für einen einzelnen Buchstaben. Das Prozentzeichen steht dagegen für eine beliebige Sequenz von Buchstaben, wozu auch die leere Sequenz gehört. Der Test

```
BeispielText.Contains(' "Standard%" ')
```

wäre nun auch erfolgreich, wenn *Standard* oder *Standardisierung* vorkäme.

Es gibt noch die Möglichkeit aus einem Startausdruck mehrere Patterns generieren zu lassen, die in bestimmten Zusammenhängen mit dem Startausdruck stehen. Dazu gibt es folgende Arten von Pattern:

- *FT_ Soundex:* Suche aufgrund des Klanges (Aussprache)
- *FT_ Fuzzy:* Suche nach Ausdrücken mit ähnlicher Buchstabenfolge
- *FT_ BroaderTerm:* Suche nach allgemeineren Ausdrücken im spezifizierten Thesaurus
- *FT_ NarrowerTerm:* Suche nach spezielleren Ausdrücken im spezifizierten Thesaurus
- *FT_ Synonym:* Suche nach Synonymen
- *FT_ PreferredTerm:* Suche nach einem bevorzugten Ausdruck zu dem spezifizierten Ausdruck
- *FT_ RelatedTerm:* Suche nach gleichwertigen verwandten Ausdrücken
- *FT_ TopTerm:* Suche nach dem allgemeinsten Überbegriff

Hierzu nun ein Beispiel für die Suche nach synonymen Begriffen. Wenn der Thesaurus von *Informatik* so gesetzt wurde, dass *Liste* und *Sequenz* synonym verwendet werden, dann wird folgender Test

```
BeispielText.Contains(' THESAURUS "Informatik"
EXPAND SYNONYM TERM OF "Liste" ')
```

auch true zurückliefern, wenn *Sequenz* in *BeispielText* vokommt.

Es ist auch möglich, einfach eine Liste von einzelnen Wortpattern durch Komma getrennt anzugeben, wobei auch Unterstrich und Prozentzeichen verwendet werden dürfen. Das Pattern

```
' ( "Standard_" , "International" ) '
```

passt zu einem Text, wenn in dem Text eines der gebildeten Wörter vorkommt. Also zum Beispiel *Standards* oder auch *International.*

Auch die Stammform eines Wortes kann miteinbezogen werden. So wird das Pattern

```
' STEMMED FORM OF "Mäuse" '
```

genauso behandelt wie

```
' ( "Maus" , "Mäuse" ) '
```

3.4 Suche nach mehreren Phrasen bzw. Phrasenvariationen

Auch bei Folgen von Wörtern sind wie bei einzelnen Wörtern mengenwertige Pattern möglich. In vielen Punkten ist die Verwendung genauso wie bei den einzelnen Wörtern, weshalb hier nur auf einige Besonderheiten eingegangen wird.

So kann das Prozentzeichen hier für ein potentielles Wort stehen. Der Test

```
BeispielText.Contains(' "der % Standard" ')
```

ist erfolgreich, wenn *der Standard* in dem Text vorkommt, aber auch, wenn zwischen *der* und *Standard* ein beliebiges Wort steht, wie z. B. *der internationale Standard.*

Bei den Stammformen würde die Phrase

```
' STEMMED FORM OF "Internationale Standards" '
```

wie

```
' ( "International Standard", "Internationaler Standard",
"Internationales Standard", ... ' )
```

behandelt werden. Es werden also für jedes Wort der Phrase alle möglichen Formen gebildet und dann alle Permutationen gebildet.

3.5 Suche nach benachbarten Termen

Hier werden zwei Arten von Pattern unterschieden, das proximity expansion pattern und das context condition pattern.

Ein proximity expansion pattern hat folgende Form:

```
' ( "Standard", "International" )    --erstes Pattern
  NEAR "Sprache"                       --zweites Pattern
  WITHIN 0                               --Anzahl der Einheiten
  SENTENCES                              --Art der Einheit
  IN ORDER )'
```

Das erste und das zweite Pattern kann ein single word pattern oder ein set of single word pattern sein. Durch die Anzahl und die Art der Einheiten (Paragraphen, Sätze, Wörter oder Buchstaben) wird ein Fenster für das Auftreten der Pattern spezifiziert. Durch den letzten Teil wird angegeben, ob die Pattern in einer bestimmten Reihenfolge auftreten müssen. Ein Text passt zu diesem Pattern, wenn das erste Pattern in dem Text gefunden wird und im angegebenen Fenster (welches mit dem ersten Auftreten des ersten Patterns beginnt) das zweite Pattern, also *Sprache,* gefunden wird. Da IN ORDER angegeben wurde, muss *Standard* oder *International* auch vor *Sprache* vorkommen.

Ein context condition pattern besteht aus zwei oder mehr Pattern und einer Fensterangabe, welche ein Satz oder ein Paragraph sein kann. Das obige Beispiel kann man auch als context condition pattern angeben:

```
' ( "Standard", "International" )
  IN SAME SENTENCE AS
  "Sprache" '
```

Hier kann man allerdings das Fenster nicht variabel festlegen, dafür ist es möglich mehr als zwei Pattern anzugeben, die in dem Fenster liegen müssen.

3.6 Boolsche Operatoren

Teile von Pattern können auch mittels booleschen Operatoren verknüpft werden. Es gibt drei Operatoren und zwar "NOT", "&" (AND) und "|" (OR), deren Präzedenz auch in dieser Reihenfolge besteht. Folgendes Beispiel

```
BeispielText.Contains( ' NOT "Internationaler Standard" & "Test" ' )
```

liefert true zurück, wenn *Test* vorkommt, aber *Internationaler Standard* nicht im Text vorkommt.

3.7 Scoring

Da die Contains-Methode keine Aussage darüber macht, wie gut der Text von dem Pattern charakterisiert wird, gibt es eine Score- und eine NumberOfMatches-Methode.

Die Score-Methode wird auf die gleiche Weise wie Contains angewandt, gibt jedoch eine nicht-negative Fließkommazahl zurück. Je größer diese Zahl ist, desto besser passen der Text und das Pattern zusammen. Der genaue Zusammenhang zwischen Text und Pattern und der Bewertungszahl ist der Implementierung überlassen.

Die NumberOfMatches-Methode kann kein beliebiges Pattern übergeben bekommen, sondern nur ein Pattern, welches aus Wörtern oder Phrasen besteht und liefert die Anzahl des Auftretens des Pattern zurück.

3.8 Sprachaspekte

Alle Werte des FullText Typ sind mit einer bestimmten Sprache verbunden. Dies wird benötigt um Wort-, Satz- und Absatzgrenzen zu erkennen, um ähnlich klingende Wörter oder Wörter in ihrer Stammform zu finden, um Stopp-Wörter zu verwenden und für die Wort-Normalisierung.

Stopp-Wörter kommen in Texten meist so häufig vor, dass sie für die Texterkennung nutzlos sind. Beispiele für solche Worte im Deutschen sind der, die, das, und, oder, ... Bei der Texterkennung können Stopp-Wörter in Phrasen durch andere Stopp-Wörter ersetzt werden. So liefert der Test

```
BeispielText.Contains(' "Satz oder Paragraph" ')
```

true zurück, auch wenn im Text nur *Satz und Paragraph* vorkommt, wobei *und* dabei auch durch andere Stopp-Wörter ersetzt werden kann.

Bei der Auswertung von Score-, NumberOfMethods- oder Contains-Methoden darf die Implementierung die verwendeten Wörter in den Pattern normalisieren. Die Art der Normalisierung wird auch der Implementierung überlassen. So kann z.B. ' "Möller" ' durch ' " Moeller" ' ersetzt werden.

4 Spatial

Durch den Teil Spatial von SQL/MM wird die Speicherung von geographischen Daten spezifiziert. Dazu werden für verschiedene geometrische Figuren, wie z.B. Punkte, Kurven oder Polygone neue benutzerdefinierte Typen definiert. Diese Datentypen stellen die Basisfunktionalität für geographische Informationssysteme dar. So kann man beispielsweise die Grundstücke in einer Tabelle verwalten.

```
CREATE TABLE Grundstück
(id          INTEGER PRIMARY KEY,
 Geometrie   ST_POLYGON,
 Besitzer    CHAR(40));
```

Die Ausdehnung der Grundstücke wird hier in dem Attribut Geometrie, welches von Typ ST_Polygon ist, verwaltet. Durch auf diesem Typ definierten Methoden, kann man die Entfernung zwischen zwei Grundstücken berechnen.

```
SELECT G1.id, G2.id, G1.Geometrie.ST_Distance(G2.Geometrie)
FROM Grundstück G1, Grundstück G2
WHERE ...
```

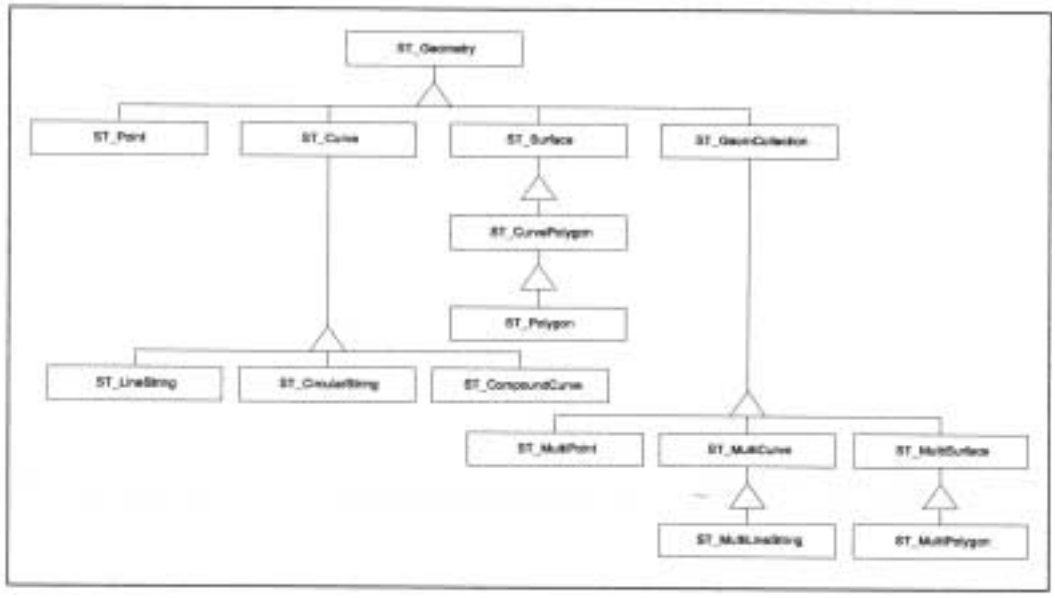
In SQL/MM Spatial gibt es auch noch andere Methoden, welche später noch genauer erläutert werden.

Weiterhin wird jede Geometrie in Bezug auf ein Referenzsystem (ST_SpatialRefSys) betrachtet, dieses kann beispielsweise die Einteilung in Längen- und Breitengrade sein. Durch diese Zuordnung werden den Daten der Geometrien erst Bedeutungen zugeordnet. Weiterhin gibt es noch zwei weitere Datentypen welche eine Richtung (ST_Direction) oder einen Winkel (ST_Angle), zum Beispiel zwischen zwei sich schneidenden Linien, repräsentieren.

Im folgenden werden nun die verschiedenen Geometrietypen beschrieben, bevor zum Ende noch auf eine XML-basierte Darstellung von geographischen Daten eingegangen wird.

4.1 Geometry Types

SQL/MM Spatial definiert mehrere Typhierarchien, welche alle ST_Geometry als Supertyp haben, wie man in folgender Abbildung sieht. ST_Geometry und einige Subtypen werden hier im folgenden vorgestellt.



4.1.1 ST_Geometry

ST_Geometry ist der Basistyp aller geometrischen Typen (von SQL/MM) und nicht instantiierbar. Die instantiierbaren Subtypen von ST_Geometry repräsentieren 0-dimensionale, 1-dimensionale oder 2-dimensionale Geometrien. Alle Methoden auf ST_Geometry-Werten und ihren Subtypen arbeiten im räumlichen Referenzsystem (ST_SpatialRefSys) des ersten Wertes der Parameterliste. Alle Rückgabewerte der Methoden liegen ebenfalls in diesem Referenzsystem. ST_Geometry besitzt einige Standardmethoden:

- *ST_Dimension:* gibt die Dimension von ST_Geometry zurück
- *ST_CoordDim:* gibt die Dimension des Koordinatensystems von ST_Geometry zurück, diese stimmt mit der Dimension des spatial Referenzsystems überein
- *ST_GeometryType:* gibt den Typ des ST_Geometry zurück

- *ST_SRID:* zum Überwachen und Ändern des Referenzsystems
- *ST_Transform:* transformiert einen ST_Geometry in das angegebene Referenzsystem
- *ST_IsEmpty:* testet, ob ein ST_Geometry mit der leeren Menge übereinstimmt
- *ST_IsSimple:* testet, ob ein ST_Geometry keine anormalen geometrischen Punkte hat, wie z. B. ein sich selbst schneidendes Polygon, Subtypen definieren diese Methode neu, um auf ihre eigenen Eigenschaften einzugehen

Es gibt noch weitere Methoden, welche etwas komplexere Operationen auf einem ST_Geometry durchführen:

- *ST_Boundary:* gibt den Rand des ST_Geometry, als ST_Geometry zurück
- *ST_Envelope:* gibt das umfassende Rechteck des ST_Geometry zurück
- *ST_ConvexHull:* gibt die konvexe Hülle des ST_Geometry zurück
- *ST_Buffer:* gibt ein ST_Geometry zurück, welches aus allen Punkten besteht, die einen übergebenen Abstand zum ST_Geometry nicht übersteigen

Weitere Methoden stellen Beziehungen zwischen zwei ST_Geometry- Werten her:

- *ST_Intersection:* gibt den ST_Geometry-Wert zurück, der entsteht, wenn zwei Geometrien geschnitten werden
- *ST_Union:* gibt die Vereinigung von zwei Geometrien als ST_Geometry zurück
- *ST_Difference:* die übergebene Geometrie wird von der zugrundeliegenden Geomtrie abgezogen und die entstehende Geometrie zurückgegeben
- *ST_SymDifference:* zieht zwei Geometrien gegenseitig voneinander ab und vereinigt das Ergebnis
- *ST_Distance:* gibt die minimale Distanz von zwei ST_Geometry-Werten zurück

Zum Ein- und Auslesen von ST_Geometry-Werten gibt es noch sechs weitere Methoden, jeweils zwei für eine externe Textrepräsentation, eine externe Binärrepräsentation und eine externe XML-Repräsentation (GML), welche später noch vorgestellt wird.

4.1.2 ST_Point

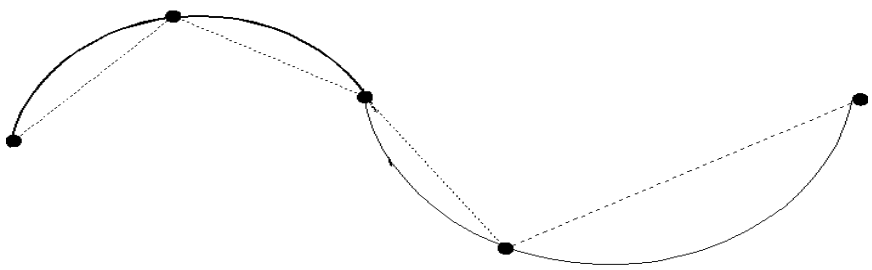
ST_Point ist eine Subklasse von ST_Geometry, welche instantiierbar ist. Sie repräsentiert 0-dimensionale Geometrien, also einen einzelnen Punkt. Ein ST_Point-Wert hat einen x- und einen y-Koordinatenwert und noch folgende spezielle Methoden:

ST_X und *ST_Y:* zum Überwachen und Ändern der Koordinaten
ST_ExplicitPoint: gibt die Koordinaten in einem Array der Länge 2 zurück
Die Methode *ST_IsSimple* gibt bei einem ST_Point immer true zurück.

4.1.3 ST_Curve

ST_Curve ist ein nicht instantiierbarer Subtyp von ST_Geometry. Er steht für eine 1-dimensionale Geometrie, die normalerweise als Sequenz von Punkten gespeichert wird, wobei die Untertypen ST_LineString (lineare Interpolation) und ST_CircularString (circular Interpolation) die Interpolation zwischen den Punkten bestimmen.

Im folgenden Beispiel zeigt die gepunktete Linie einen ST_Curve-Wert mit linearer Interpolation. Die fünf Punkte werden jeweils mit einer Gerade verbunden, wodurch sich der ST_Curve-Wert ergibt. Wird zirkulare Interpolation angewandt, so werden jeweils drei Punkte zu einem Segment zusammengefasst, wobei der letzte Punkt eines Segments immer als erster Punkt des folgenden Segments dient. Die drei Punkte eines Segments werden nun durch einen Kreisbogen verbunden, der vom ersten bis zum dritten Punkt reicht. Dadurch ergibt sich im foldenden Beispiel die durchgezogene Linie.

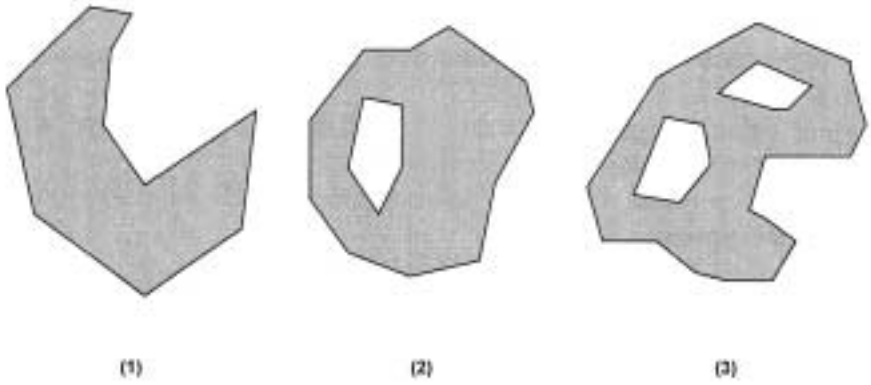


Ein anderer Subtyp ist ST_CompoundCurve, welcher eine Sequenz von beliebigen ST_Curve-Werten ist. Diese einzelnen ST_Curve-Werte werden dazu an ihren Endpunkten mit den Nachbarkurven verbunden.

Ein ST_Curve-Wert gibt bei der ST_IsSimple Methode true zurück, wenn kein Punkt zweimal durchlaufen wird. Eine Kurve ist geschlossen (ST_IsClosed), wenn der Startpunkt gleich dem Endpunkt ist, für den Fall, dass sie einfach (ST_IsSimple) und geschlossen ist, so nennt man sie auch Ring (ST_IsRing). Weiterhin gibt es noch Methoden für Start- und Endpunkt (ST_StartPpoint und ST_EndPoint) und die Länge der Kurve (ST_Length).

4.1.4 ST_CurvePolygon

ST_CurvePolygon ist ein instantiierbarer Subtyp von ST_Surface, welcher ein nicht instantiierbarer Subtyp von ST_Geometry ist. Ein ST_CurvePolygon-Wert ist eine ebene Fläche, welche durch einen äußeren Rand und mehrere innere Ränder definiert wird, wobei ein innerer Rand ein Loch im ST_CurvePolygon darstellt. Zwei Ringe dürfen sich dabei allerdings nicht schneiden, sondern nur berühren. Hier einige Beispiele für ST_Polygon-Werte, die sich von ST_CurvePolygon-Werten dadurch unterscheiden, dass sie keine Krümmungen in den Rändern erlauben:



Beim ST_CurvePolygon gibt es auch spezielle Methoden, z.B. zum Bearbeiten des äußeren (ST_ExteriorRing) und des inneren (ST_InteriorRing) Rings oder zum Ermitteln der Anzahl der inneren Ringe (ST_NumInteriorRing).

4.1.5 Kollektionstypen

Weiterhin gibt es noch Kollektionen von verschiedenen ST_Geometry-Werten. In der allgemeinen Form ist das ST_GeomCollection, welcher aus null oder mehreren ST_Geometry-Werten eines räumlichen Referenzsystems besteht. Subtypen von ST_GeomCollection (ST_MultiPoint, ST_MultiCurve, ST_MultiLineString, ST_MultiSurface, ST_MultiPolygon) nehmen dann immer nur die entsprechenden Subtypen von ST_Geometry auf.

4.2 Spatial Relationships using ST_Geometry

Das Testen von Lagebeziehungen zwischen Geometrien ist eine zentrale Funktionalität des Standards. Dazu werden einige Methoden definiert, die alle auf dem im folgenden erklärten Modell basieren:

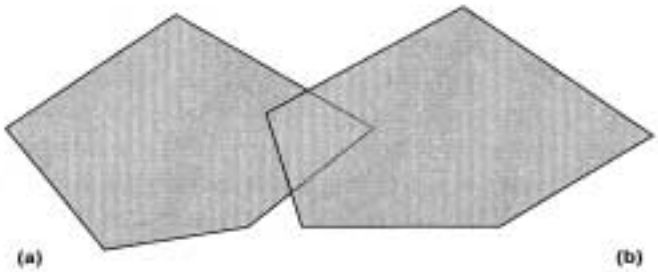
- *ST_Relate:* testet allgemein Beziehungen; alle anderen nun folgenden Methoden lassen sich darauf zurückführen
- *ST_Disjoint:* testet, ob zwei ST_Geometry-Werte disjunkt sind
- *ST_Intersects:* testet, ob sich zwei Geometrien überschneiden
- *ST_Touches:* testet auf die Berührung von zwei ST_Geometry-Werten
- *ST_Crosses:* testet, ob sich zwei Werte schneiden
- *ST_Within:* testet, ob eine Geometrie in einer Anderen liegt
- *ST_Contains:* testet, ob eine Geometrie eine andere Geometrie enthält
- *ST_Overlaps:* testet auf die Überlappung von zwei Geometrien

Die Tests dieser Methoden basieren auf einem paarweisen Test der Überschneidungen des Inneren, des Äußeren und des Randes der Geometrien. Die Ergebnisse dieser Tests werden zur Verdeutlichung in einer 3x3-Matrix repräsentiert und mit Hilfe dieser Matrix können verschiedene Lagetests definiert werden. Im folgenden wird nun diese Matrix und die damit definierten Methoden erklärt.

Der Rand eines ST_Geometry ist immer eine Menge von ST_Geometry-Werten der nächstniedrigeren Dimension, so ist der Rand eines Punktes oder einer geschlossenen Kurve die leere Menge. Das Innere bezeichnet den ST_Geometry-Wert ohne die Werte des Randes und das Äußere, sind alle Punkte die nicht zu dem Rand oder dem Inneren gehören. x.ST_Dimension gibt die maximale Dimension {-1, 0, 1, 2} von x zurück, wobei -1 der leeren Menge entspricht. Wenn a und b ST_Geometry-Werte sind, ergibt sich die DE-9IM (dimensionally extended nine intersection matrix) zu:

	Interior	Boundary	Exterior
Interior	(Interior(a) ∩ Interior(b)). ST_Dimension()	(Interior(a) ∩ Boundary(b)). ST_Dimension()	(Interior(a) ∩ Exterior(b)). ST_Dimension()
Boundary	(Boundary(a) ∩ Interior(b)). ST_Dimension()	(Boundary(a) ∩ Boundary(b)). ST_Dimension()	(Boundary(a) ∩ Exterior(b)). ST_Dimension()
Exterior	(Exterior(a) ∩ Interior(b)). ST_Dimension()	(Exterior(a) ∩ Boundary(b)). ST_Dimension()	(Exterior(a) ∩ Exterior(b)). ST_Dimension()

Als Beispiel sieht man hier eine DE-9IM für zwei sich schneidende Polygone



	Interior	Boundary	Exterior
Interior	2	1	2
Boundary	1	0	1
Exterior	2	1	2

Auf der Basis der DE-9IM wird nun die Methode ST_Relate definiert, welche die Matrix als String der Länge 9 als Pattern übergeben wird. So sieht ein Aufruf von ST_Relate mit a und b als ST_Geometry-Werte so aus:

```
a.ST_Relate(b,'TF*012***')
```

Der String wird dabei mit der Matrix, welche aus a und b berechnet wird verglichen. Das T bedeutet, dass (Interior(a)∩Interior(b)).ST_Dimension() ∈ {0.1.2}, der zweite Eintrag im String bedeutet, dass der zweite Wert der Matrix (Interior(a) ∩Boundary(b)) gleich -1 sein muss. Steht ein Stern in dem String darf dieser Wert beliebig, also {-1, 0, 1, 2} sein. Steht eine 0, 1, oder 2 so wird eine entsprechende 0, 1 oder 2 in der Matrix erwartet.

Mit Hilfe dieser Methode lassen sich dann folgende Methoden definieren, wobei a und b ST_Geometry-Werte sind:

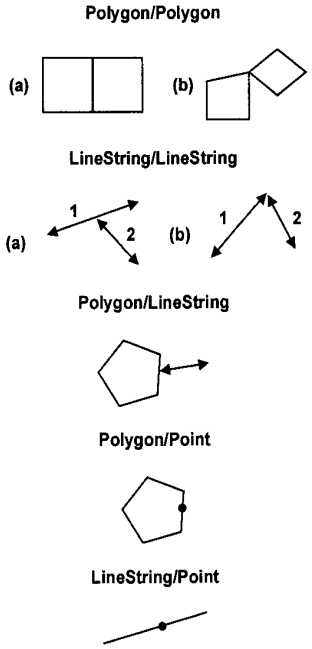
ST_Disjoint: testet ob zwei ST_Geometry-Werte disjunkt sind

a.ST_Disjoint(b)⇔
(Interior(a)∩Interior(b)=∅)∧(Interior(a)∩Boundary(b)=∅)∧
(Boundary(a)∩Interior(b)=∅)∧(Boundary(a)∩Boundary(b)=∅)⇔
a.ST_Relate(b,'FF*FF****')

ST_ Touches: testet ob zwei ST_Geometry-Werte sich berühren

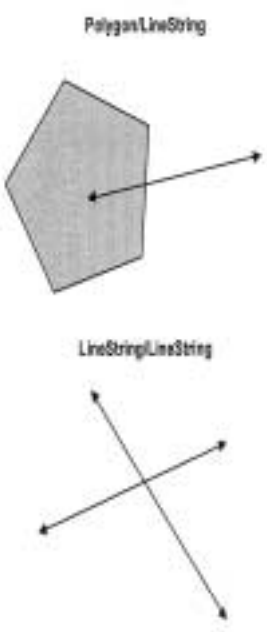
a) If a.ST_Dimension()=0 and b.ST_Dimension()=0 then return the null value
b) Sonst:
a.ST_Touches(b)⇔
(Interior(a)∩Interior(b)=∅)∧
((Boundary(a)∩Interior(b)≠∅)∨
 (Interior(a)∩Boundary(b)≠∅)∨
 (Boundary(a)∩Boundary(b)≠∅))⇔
a.ST_Relate(b,'FT******')∨
a.ST_Relate(b,'F**T*****')∨
a.ST_Relate(b,'F***T****')

Hier noch einige Beispiele für Geometrien, die sich berühren:

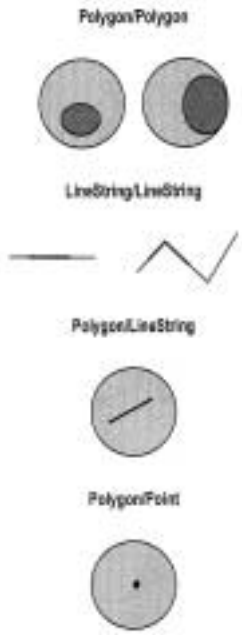


Auch die anderen Methoden werden auf ähnliche Weise definiert. Hier einige Beispiele für Überschneidung, Beinhalten und Überlappung:

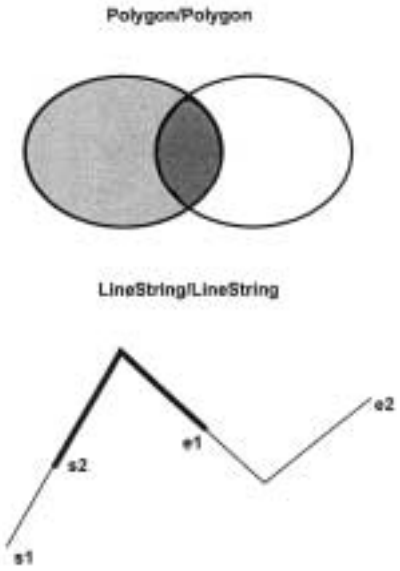
Überschneidung:



Beinhalten:



Überlappung:



4.3 XML (GML Repräsentation von OpenGIS)

Für das Ein- bzw. Auslesen von ST_Geometry-Werten werden im SQL/MM-Standard Methoden definiert, die sich auf eine XML-Repräsentation von ST_Geometry-Werten

bezieht. Diese ist die Geography Markup Language (GML) [10] von OpenGIS in der Version 2.0, welche in Zukunft noch zu einem ISO-Standard werden soll. Im folgenden werden die Grundideen von GML vorgestellt.

Darstellung von Geometrien werden in einem *geometry schema* die folgende Basistypen Point, LineString, LinearRing, Polygon, MultiPoint, MultiLineString, MultiPolygon und MultiGeometry vordefiniert. Dies geschieht in Anlehnung an das *OpenGIS Simple Feature model* [9], welches dem SQL/MM Spatial Standard zu Grunde liegt. Ein Punkt wird dann in folgender Form dargestellt:

```
<Point gid="P1" srsName="Referenzsystem">
  <coord><X>5.0</X><Y>0.57</Y></coord>
</Point>
```

Die *gid* dient als einzigartige Kennzeichnung einer Geometrie, welche allerdings optional ist. Ebenso optional ist der *srsName,* welcher die Geometrie einem Koordinatenreferenzsystem zuordnet. Durch <coord> werden dem Punkt zwei Koordinaten zugeordnet. Auf die gleiche Weise kann man auch die anderen Basistypen darstellen, so werden z.B. einem LineString mehrere Koordinatenpaare zugeordnet. Um einen LinearRing darzustellen gibt es noch eine andere Möglichkeit die Koordinaten darzustellen:

```
<LinearRing>
  <coordinates>0.0,0.0 5.0,0.0 5.0,5.0 0.0,5.0 0.0,0.0</coordinates>
</LinearRing>
```

Auf diese Weise können die einzelnen Koordinaten durch Leerzeichen getrennt angegeben werden, was von Vorteil ist, wenn man wie bei Ringen viele Koordinaten angeben kann. Diese Schreibweise kann aber auch bei Punkten verwendet werden, so können beide Schreibweisen äquivalent verwendet werden. Die anderen Basistypen werden durch Kombination von Koordinaten und Basistypen auf ähnliche Weise dargestellt.

5 Still Image

In diesem Teil von SQL/MM wird ein neuer Typ SI_StillImage definiert, welcher es erlaubt Bilder in Datenbanken zu speichern, sie zu bearbeiten und sie mit Hilfe von "visuellen" Prädikaten zu suchen [7]. Dieser Typ wird nun hier vorgestellt und noch einige dazu gehörende Konzepte erläutert [6].

5.1 Attribute von SI_StillImage

Der SI_StillImage Typ stellt nur Container-Funktionalität bereit, um mit Bildern umzugehen. Dies ist nötig, da es zu viele verschiedene Bildformate mit verschiedenen Attributen oder unterschiedlichen Namen für die gleichen Attribute gibt. So werden auch keine Aussagen über unterstützte Bildformate und die dafür unterstützten Operationen getroffen. Im Standard werden folgende Basisattribute definiert:

- *SI_content:* repräsentiert die Rohdaten des Bildes (als BLOB), zu diesen gehören nicht nur die Pixel des Bildes, sondern auch Header Informationen oder Farbtabellen. Dadurch soll gewährleistet werden, dass keine Bildinformationen verloren gehen.
- *SI_contentLength:* repräsentiert die Länge der Rohdaten, inklusive Header und sonstigen zusätzlichen Informationen.

- *SI_reference:* eine Referenz auf die Pixeldaten des Bildes in Form eines Datalinks.
- *SI_format:* hier wird das Format des Bildes gespeichert, also z.B. GIF oder TIFF.
- *SI_height, SI_width:* die Höhe bzw. Breite des Bildes wird hier repräsentiert, sofern dies möglich ist. Bei Vektorgrafiken ist dies nicht möglich, da sie keine festgelegte Größe haben.

5.2 Methoden von SI_StillImage

Neben den Standardmethoden für die Attribute, gibt es noch vier Konstruktoren *SI_StillImage,* welche Rohdaten oder eine Referenz darauf und optional einen String, welcher das Format angibt, übergeben bekommen müssen. Durch das Angeben des Formats kann auch ein Bild in einem nicht unterstützten Format, einem benutzerdefinierten Format gespeichert werden. Weiterhin werden noch folgende Methoden unterstützt:

- *SI_setContent:* ein vorhandener SI_StillImage-Wert wird mit einem Bild des gleichen Formats überschrieben, dabei werden auch SI_height, SI_width und SI_contentLength korrekt verändert.
- *SI_changeFormat:* transformiert das Bild in ein neues Format und setzt auch alle Attribute des Bildes auf die dem neuen Bild entsprechende Werte. Dies ist nur möglich, wenn die Konversion für Quellformat zu Zielformat unterstützt wird.
- *SI_Scale:* hier gibt es zwei Methoden. Das Bild wird bei beiden in der Größe verändert und zwar unter Berücksichtigung des Verhältnisses von Höhe zu Breite. Eine Methode bestimmt die neue Größe durch Höhe und Breite, die andere Methode bestimmt die Größe durch einen Skalierungsfaktor.
- *SI_Resize:* passt das Bild auf die übergebene Höhe und Breite an, ohne auf das Verhältnis von Höhe und Breite zu achten.
- *SI_Rotate:* rotiert des Bild um den angegebenen Winkel. Bei einer Rotation, welche kein Vielfaches von 90 Grad ist, wird das Bild an den Rändern erweitert und die Breite und Höhe angepasst.
- *SI_Thumbnail:* erzeugt ein Thumbnail von einem SI_StillImage-Wert
- *SI_Score:* gibt einen numerischen Wert zurück, welcher ein Maß dafür ist, inwieweit das übergebene Bildmerkmal (siehe nächster Abschnitt) mit dem eigenen Bildmerkmal-Wert übereinstimmt

5.3 Bildmerkmale (Image features)

Um auch nach den Bildern suchen zu können, werden vier Basismerkmale und ein zusammengesetztes Merkmal definiert. Diese werden bei der Suche nach bestimmten Bildern als Suchargument genutzt. Im folgenden werden nun diese Merkmale vorgestellt.

SI_AverageColor charakterisiert die durchschnittliche Farbe eines Bildes. Dazu gibt es zwei Konstruktormethoden. Die erste bekommt einen Farbwert übergeben und nimmt diesen als durchschnittliche Farbe. Die zweite Konstruktormethode berechnet die durchschnittliche Farbe aus einem übergebenen SI_StillImage-Wert. Für diese Berechnung werden die einzelnen Farbwerte (z.B. rot, grün und blau)

jedes Pixels aufsummiert und dann durch die Anzahl der Pixel im Bild dividiert, woraus sich dann die durchschnittliche Farbe ergibt.

Bei der Anwendung gilt es allerdings zu beachten, dass zwei unterschiedliche Bilder durchaus die gleiche durchschnittliche Farbe haben kann. So hat ein Bild, welches halb rot und halb blau ist als durchschnittliche Farbe lila. Ein Bild, welches komplett lila ist, hat allerdings die gleiche durchschnittliche Farbe

SI_ColorHistogram beschreibt die relative Frequenz der Farben des Bildes. Für dieses Merkmal gibt es drei Konstruktoren. Der erste Konstruktor erzeugt ein SI_ColorHistogram mit nur einem Farbwert und einer Frequenz, welches mit Hilfe der Methode SI_Append vervollständigt werden kann. Mit SI_Append kann einem bestehenden SI_ColorHistogram ein weiterer Farbwert mit einer Frequenz hinzufügen. Ein zweiter Konstruktor erzeugt mit Hilfe eines Arrays von Farbwerten und eines Arrays mit den zugehörigen Frequenzen ein vollständiges Histogramm. Der dritte Konstruktor erzeugt das Histogramm aus einem übergebenen SI_StillImage.

Zur Berechnung eines Histogramms aus einem Bildwert wird der Farbraum in eine implementierungsabhängige Anzahl an Teilen zerlegt. Zu jedem Teilbereich werden die Pixel gezählt, welche im jeweiligen Farbraum liegen. Die Frequenz wird dann durch Division dieser Werte mit der Anzahl der insgesamt vorhandenen Pixel ermittelt. Ein so erzeugtes SI_ColorHistogram besteht aus den Farbbereichen und den zugehörigen Frequenzen.

Das *SI_PositionalColor* Merkmal teilt das Bild in n mal m Rechtecke ein und repräsentiert für jedes Rechteck die durchschnittliche Farbe. Für dieses Merkmal existiert nur ein Konstruktor, welchem ein SI_StillImage als Argument übergeben wird. Das Bild wird in n mal m Rechtecke eingeteilt, wobei n und m implementierungsabhängig sind und für jeden Teilbereich der durchschnittliche Farbwert auf die gleiche Weise wie bei SI_AverageColor berechnet. Da SI_PositionalColor nur durch Übergabe eine Bildes erzeugt werden kann, muss für Anfragen erst ein Beispielbild erzeugt werden.

SI_Texture charakterisiert ein Bild anhand der Größe sich wiederholender Punkte (Grobheit), Helligkeitsvariationen (Kontrast) und der vorherrschenden Richtung. Der einzige Konstruktor erhält einen SI_StillImage-Wert übergeben und erzeugt daraus einen SI_Texture-Wert. Weitere Details über die Erzeugung des SI_Texture-Wertes werden vom Standard nicht gegeben und sind somit der Implementierung überlassen. Auch Grobheit, Kontrast und die vorherrschende Richtung werden nicht weiter definiert, was dazu führt, dass die Arten, wie ein SI_Texture-Wert beschrieben werden kann, sich sehr stark unterscheiden können.

Für diese vier Basismerkmale gibt es auch Funktionen, welche das entsprechende Merkmal aus einem SI_StillImage-Wert gewinnen. Dies sind:

- *SI_findAvgClr* für SI_AverageColor-Werte
- *SI_findClrHstgr* für SI_ColorHistogram-Werte
- *SI_findPstnlClr* für SI_PositionalColor-Werte
- *SI_findTexture* für SI_Texture-Wert

Das zusammengesetzte Merkmal *SI_FeatureList* kann genutzt werden um nach mehreren Basismerkmalen gleichzeitig zu suchen. Durch das Merkmal kann man spezifizieren, welche der Basismerkmale Teil des SI_FeatureList-Wertes sind und mit einer Gewichtung versehen. Dies kann man mit Hilfe des Konstruktors erreichen, der die vier Basismerkmale und ihre Gewichtung übergeben bekommt und daraus ein SI_FeatureList-Wert erzeugt. Soll ein Basismerkmal unberücksichtigt bleiben, kann man dies durch Übergabe eines NULL Wertes für das Merkmal und seine Wichtung erreichen. Für das spätere Verändern der Basismerkmale in einem

SI_FeatureList-Wert gibt es die Methoden *SI_ setFeature*, wobei es für jedes Basismerkmal eine separate Methode gibt.

Für die Bildsuche ist für jedes Bildmerkmal eine seperate Methode SI_Score definiert, welche eine Fließkommazahl größer oder gleich null zurückgibt. Dabei stellt eine niedrige Zahl eine bessere Übereinstimmung der Merkmale dar, als eine hohe Zahl. Auch für das zusammengesetzte Merkmal SI_FeatureList wird SI_Score-Methode definiert, dabei wird auch angegeben, wie die Wichtung der einzelnen Merkmale bei der Berechnung zu berücksichtigen ist:

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

wobei N die Anzahl der Merkmale ist, welche ungleich NULL sind,
Fi der Wert des Attributs und
Wi die Wichtung des Attributs ist.

5.4 SI Information Schema

In allen Teilen des SQL/MM-Standards gibt es ein Informationsschema, welchem allerdings im Teil Still Image eine besondere Bedeutung zukommt. Dies ist der Fall, da für Still Image nicht spezifiziert ist, welche Bild-Datenformate unterstützt werden sollen. Es wird also dem Hersteller überlassen, welche Formate er für wichtig erachtet, weshalb hier das Informationsschema benötigt wird, um sich über die, für das benötigte Bildformat unterstützte Funktionalität zu informieren.

Das angesprochene SI_INFORMTN_SCHEMA besteht aus sechs views, welche im folgenden vorgestellt werden:

- *SI_FEATURES* listet alle Merkmale auf, die vom System unterstützt werden.
- *SI_IMAGE_FORMATS* gibt die unterstützten Bildformate an und die darauf erlaubten Operationen. Vom Standard sind keine Bildformate vorgeschrieben, welche untertützt werden müssen.
- *SI_IMAGE_FORMAT_CONVERSIONS* stellt Paare von Quell- und Zielbildformaten da, welche die unterstützten Formatkonvertierungen darstellen.
- *SI_IMAGE_FORMAT_FEATURES* listet Paare von Bildformaten und Bildmerkmalen auf. Die Paare stehen für Merkmale die für das jeweilige Format unterstützt werden. Eine Implementierung kann sich so z.B. entscheiden für Vektorgrafikformate das Merkmal SI_PositionalColor oder andere Merkmale nicht zu implementieren
- *SI_THUMBNAIL_FORMATS* listet alle Formate auf, für die Thumbnails vom System erstellt werden können.
- *SI_VALUES* listet implementierungsabhängige Metavariablen und ihre Werte auf.

6 Zusammenfassung

Der SQL/MM Standard standardisiert neue Datentypen für die Speicherung und Verarbeitung von Textdaten, ruhenden Bilddaten, geographischen Daten und für Data Mining. In dieser Ausarbeitung haben wir die Datentypen mit ihren Attributen und Methoden für drei dieser Teile kennen gelernt.

So haben wir im Teil FullText die Datentypen für Texte besprochen, durch welche sich neue Möglichkeiten für die Suche in Texten ergeben, wie zum Beispiel die Suche nach zwei Wörtern in einem Satz oder nach einem ähnlich klingenden Ausdruck.

Im Teil Spatial haben wir gesehen, wie die Speicherung und Verarbeitung von geographischen Daten spezifiziert wird, wozu verschiedene Typen für Punkte, Kurven und Polygone definiert werden. Durch die für diese Typen vorgesehene Funktionalität wird es ermöglicht geographische Abhängigkeiten zwischen Daten leichter zu ermitteln.

Zur Speicherung von ruhenden Bildern haben wir den Datentyp SI_StillImage kennen gelernt, welcher durch seine Attribute und Methoden neue Funktionalität für die Bildsuche spezifiziert. Wir haben zudem noch besondere Merkmale von Still Image, wie SI_ColorHistogram, SI_AverageColor, SI_PositionalColor und SI_Texture kennen gelernt. Diese können auch zur Bildsuche bzw. zum Bildervergleich genutzt werden.

7 Literatur

[1] ISO/IEC 13249-1:2002 SQL/MM: Information Technology - Database Languages - SQL MultiMedia and Application Packages - Part1: Framework, November 2001, Takaaki Shiratori, editor.

[2] ISO/IEC 13249-2: Information Technology - Database Language - SQL MultiMedia and Application Packages - Part2: Full Text, März 2002

[3] ISO/IEC 13249-3: Information Technology - Database Language - SQL MultiMedia and Application Packages - Part3: Spatial, März 2002

[4] ISO/IEC 13249-5: Information Technology - Database Language - SQL MultiMedia and Application Packages - Part3: Still Image, März 2002

[5] ISO/IEC 13249-6: Information Technology - Database Language - SQL MultiMedia and Application Packages - Part3: Data Mining, April 2000

[6] Knut Stolze, SQL/MM Part 5: Still Image -The Standard and Implementation Aspects-, BTW 2001: 345-363

[7] Knut Stolze: SQL/MM Part 5: Still Image - An Introduction. Datenbank Rundbrief 25: 18-29 (2000)

[8] Jim Melton, Andrew Eisenberg: SQL Multimedia and Application Packages (SQL/MM), SIGMOD Record 30 (4): 97-102 (2001)

[9] Open GIS Consortium Inc., OpenGIS Simple Feature Specification, Revision 1.1, Mai 1999

[10] Open GIS Consortium Inc., Geography Markup Language (GML) 2.0, Februar 2001