Seminar
Multimediale Informationssysteme

- ORDBMS mit Multimediaerweiterungen -

Sven Welte

1. Juli 2002

Inhaltsverzeichnis

1 Einführung

Waren früher die Benutzeroberflächen und Darstellungsmöglichkeiten eines Computers im Wesentlichen auf Text beschränkt, so hat sich dies mit der Entwicklung grafischer Benutzeroberflächen grundlegend geändert. Heute können mit jedem handelsüblichen PC multimediale Dokumente wie Bilder, Tonaufnahmen und Videos dargestellt und bearbeitet werden, für die früher noch spezielle und teure Workstations benötigt wurden. Durch neue Übertragungstechniken wie UMTS ist es sogar möglich, auf multimediale Dokumente mobil zuzugreifen.

Mit diesen neuen Möglichkeiten entsteht aber gleichzeitig auch das Problem, dass diese Daten irgendwo gespeichert *und* wiedergefunden werden müssen. Dies stellt für die Bilder einer kleinen Diasammlung sicher kein Problem dar, da diese in einem einfachen Verzeichnisbaum abgespeichert werden könnten. Für eine große Versicherung, die jeden Tag bis zu 20000 Dokumente versendet, die vorher alle digitalisiert werden, müssen andere Formen der Datenspeicherung verwendet werden.

Eine Möglichkeit, große Datenmengen zu speichern, besteht in der Verwendung von relationalen Datenbankverwaltungssystemen (RDBMS), die in der Praxis eine weite Verbreitung erfahren haben und eigentlich in jedem größeren Unternehmen zu finden sind. Diese RDBMS bieten eine recht gute Unterstützung für einfache Attribute wie z.B. CHAR, INT und teilweise auch die Möglichkeit, große Objekte (sog. LOBs) zu speichern. Inzwischen bieten einige Hersteller objektrelationaler Datenbankverwaltungssyseme (ORDBMS) mit speziellen Erweiterungsmodulen eine bessere Integration multimedialer Daten in ihre Systeme an.

Diese Arbeit beschäftigt sich mit der Funktionalität, die diese speziellen Erweiterungsmodule im Einzelnen bieten. In Kapitel 2 wird ein Überblick über große Objekte gegeben und erklärt, warum ORDBMS im Gegensatz zu RDBMS besser geeignet sind, multimediale Daten zu verwalten. Kapitel 3 befasst sich mit der Möglichkeit, benutzerdefinierte Datentypen als “first Class Citizens” in ein DBS einzubringen. Kapitel 4 soll einen Überblick über die von den verschiedenen Erweiterungsmodulen zu Verfügung gestellten Funktionalität geben. Die einzelnen Erweiterungsmodule werden in Kapitel 5-9 dargestellt. Kapitel 10 bietet neben einer kurzen Zusammenfassung einen Überblick über Stärken und Schwächen der einzelnen Module.

2 Große Objekte

Aus Datenbanksicht unterscheidet man strukturierte und unstrukturierte Typen. Ein strukturierter Typ wie z.B. ein Angestellter kann in seine einzelnen Attribute (z.B. PNR, NAME, VORNAME, GEBDATUM,...) zerlegt werden und hat einen Platzbedarf von vielleicht einigen hundert Byte. Unstrukturierte Typen haben hingegen die Eigenschaft, tendentiell sehr viel mehr Platz zu beanspruchen und können von dem Datenbanksystem erst einmal nur als langer Bitstring gespeichert werden. Diese Bitstrings, auch Large Objects (LOBs) genannt, können in zwei Klassen unterteilt werden:

- CLOB: Character Large Object für Textdaten
- BLOB: Binary Large Object für Binärdaten aller Art

Wir werden uns hier mit dem letzteren etwas genauer auseinandersetzten, da dieser Typ für Multimediadaten die größte Bedeutung hat.

Nahezu alle großen RDBMS Hersteller boten bereits zu Zeiten von SQL92 irgendeine Unterstützung von BLOBs. Eine Standardisierung von BLOBs erfolgte aber erst mit SQL:1999. Mit SQL:1999 sind auch weitgehende Erweiterungen im Typsystem vorgenommen worden. Dieses reichhaltigere Typsystem ermöglicht es jetzt, zusammengesetzte Typen zu konstruieren und mit Hilfe von benutzerdefinierten Funktionen Operationen auf diesen Typen durchzuführen, was letztendlich die Schaffung von abstrakten Datentypen (ADTs) ermöglicht. Somit wäre ein BLOB nicht mehr als Black Box aufzufassen, sondern könnte mit benutzerdefinierten Funktionen zu einem abstrakten Datentyp aufgewertet werden.

Innerhalb der Datenbanken wird hinsichtlich des Speicherorts zwischen internen und externen LOBs unterschieden. Interne LOBs werden innerhalb der Datenbank abgespeichert und können (mit

Interne LOBs:	Externe LOBs
+ ACID (mit kleinen Einschränkungen)	- kein ACID
+ Zugriffsoptimierung durch DBVS möglich (z.B. Pufferung, bessere Speicherungsstrukturen)	o Zugriffsoptimierung durch Betriebssystem
- Zugriff nur über DBVS-API	+ Bestehende Anwendungen können LOBs nutzen ohne DBVS-Zugriff
o So sicher wie DBVS	o So sicher wie Betriebssystem
+ in SQL:1999 spezifiziert	- keine formale Spezifikation

Tabelle 1: Unterschied zwischen internen und externen LOBS [ORA01c]

einigen Einschränkungen) auf dieselben Vorteile eines DBMS zurückgreifen wie strukturierte Typen. Externe LOBs verweisen auf eine Lokation außerhalb der Datenbank (z.B. Datei im Dateisystem, Webadresse). Dies ist insbesonders deshalb von großer Relevanz, da so Daten aus bestehenden Anwendungen in das DBVS integriert werden können. Die Tabelle 1 bietet einen Überblick über die Unterschiede zwischen internen und externen LOBs. Eine formale Spezifikation der Einbindung externer Daten ist erst mit dem DataLink-Konzept des kommende Standards SQL/MED zu erwarten. Die einzelnen Ausprägungen von externen LOBs und Weiterführendes zu internen LOBs sind in den einzelnen Handbüchern der Datenbankhersteller zu finden [INF99b, IBM00b, ORA01c].

3 Erweiterungsmodule für Datenbanksysteme

Wie wir bereits im Kapitel "Große Objekte" gesehen haben, bieten die ORDBMS mit der Einführung von SQL:1999 die Möglichkeit, abstrakte Datentypen zu definieren. Damit diese ADTs auch performant arbeiten, werden häufig zusätzliche spezielle Indexstrukturen oder eine andere Anfragebearbeitung benötigt, die vom Datenbanksystem nicht zur Verfügung gestellt werden. Deshalb bieten einige Datenbanken eine Erweiterungsschnittstelle, über die ein Erweiterungsmodul auf low-level Funktionalität in der Datenbank zugreifen kann. Eine Vorstellung der generellen Architektur bietet Abbildung 1.

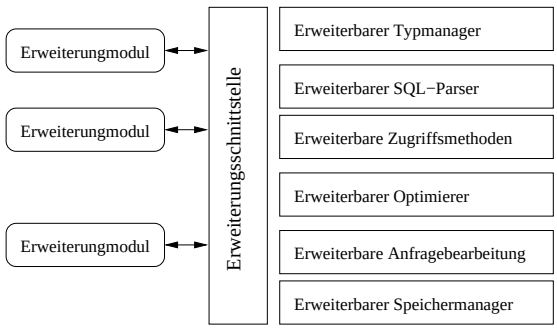

Abbildung 1: Eine mögliche Anbindung von Erweiterungsmodulen [TÜR01]

Eine Erweiterung der Datenbank um neue Funktionaliät erfolgt in der Regel über eine spezielle Erweiterungsschnittstelle. Über diese Schnittstelle kann auf einen erweiterbaren Typmanager zugegriffen werden, der es erlaubt, bestehende Typen zu erweitern und neue Typen anzulegen. Weiterhin muss die Möglichkeit bestehen, den SQL-Parser um neue Konstrukte zu erweitern. Im Allgemeinen werden lediglich neue benutzerdefinierte Funktionen in das System eingebracht, was bedeutet, dass die SQL-Syntax, die der Parser parsen muss, gleich bleibt. Die erweiterbaren Zugriffsmethoden in

Verbindung mit dem erweiterbaren Optimierer sind vermutlich die wichtigsten Erweiterungen. So legt die Anwendung beispielsweise das Format und den Inhalt eines neuen Indexes fest, wohingegen das DBS für die Speicherung des Indexes zuständig ist und damit ACID garantieren kann (bei korrektem Anwendungscode). Eine Auswertung des Indexes bei Anfragen erfolgt durch das benutzerdefinierte Erweiterungsmodul, welches auch die Verwaltung des Indexes (z.B. Tupel gelöscht, eingefügt) übernimmt. Der erweiterbare Optimierer erhält von der diesem Modul Informationen über die Selektivität eines Indexes, Statistikinformationen und Kostenfunktionen. Ein erweiterbarer Speichermanager ermöglicht es, Daten auf unterschiedliche Art und Weise zu speichern. So kann eine Anforderung an die Speicherung eines LOB sein, dass das LOB an beliebigen Stellen geändert werden können muss, ohne dass es zu Geschwindigkeitseinbußen kommt. Bietet das DBS nur das schnelle Ändern am Ende von LOBs an, so könnte durch eine veränderte benutzerdefinierte Speicherungsstruktur diese Anforderung erfüllt werden.

Nahezu alle Datenbankhersteller bieten eine Form von Erweiterungsschnittstelle an, mit der ihre Datenbanksysteme erweitert werden können. DB2 kann durch sog. Extender erweitert werden. Informix erlaubt durch seine DataBlades eine Integration neuer Funktionalität in das DBS [INF00]. Oracle kann durch DataCatridges erweitert werden [ORA02].

Im Folgenden wird ein Überblick über die verschiedenen Erweiterungen gegeben, die sich von den großen Herstellern objektrelationaler Datenbanken für die eigenen Datenbanksysteme im Augenblick am Markt befinden. Alle hier vorgestellten Systeme ermöglichen es, multimediale Inhalte als “first Class-Citizens” in den Datenbanksystemen zu verwenden. Microsoft bietet als ein weiterer großer Datenbankhersteller in seinem SQL-Server zwar einen Datentyp Image an, dieser Datentyp bietet aber keine weitergehenden Funktionen an, die über die Funktionalität eines BLOBs hinausgehen würden.

4 Architektur der vorgestellten Systeme

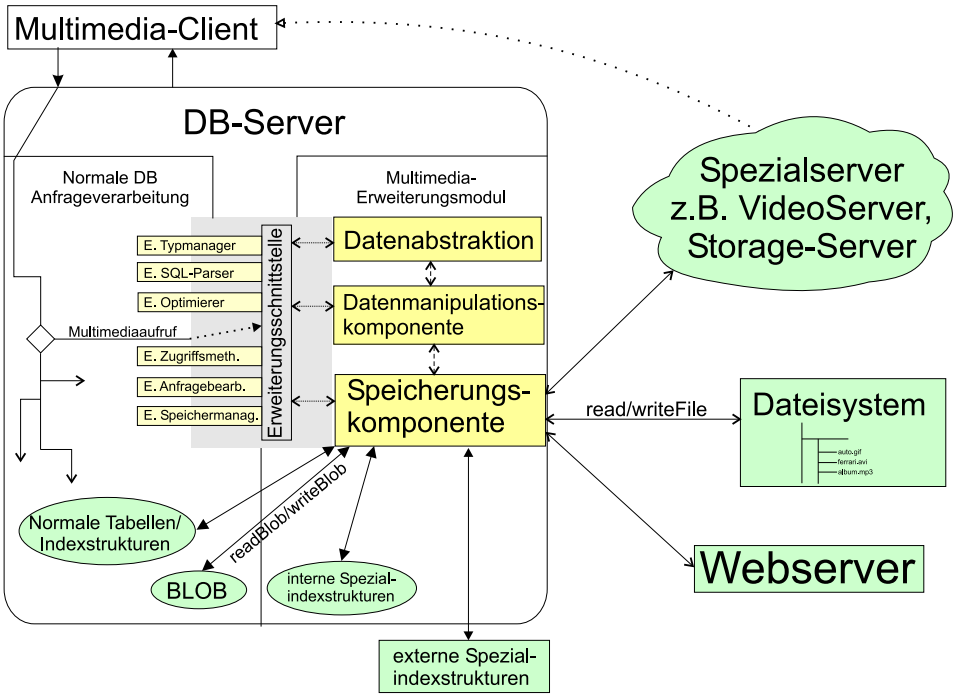

Abbildung 2: Überblick eines DB-Servers mit Multimedia-Erweiterungsmodul

Im Folgenden soll ein Überblick über den generellen Aufbau der vorgestellten Systeme gegeben werden. Dieser Aufbau, der in Abbildung 2 grafisch dargestellt ist, stellt eine *mögliche* Einteilung dar. Das soll nicht bedeuten, dass jedes System komplett mit diesem Aufbau übereinstimmt, gemäß diesem Aufbau implementiert ist oder dass alle Komponenten enthalten sind. Der Aufbau stellt lediglich eine Art Orientierungshilfe dar, an Hand derer jetzt die *Funktionalität* genauer erläutert wird.

Wenn ein Multimedia-Client eine Anfrage an den DB-Server stellt, so wird innerhalb der normalen DB-Anfrageverarbeitung entschieden, ob es sich um eine Anfrage handelt, die auf Multimediadaten oder normale relationale Datenstrukturen zugreift. Bei einem Multimediaaufruf wird die Anfrage an das Multimedia-Erweiterungsmodul weitergeleitet, das über die Erweiterungsschnittstelle (siehe Kapitel 3) in den DB-Server eingebunden wurde. Innerhalb des Multimedia-Erweiterungsmoduls sind hier die drei funktionalen Teilbereiche, nämlich Datenabstraktion, Datenmanipulationskomponente und Speicherungskomponente dargestellt. Ein solches Erweiterungsmodul stellt häufig zusätzliche Funktionalität im Bereich der Anfrageoptimierung und der inhaltsbasierten Suche bereit, auf die hier nicht näher eingegangen wird.

Datenabstraktion: Die Datenabstraktion bzw. Datenunabhängigkeit ist eine sehr wichtige Eigenschaft von Datenbanksystemen, die sehr zu deren Verbreitung beigetragen hat. Alle vorgestellten Systeme bieten die Möglichkeit, auf Attribute von gespeicherten Multimediadaten zuzugreifen (z.B. Höhe, Breite oder Format eines Bildes). Sind innerhalb der Datenbank Multimediadaten in einem anderen Multimediaformat gespeichert, als der Multimedia-Client verarbeiten kann, so können diese Bilder mit Hilfe von Transformationsfunktionen in das entsprechende Format umgewandelt werden, wodurch eine einfache Form von Datenunabhängigkeit erreicht wird.

Datenmanipulationskomponente: Viele der vorgestellten System erlauben es, innerhalb des DB-Servers Änderungen an Multimediadaten vorzunehmen. Damit wird verhindert, dass die tendentiell großen Multimediadaten für einfache Manipulationsfunktionen jedesmal auf den Client transferiert werden müssen.

Speicherungskomponente: Auf Multimediadaten wird selten nur exklusiv über den DB-Server zugegriffen. Daten, auf die nur exklusiv zugegriffen wird, können in der DB als BLOB gespeichert werden und bieten alle Vorteile die für interne BLOBs gelten (siehe Kapitel 2). In der Regel werden Multimediadaten aber von mehreren Applikationen benötigt, die nicht alle unbedingt über den DB-Server auf diese Daten zugreifen können. Hierzu bietet die Speicherungskomponente Integrationsmöglichkeiten an, um externe Datenquellen anzubinden, z.B. Webserver, Dateisysteme oder Spezialserver. Bei den Spezialservern (z.B. Videoserver) besteht bei einer Anfrage über den DB-Server die Möglichkeit, die Daten direkt an den Client auszuliefern, ohne den Weg zurück über den DB-Server nehmen zu müssen (siehe Abschnitt 9.3).

Der Verweis auf die gespeicherten Daten erfolgt im Optimalfall vollkommen transparent, d.h. der Client braucht sich beim Datenzugriff nicht darum zu kümmern, wo die Daten gespeichert sind. Hierdurch wird auch eine gewisse Art der Datenlokationsunabhängigkeit geboten.

Zusätzlich zu diesen Möglichkeiten muss die Speicherungskomponente externe oder interne Spezialindizies verwalten, um eine inhaltsbasierte Suche über den Multimediadaten zu ermöglichen. Hierbei können Daten, die die Suche und den Zugriff auf Attribute von Multimediadaten benötigt werden, in Verwaltungsunterstützungstabellen gespeichert werden.

5 IBM DB2 Image, Audio und Video Extender

Mit dem DB2 Image, Audio und Video Extender können die drei wichtigsten Multimediadatenarten verwaltet werden. Hierzu verwendet DB2 die Datentypen DB2Image, DB2Audio und DB2Image auf die im Abschnitt 5.1 eingegangen wird. Über DB2Image kann auf Attribute von gespeicherten Bilddaten zugegriffen werden und inhaltsbasierte Abfragen auf dem Bildinhalt durchgeführt werden. Zusammen mit den Bildmanipulationsmöglichkeiten wird dies in Abschnitt 5.2 behandelt. Einen Über-

blick über die nicht besonders ausgeprägte Audiofunktionalität des Extenders verschafft Abschnitt 5.3. Der Teil 5.4 zeigt die Möglichkeit auf, Storyboards für Videosequenzen zu erstellen.

Informationen zu diesem Extender sind aus den Handbüchern von DB2 [IBM00a, IBM00b] entnommen. Die Betrachtung für Bilder im Besonderen stammt aus [STO02].

5.1 DatenTypen

Für die drei Extender gibt es die drei entspechenden Datentypen *DB2Image*, *DB2Audio* und *DB2Video*. Der Datentyp *DB2Image* wird verwendet, um das eigentliche Bild und die damit verbundenen Attribute zu speichern. Ein solches *DB2Image* stellt aber lediglich eine Art Zugriffshandle auf die eigentlichen Daten dar, die in sogenannten Verwaltungsunterstützungstabellen (VUT) untergebracht sind. Diese VUT müssen vom Datenbankadministrator angelegt werden, bevor Daten vom Typ *DB2Image* in die Datenbank eingefügt werden. Ein Beispiel für den prinzipiellen Ablauf:

```
(0) ENABLE DATABASE FOR DB2Image;
```

Dieser Befehl bereitet eine Datenbank auf die Verarbeitung von Bilddaten vor. Dies beinhaltet das Anlegen von Hilfstabellen und benutzerdefinierter Funktionen (UDFs).

```
(1) ENABLE TABLE Person FOR DB2Image;
```

Dies fügt die Verwaltungsunterstützungstabellen für die Tabelle `Person` hinzu.

```
(2) ENABLE COLUMN Person passbild FOR DB2Image;
```

Legt die benötigten VUT für das Passbild an und fügt der Spalte `passbild` Trigger hinzu, die eine Aktualisierung dieser VUT gewährleisten, wenn sich das Passbild in der Tabelle `Person` ändert. Für die Datentypen *DB2Audio* und *DB2Video* werden nach dem selben Schema Verwaltungsunterstützungstabellen angelegt.

5.2 Spezielle Funktionen zur Bildverwaltung mit dem Image Extender

Über *DB2Image* kann auf die Bildattribute wie Bildformat, Höhe, Breite und Tiefe (Anzahl der Farben) zugegriffen werden sowie auf zusätzliche Attribute, die für die Verwaltung oder für beschleunigte Abfragen wichtig sind (thumbnail, importer, importTime, updater, updateTime, fileName und comment). Alle diese Attribute können über spezielle UDFs abgefragt werden, welche *DB2Image* dann als Parameter verwenden. Der eigentliche Bildinhalt kann entweder in einer externen Datei oder als BLOB in der Verwaltungsunterstützungstabelle gespeichert werden. Die von dem Image Extender zurückgegebenen Thumbnails haben alle dieselbe Größe, was bedeutet, dass das Thumbnail beim Einfügen/Verändern des eigentlichen Bildes mitgeliefert werden muss, wenn eine unterschiedliche Größe gewünscht ist.

5.2.1 Bildmanipulation

Mit dem DB2 Image Extender ist es auf verschiedene Arten und Weise möglich, den Bildinhalt zu verändern. Dies geschieht aber nicht durch verschiedene separat implementierte benutzerdefinierte Funktionen (UDF), die orthogonal miteinander verwendet werden könnten, sondern durch zusätzliche Parameter, die beim INSERT oder UPDATE eines *DB2Image* übergeben werden. Diese zusätzlichen Parameter erinnern etwas an die Parameter, die einem Programm beim Kommandozeilenaufruf übergeben werden (siehe Tabelle 2).

```
INSERT INTO Person (id, passbild) VALUES (007,
       DB2Image (CURRENT SERVER, -- aktuellen Server verwenden
       '/home/bilder/roger.bmp', -- wo liegt Datei auf Server
                'BMP',                    -- aktuelles Dateiformat
                'GIF',                    -- in GIF abspeichern/konvertieren
                '-x 300 -y 200 -r 1'      -- Manipulationsoperation
                ' ',    -- in welcher Datei ablegen bzw. ' ' -> in BLOB speichern
                'Roger Moore Einstellungsphoto' -- Kommentar
        ));
```

Dieser SQL-Befehl fügt das Bild, das sich auf dem Server im Verzeichnis '/home/bilder' befindet in die Datenbank ein. Während des Einfügens wird das Bild in das gif-Format konvertiert, auf die Maße x=300 und y=200 skaliert und abschließend um 90 Grad gedreht. Es ist beim Einfügen jedes Mal notwendig, alle Parameter mit anzugeben, auch wenn keine Manipulation des Bildes beabsichtigt ist.

Parameter	Funktionalität
-s	Skalieren auf einen Wert (0..1 bzw. 0 bis 100%)
-p	Bild invertieren
-r	Rotieren (0 = 0°, 1=90° (gegen Uhrzeigersinn, 2=90° (mit dem Uhrzeiger.), 3=180°)
-y	Höhe in Pixeln
-x	Breite in Pixeln
-c	Komprimierung festlegen (z.B. 4=CCITT Gruppe4, 10=unkomprimiert, 14=LZW)

Tabelle 2: Mögliche Parameter und deren Funktionalität [IBM00a]

Auf den zusätzlich mit dem Bild gespeicherten Informationen kann über verschiedene Funktionen zugegriffen werden z.B.

```
SELECT COMMENT (passbild) FROM Person WHERE id=007;
-- Neuen Kommentar festlegen
UPDATE Person SET passbild=COMMENT (passbild, 'Ab sofort streng geheim') WHERE id=007;
```

Eine Abfrage oder Änderung des Bildinhaltes kann über die Funktion CONTENT realisiert werden:

```
SELECT  CONTENT (
   Passbild,               -- Handle auf das Foto
   '/tmp/neuerausweis.gif', -- Ergebnis der Anfrage speichern in Datei auf Server
   1)                      -- Datei darf überschrieben werden
FROM Person WHERE id=007;
UPDATE Person SET Passbild =CONTENT (
        Passbild,                    -- Handle auf das Foto
        '/home/photos/t_dalton.gif',  -- Neues Foto
        'ASIS'                       -- Bildformat beibehalten
        '')                         -- Bild als BLOB in der DB speichern
FROM Person WHERE id=007;
```

5.2.2 Möglichkeiten der Bildabfrage

Nun kann neben den einfachen Funktionen wie z.B. COMMENT, FILENAME, HEIGHT, WIDTH, die im Wesentlichen nur auf die Metadaten des beim Einfügen abgespeicherten Bildes zugreifen, auch eine inhaltsbasierte Suche auf den Bildern durchgeführt werden. Dazu muss als Voraussetzung ein sogenannter QBIC-Katalog angelegt werden (QBIC= Query By Image Content), der außerhalb der Datenbank im Dateisystem gespeichert wird. Dies ist damit begründet, dass in der ersten Version des Image Extenders von DB2 noch keine Unterstützung für erweiterbare Indexstrukturen angeboten wurde. In diesen Katalog werden alle diejenigen Bilder indexiert, über die Suchabfragen durchgeführt werden, was entweder automatisch geschieht oder manuell durchgeführt werden kann. Mit der API, die der Image Extender zur Verfügung stellt, können diese Kataloge auch im Clientdateisystem angelegt werden und von dort für Abfragen genutzt werden. Eine Pflege eines solchen Client-Kataloges kann natürlich nur manuell erfolgen, da keine Verbindung zur Datenbank besteht. Eine Sicherung der Kataloge muss aber in jedem Fall von Hand vorgenommen werden, da diese nicht Teil des Datenbankbackups/restore sind und auch kein ACID dafür garantiert wird. Ein Katalog kann die folgenden Merkmale zu einem Bild speichern: Durchschnittsfarbe (QbColorFeatureClass), Histogrammfarbe (QbColorHistogramClass), positionsgebundene Farbe (QbDrawFeatureClass), Textur (QbTextureFeatureClass).

Über diesen Merkmalen ist mit den Funktionen *QbScoreFromName*, *QbScoreTbFromName*, *QbScoreFromStr* und *QbScoreTbFromStr* eine (gewichtete) Bewertung eines Bildes oder mehrerer Bilder möglich. Eine Abfrage ordnet jedem Bild eine Bewertungszahl zu, die darüber Auskunft gibt, wie weit das gewünschte Zielfoto mit dem aktuellen Bild übereinstimmt. Ein hoher Wert kennzeichnet eine hohe Übereinstimmung, ein niedriger Wert entspechend eine niedrige Übereinstimmung. Bei einer Abfrage über mehrere Bilder entsteht so ein Ranking, bei dem die interessantesten Bilder die höchsten Bewertungszahlen haben und damit am weitesten oben stehen. Die Formulierung einer Abfrage erfolgt -ähnlich wie bei den Manipulationsfunktionen- nicht über Funktionen, die orthogonal zueinander eingesetzt werden können, sondern über einen String, in dem die gewünschten Merkmale beschrieben sind und der dann z.B. der Funktion *QbScoreFromStr* übergeben wird. Hierzu ein Beispiel, das die Tabellenfunktion *QbScoreTbFromStr* verwendet:

```
SELECT image_id FROM TABLE (QBScoreTbFromStr (
       'texture file=<server,fotos/fahndungsfoto.gif>', --Eigentliche Abfrage
       'person',       -- Suche in Tabelle Person
       'passbild'      -- Benutze Spalte Passbild
       5       -- Maximal 5 Bilder zurückgeben
)) AS T1
```

Diese Anfrage gibt alle Bilder der Spalte `Passbild` in der Tabelle `Person` zurück, die dem Bild 'fotos/fahndungfoto.gif', welches sich auf dem Server befindet, ähnlich sehen. Da es eventuell sehr viele ähnliche Bilder geben kann, wird die maximale Anzahl von Bildern auf 5 begrenzt. Die Spalte `image_id` ist vom Typ `DB2Image`.

5.3 Spezielle Funktionen zur Audiodatenverwaltung

Für Audiodaten gibt es nur eine geringe Unterstützung der inhaltsbasierten Suche. Auch ist es nicht möglich, Audiodaten innerhalb der Datenbank zu verändern.

Die inhaltsbasierte Suche ist recht eingeschränkt und bezieht sich im Wesentlichen darauf, dass bestimmte Charakteristika eines MIDI-Stückes abgefragt werden können. So kann mit der Funktion FindInstrument beispielweise abgefragt werden, ob ein bestimmtes Instrument in einem MIDI-Stück vorkommt oder nicht. Weitergehende Funktionalität wie z.B. die Suche nach bestimmten Rhythmen oder Tonfolgen wird nicht angeboten.

Durch den DB2 Audio Extender ist mit entsprechenden Funktionen ein Zugriff auf die einzelnen audiospezifischen Attribute eines Tondokuments (*DB2Audio*) möglich, die vom AudioExtender automatisch gepflegt werden, wenn ein neues Tondokument in die Datenbank eingefügt wird. Mit diesen Funktionen können die Anzahl der Tonspuren, die Anzahl an Bytes pro Sekunde, die Namen aller verwendeten Instrumente, die Namen aller verwendeten Spuren, die Abtastrate, die Taktgeschwindigkeit oder das verwendete Audioformat abgefragt werden.

5.4 Spezielle Funktionen zur Videodatenverwaltung

Der Video Extender stellt keinerlei Funktionen zur Manipulation von Videodaten zur Verfügung. Ähnlich wie bei *DB2Image* und *DB2Audio* werden mit dem Datentyp *DB2Video* neue Funktionen eingeführt, um auf videospezifische Attribute eines Videos zuzugreifen (z.B. *frameRate*, *compressType*, *Duration*, *heigth*, *width*). Diese Funktionen haben alle denselben Namen wie das abzufragende Attribut und erhalten als Parameter ein *DB2Video*.

5.4.1 Storyboards

Ein Storyboard ist eine Art Kurzüberblick über einen Videoclip und enthält Schlüsselszenen als Standbilder. Mit der recht umfangreichen C-API des Video Extender ist es möglich, solche Schlüsselszenen zu identifizieren und in einem Aufnahmekatalog zusammenzufassen.

Ein Szenenwechsel wird erkannt, wenn ein deutlicher Unterschied zwischen zwei Bildern besteht. Hierzu sind verschiedene Verfahren implementiert, die z.B. auch Überblendungen berücksichtigen.

Vom Anwendungsprogrammierer kann festgelegt werden, welche Länge eine Szene minimal haben muss, um als Szene erkannt zu werden. Mit Hilfe des Video Extenders ist so eine Klassifikation in kurze und lange Szenen möglich. Wenn eine Szene die vorgeschriebene Mindestlänge übersteigt, so wird eine repräsentatives Vollbild ermittelt, welches genau zwischen Beginn und Ende dieser Szene liegt. Dieses Vollbild kann verwendet werden, um den Aufnahmekatalog zu erstellen.

Sollen auf Zeitpunkte innerhalb eines Videoclips zugegriffen werden, so muss dieser davon indiziert werden. Mit dieser Indizierung wird eine Abbildung von dem Bitstrom auf Vollbildnummern vollzogen. Die Indexdatei wird außerhalb der Datenbank bei dem entsprechenden Video gespeichert.

6 IBM Informix Excalibur Image DataBlade

Das Excalibur Image Datablade ist eines von zwei Datablades aus dem Hause Informix (jetzt IBM), das sich mit Bildern in der Datenbank beschäftigt. Es ist eine Entwicklung des Datenbankherstellers Informix in Zusammenarbeit mit der Firma Excalibur, die ihrerseits eine Bibliothek aus dem Bereich Bildverarbeitung eingebracht hat.

Das Excalibur Image Datablade stellt neben den üblichen Manipulationsfunktionen hauptsächlich Funktionen zur inhaltsbasierten Suche zur Verfügung. So beschränkt sich die Funktionalität zur Veränderung von Bildern im Wesentlichen auf die Konvertierungsfunktionen zwischen verschiedenen Grafikformaten, Skalierung bestehender Bilder (z.B. Thumbnails) und Veränderung der Farbzusammensetzung bestehender Bilder (z.B. Farbbild in Graustufen konvertieren).

Informationen über das Datablade entstammen aus dem Benutzerhandbuch zu diesem Datablade [INF99a] und aus [STO02].

6.1 Datentypen

In diesem Datablade werden im Wesentlichen zwei neue Datentypen eingeführt: *IfdImgDesc* für das eigentliche Bild und *IfdFeatVec* für Featurevektoren.

```
CREATE ROW TYPE IfdImgDesc (
          location IfdLocator,
          imgformat varchar (25),
          imgtype varchar (25),
          pixelheight integer,
          pixelwidth integer,
          params varchar(128) );
```

In dem strukturierten Typ *IfdImgDesc* werden neben dem Speicherungsort *IfdLocator* (externe Datei, interner BLOB) zusätzliche Informationen (wie z.B. Höhe oder Breite des Bildes) automatisch mitgepflegt, wenn das Bild manipuliert oder eingefügt wird. Das Attribut params ermöglicht es dem Benutzer, selbst Informationen zu einem Bild abzulegen, da dieses Attribut vom Image DataBlade nicht genutzt wird. Auf alle Attribute kann entweder direkt oder mit Hilfe von entsprechenden Funktionen zugegriffen werden (z.B. `ImgHeight`(), `ImgFormat`()). Der unstrukturierte Datentyp *IfdFeatVec* nimmt Informationen eines Featurevektors auf, der die Ausprägung bestimmter Merkmale eines Bildes (z.B. Farbinhalt, Farbstruktur, Texturinhalt) enthält. Beispiel wie der Datentyp *IfdImgDesc* verwendet werden kann:

```
CREATE TABLE Person (
   id INTEGER,
   passbild IfdImgDesc,
   featureVektor IfdFeatVec );

-- Fügt ein Bild in die Datenbank ein, das extern gespeichert wird
INSERT INTO Person (id, passbild) VALUES (007, IfdImgDescFromFile ('/home/foto/007.gif'));
```

6.2 Bildmanipulation

Wie bereits erwähnt, unterstützt das Excalibur Image DataBlade nur sehr wenige Manipulationsfunktionen, die hier in Beispielform komplett dargestellt werden:

```
-- Bild in ein anderes Format konvertieren
UPDATE Person SET passbild = ConvertImgFormat (passbild, 'TIFF');
-- Den Bildtyp verändern
UPDATE Person SET passbild = ConvertImgType (passbild, 'CTAB');
-- Das Bild auf 150% hochskalieren und dabei einen langsamen Algorithmus
-- verwenden, der aber aber (H)ohe Quoalität liefert
UPDATE Person SET passbild = ScaleImgBy(passbild, 1.5, 'H');
-- Das Bild auf Breite 100 Höhe 200 skalieren..
-- schnellen Algorithmus -> Bilder schlechter Qualität  ( (l)ow quality)
-- Seitenverhältnisse beibehalten(T)
UPDATE Person SET passbild = ScaleImgTo (passbild, 200, 300, 'L', 'T');
```

Alle Funktionen können vollkommen orthogonal zueinander verwendet werden.

6.3 Inhaltsbasierte Suche

Im Gegensatz zu den doch recht engeschränkten Bildmanipulationsmöglichkeiten bietet das Excalibur Image DataBlade Modul eine recht breite Unterstützung für inhaltsbasierte Suche an. Bevor jedoch eine solche Suche durchgeführt werden kann, muss zu erst aus jedem Bild, das sich im gewünschten Suchraum befindet, ein Featurevektor extrahiert werden. Dies geschieht mit der Funktion `getFeatureVector` ().

```
UPDATE Person SET featureVektor = getFeatureVector (passbild).
```

Das Feld `featureVektor` enthält jetzt Informationen über den Farbinhalt des Bildes (bzw. ein Farbhistogramm), die Farbstruktur (sagt, wo auf dem Bild welche Farbe zu finden ist), die Textur einfacher geometrischer Formen (sagt, welche Linien es gibt und welche Ausrichtung diese haben), Helligkeitsstruktur (gibt Auskunft über die Verteilung der Helligkeit über alle Pixel) und die Seitenverhältnisse eines Bildes (z.B. 4:3 oder 16:9).

Für eine Pflege der Featurevektoren ist ganz allein der Benutzer bzw. der Anwendungsprogrammierer zuständig, da im Datenbanksystem nicht von Anfang an festgelegt ist, wo und wann Featurevektoren gespeichert sind. Hat sich also ein Bild durch eine Manipulationsfunktion geändert, darf nicht vergessen werden, den dazugehörigen gespeicherten Featurevektor zu erneuern. Eine automatische Pflege der Featurevektoren wäre z.B. mit Triggern zu erreichen. Das Excalibur Image Datablade bietet keinerlei Indexunterstützung für den DatenTyp *IfdFeatVec*, was bedeutet, dass je nach Selektivität des Abfrageprädikats jeder gespeicherter Featurevektor mit dem gesuchten Featurevektor verglichen werden muss, was zu einem Tablescan führen könnte.

Für die eigentliche Abfrage existiert in dem Excalibur Image DataBlade die Funktion Resembles, mit der zwei Featurevektoren gewichtet miteinander verglichen werden können und die als Ergebnis einen Wahrheitswert zurückgibt, aus dem abzulesen ist, ob die beiden Featurevektoren innerhalb eines Schwellwertes als gleich anzusehen sind. Die Gewichtung erfolgt mit der zusätzlichen Übergabe von sechs Gewichtungsfaktoren, die zueinander relativ sind, aber zusammen nicht mehr als 100 ausmachen dürfen, wobei es sich hierbei nicht um Prozentangaben handelt. Eine solche Abfrage könnte zum Beispiel so aussehen:

```
SELECT verraeter.id, rank FROM Person verraeter, Person suchfoto
WHERE Resembles ( verraeter.featureVektor,
                  suchfoto.featureVektor,
                  0.80,   -- Schwellwert von 80% muss überschritten werden
                  1,      -- Gewichtung Farbinhalt
                  1,      -- Gewichtung Geometrische Formen
                  1,      -- Gewichtung Textur
                  0,      -- Gewichtung Helligkeitsstruktur
                  0,      -- Gewichtung Farbstruktur
                  0,      -- Gewichtung Seitenverhältnisse
                  rank #REAL)
      AND suchfoto.id=1234 ORDER BY rank;
```

7 IBM Informix Image Foundation DataBlade

Das Image Foundation DataBlade stellt im Gegensatz zu dem Excalibur Image DataBlade nur Funktionalität im Bereich der Bildmanipulation zur Verfügung. Dies ist damit begründet, dass das Image

Feldname	Beschreibung	Beispiel
Scheme	Methode um auf die Ressource zuzugreifen (CLOB, BLOB oder FILE)	file
Authority	Gültige Werte sind leerer String -> CLOB, BLOB auf aktuellem Server gespeichert Servername auf dem FILE gepeichert ist	moorhuhn
Path	Dateipfad auf dem Dateiserver oder Beschreibung der Tabelle/Spalte im Datenbankserver	/images/logo.jpg
Query	String, der von der ressource-authority interpretiert wird	?FMT=image/gif
Fragement	nicht von Informix unterstützt	

Tabelle 3: Beschreibung der Felder einer URI [INF00]

Foundation DataBlade als Basis für weitere DataBlades oder Entwicklungen dienen soll, welche im Allgemeinen mit Bildern in einer Datenbank umgehen. So werden eine Reihe von Bildmanipulationsfunktionen und Konvertierungsfunktionen geboten, die über die Fähigkeiten des Excalibur Image DataBlade weit hinausgehen. Eine Integration des Image Foundation Datablade in das Excalibur Image DataBlade oder umgekehrt ist nicht vorhanden, was bedeutet, dass z.B. die Grunddatentypen für Bilder nicht zueinander kompatibel sind und vor Benutzung erst in das entsprechende andere Format umgewandelt werden müssen. Bei dem Image Foundation DataBlade ist es dem Anwendungsprogrammierer möglich, noch nicht unterstützte Bildformate durch einen Erweiterungsmechanismus nachzurüsten.

Die verwendeten Informationen entstammen hauptsächlich aus dem Benutzerhandbuch zu diesem Datablade [INF00] und aus [STO02].

7.1 Datentypen

Das Image Foundation DataBlade benutzt den Datentyp *URI* (Uniform Resource Identifier) um Bilder zu referenzieren und Operationen auf diesen Bildern anzuwenden. Eine *URI* hat die generelle Form `scheme://authority path?query#fragments`. Die Bedeutung der einzelnen Felder ist in Tabelle 3 dargestellt.

Der Datentyp *IfxImage* ist ein strukturierter Typ und ist wie folgt definiert:

```
CREATE ROW TYPE IfxImage (
   location URI,           -- URI
   format lvarchar,        -- Typ des Bilder als MIME typ z.B. image/jpeg
   pixeltype lvarchar,     -- Pixeltyp z.B. UI8
   colorspace lvarchar,    -- Farbmodell des Bildes z.B. RGB
   compression lvarchar,   -- Kompressionsformat z.B. LZW
   width integer,          -- Breite in Pixeln
   heigth integer);        -- Höhe in Pixeln
```

Dieser Datentyp repräsentiert die Bilder in der Datenbank. Es existiert eine Funktion, mit der sich die einzelnen Attribute abfragen lassen. Die URI wird verwendet, um auf den eigentlichen Inhalt des Bildes zuzugreifen. Mit der gleichnamigen Funktion IfxImage kann eine neu Instanz des Typs *IfxImage* erzeugt werden, um diese z.B. in einer INSERT-Anweisung zu verwenden. Die Funktion `NewImage/NewIfxImage` erstellt ein neues Bild an einer durch eine URI spezifizierten Position und gibt als Ergebnis die URI des neu erstellten Bildes zurück. Der Unterschied zwischen `NewImage` und `NewIfxImage` besteht darin, dass die erste Funktion eine URI für das Quellbild als Parameter erwartet, wohingehen die Funktion `NewIfxImage`, wie der Name schon sagt, ein *IfxImage* erwartet.

7.2 Bildmanipulation

Das Image Foundation DataBlade stellt Funktionen bereit, um Bilder durch eine Menge an Kommandos, die im Internet Imageing Protocol (IIP) genauer spezifiziert ist, zu verändern. Dies geschieht

durch die Funktion `CVT`, die als Parameter einen Kommandostring erwartet, der festlegt, welche Operationen an einem Bild durchgeführt werden sollen. Im Folgenden ein Überblick über die möglichen Kommandos:

AFN: wendet eine affinen Transformation auf das Bild an

BGCOLOR: ändert die Hintergrundfarbe des Bildes

BOX: erstellt einen Rahmen, der in das Bild eingebettet wird. Eine Skalierung wird automatisch durchgeführt

CNT: passt den Kontrast an

CTW: führt Farbkorrektur durch

CUT/RGN: schneidet einen bestimmten Teil des Bildes aus

FTR: schärft das Bild oder zeichnet es weich

HEI/WEI: legt Höhe/Breite des Bildes fest

RAR: legt das Seitenverhältnis des Zielbildes fest

ROT: dreht das Bild

ZOOMX/ZOOMY: skaliert das Bild, ohne die Seitenverhältnisse des Bildes beizubehalten

Alle diese Kommandos können beliebig miteinander kombiniert werden, wobei manche Kombinationen nicht sinnvoll sind (z.B. BOX und HEI+WEI, HEI und ZOOMY). Im Folgenden ein Beispiel, bei dem eine auf der Festplatte befindliche Datei 'image.jpg' mit einem Rahmen versehen wird und als GIF in die Datenbank gespeichert wird:

```
CREATE Person (  id integer,     passbild IfxImage );
INSERT INTO Person (id, passbild) VALUES (007,
              IfxImage(
                  NewImage(CVT ('file://tmp/image.jpg',
                               'BOX=500,500'),
                               'blob:///?FMT=image/gif'))));
```

In diesem Fall wird nur ein Rahmen um das Bild gelegt, was mit einem recht einfachen Kommando möglich ist. Werden aber andere Kommandos in Kombination verwendet oder sind sie komplexer, so wird der Kommandostring recht schnell unübersichtlich. So benötigt die affine Transformation (AFN) als Parameter beispielsweise eine 4x4 Matrix (16 Werte mit Komma getrennt).

8 Oracle 9i interMedia

Die Oracle 9i interMedia Option ist eine Weiterentwicklung der beiden separaten Produkte Oracle 8i interMedia Catridge und Oracle 8i Visual Information Retrieval, welche die Bereiche Bildmanipulation, Bildkonvertierung und inhaltsbasierte Bildsuche abdeckt. Es ist auch möglich, mit Video und Audiodaten zu arbeiten, doch bei diesen Multimediatypen ist die Funktionalität auf Konvertierungsfunktionen und einfache zusätzliche Metadaten, die mit einem Autio/Videodokument gespeichert werden, beschränkt.

Die verwendeten Informationen stammen aus den Oracle Handbüchern [ORA01b, ORA01c]. Eine genauere Betrachtung von Bildern in der interMedia Option ist in [STO02] zu finden.

8.1 Datentypen

Mit der interMedia Option werden die neuen Daten bzw. Objekttypen *ORDAudio*, *ORDImage*, *ORDVideo* und *ORDDoc* eingeführt, welche für die Speicherung von Ton, Bild und Videodaten verwendet werden. Hierbei nimmt der Typ *ORDDoc* eine Sonderrolle ein, da mit diesem beliebige multimediale Datentypen gespeichert werden können (z.B. Postscript, XML, Bilder, Audio). Vom Prinzip unterscheidet sich *ORDDoc* damit nicht sehr von einem einfachen BLOB, nur kann ein *ORDDoc* einfacher um benutzerdefinierte Formate erweitert werden und verfügt über deutlich weitergehende Funktionen zur Speicherung/Ablage der eigentlichen Daten an verschiedenen Speicherungsorten. Alle Objekttypen haben ein Attribut von Typ *ORDSource*, das eine gewisse Abstraktion bietet, wo die eigentlichen Daten gespeichert werden. So können die entsprechenden Daten entweder als BLOB in der Datenbank, als *BFILE* im Dateisystem oder mit Hilfe einer URL auf einem Webserver gespeichert sein. Zu guter Letzt gibt es, ähnlich wie bei den Informix Lokatoren, die Möglichkeit, benutzerdefinierte Quellen hinzuzufügen. Weiterhin verfügen alle Multimedia-Objekttypen über die Attribute *format* und *mimeType*, wobei auf alle Attribute mit entsprechenden get/set Methoden zugegriffen werden kann. Eine genaue Beschreibung der gesamten vorhandenen Methoden kann hier nicht gegeben werden, da jeder der Objekttypen über einen sehr reichhaltigen Methodenschatz verfügt (beispielsweise hat *ORDImage* mehr als 30 Methoden).

- *ORDAudio* verfügt über einige zusätzliche audiospezifische Attribute: *encoding*, *numberOfChannels*, *sampingRate*, *compressionType* und *audioDuration*
- *ORDImage* verfügt zusätzlich über: *heigth, width, contentLength, contentFormat, compressionFormat*
- ORDVideo verfügt zusätzlich über: *width, heigth, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, bitRate*

8.2 Manipulation von Multimediadaten

Für *ORDAudio*, *ORDVideo* und *ORDDoc* existieren keine Möglichkeiten der Manipulation. Die beiden Objekttypen *ORDAudio* und *ORDVideo* besitzen zwar Methoden wie *processAudioCommand* und *processVideoCommand*, das eigentliche Kommando in Form eines Strings wird aber nur an ein Format-Plugin weitergeleitet, welches die eigentliche Operation ausführt. Dieses Format-Plugin für ein bestimmtes Audio/Videoformat muss vom Anwendungsprogrammierer erstellt werden und ist nicht direkt in der Oracle interMedia Option enthalten. Die durch den Objekttyp *ORDImage* repräsentierten Bilder besitzen die Methoden `process` oder `processCopy`, durch die mit Hilfe eines Kommandostrings Formatkonvertierungen und andere Transformationen möglich sind. Dies ist durch eine Reihe von Schlüsselwortern möglich, die hier kurz aufgezählt wird, um einen Überblick über die Funktionalität zu gewinnen:

- fileFormat
- contentFormat
- compressionFormat
- compressionQuality
- cut
- scale, xscale, yscale, fixedScale, maxScale

Weiter unten gibt es ein Beispiel für die genaue Syntax, bei dem ein Passbilder der Tabelle Person auf die Größe 500x500 skaliert wird.

8.3 Inhaltsbasierte Suche

Für die Objekttypen *ORDAudio*, *ORDVideo* und *ORDDoc* existiert in der interMedia Option keine Möglichkeit der inhaltsbasierten Suche. Die inhaltsbasierte Bildsuche funktioniert allerdings ähnlich wie bei dem Informix Excalibur Image DataBlade. Hierzu muss aus jedem Bild, dargestellt durch *ORDImage*, ein sogenannter Featurevektor extrahiert werden, der in dem Objekttyp *ORDImageSignature* gespeichert wird. In diesem Featurevektor sind Informationen über die Farbzusammensetzung (Histogramm, positional Color), Textur und Form eines Bildes enthalten. Alle diese Eigenschaften können bei einer Abfrage gewichtet miteinander kombiniert werden und ergeben somit eine Bewertungszahl, die zwischen 0 und 100 liegt (0=keine Überdeckung, 100=perfekte Überdeckung). Durch die Angabe eines Schwellenwertes im Bereich von 0 bis 100 können bei Anfragen weniger gut passende Bilder ausgeschlossen werden. Eine Beschleunigung von Abfragen kann erreicht werden, indem auf einem Attribut von Typ *ORDImageSignature* ein Index vom Typ ORDIMAGEINDEX angelegt wird. In der eigentlichen Abfrage kann dann die Methode `ORDImageSignature.IMGSimilar` verwendet werden, die auf Basis eines Kommandostrings zwei Bilder vergleicht und im Erfolgsfall "1" zurückgibt. Die Methode `ORDImageSignature.IMGScore` gibt die bei dem Vergleich ermittelte Bewertungszahl zurück. Da `IMGScore` in der SELECT-Klausel und `IMGSimilar` in der WHERE-Klausel verwendet werden, müssen die beiden Methoden über eine Zahl miteinander verbunden werden. Zum generellen Arbeiten mit der Oracle 9i interMedia Option hier ein Beispiel:

```
-- Anlage der Tabelle
CREATE TABLE Person (id NUMBER,
                     passbild ORDImage,
                     featureVektor ORDImageSignature);

-- Index auf dem Featurevektor anlegen
CREATE INDEX idx1 ON Person(featureVektor) INDEXTYPE IS ORDSYS.ORDIMAGEINDEX;

-- Die Tabelle Person mit Werten füllen
INSERT INTO Person (id, passbild, featureVektor)
VALUES (1, ORDSYS.ORDImage ('file', 'MYDIR', 'frodo.jpg', ORDSYS.ORDImageSignature.init());
INSERT INTO Person (id, passbild, featureVektor)
VALUES (2, ORDSYS.ORDImage ('file', 'MYDIR', 'gandalf.jpg', ORDSYS.ORDImageSignature.init());

-- Beispiel für eine Bildmanipulation in pl/SQL
DECLARE
  t_image ORDSYS.ORDImage;
  image_sig ORDSYS.ORDImageSignature;
BEGIN
  -- Bild und Featurevektor aus DB laden
  SELECT passbild, featureVektor INTO t_image, image_sig FROM Person
  WHERE id=1 FOR UPDATE;

  -- Bild auf die Größe 500x500 skalieren
  t_image.process ('maxScale=(500,500)');

  -- Eine Signatur bzw. einen FeatureVektor erstellen
  image_sig.generateSignature(t_image);

  -- Veränderte Daten speichern
  UPDATE Person SET passbild = t_image, featureVektor = image_sig WHERE id =1;
END;

-- Beispiel für eine inhaltsbasierte Suche in pl/SQL
DECLARE
  img_score NUMBER;
  i INTEGER;
  query_signature ORDSYS.ORDImageSignature;
  image_sig ORDSYS.ORDImageSignature;
  t_img ORDSYS.ORDImage;
BEGIN
  SELECT featureVektor INTO query_signature FROM Person WHERE id=42;

  SELECT id, ORDSYS.IMGScore(4711), passbild, photo_sig
    INTO i, img_score, t_img, image_sig   FROM Person
  WHERE ORDSYS.IMGSimilar(image_sig, query_signature,
     'color="0.2" texture="0.1" shape="0.5" location="0.2"',
     50,       -- Schwellwert von 50 muss überschritten werden
     4711      -- Benötigt für  Verbindung  zu IMGScore
     ) = 1
END;
```

9 IBM Informix Video Foundation DataBlade

Das Video Foundation DataBlade[INF98a] enthält, wie der Name schon sagt, grundlegende Funktionalität, um eigene Videoapplikationen oder Videodatablades zu entwickeln.

Die beiden wesentlichen Komponenten dieses Datablades sind die Temporal-Komponente, die eine Abstraktion von Zeitkodes durchführt und die Speicherungskomponente, die von den einzelnen Speicherungsmöglichkeiten und Formaten abstrahiert.

9.1 Temporal-Komponente

Verschiedene Videoformate benutzen unterschiedliche Zeitcodes, um eine bestimmte Stelle in einem Video anzusprechen. Es ist z.B. möglich, über HH:MM:SS (Stunde:Minute:Sekunde), SMPTE (Industrie-Standardformat) oder Frames einen bestimmten Zeitpunkt in einem Video genau zu bestimmen. Zu diesem Zweck wird der unstrukturierte Datentyp *MedTc* eingeführt, der intern mit dem Videostandard "drop-frame NTSC" arbeitet. Eine Umwandlung von einem in das andere Format ist nicht immer trivial, wie das folgende Beispiel (drop-Frame NTSC) zeigt: Steht der Zeitzähler auf 01:02:59:29, so folgt als nächstes Frame 01:03:00:02 und nicht 01:03:00:00. Nur wenn die nächste Minute ein gerades Vielfaches von 10 ist, so folgt 00 (01:09:59:29->01:10:00:00).

Diese kleinen Besonderheiten werden von dem Datentyp *MedTc* berücksichtigt, wobei auch Funktionen existieren, die festzustellen, ob ein Zeitpunkt vor einem anderen Zeitpunkt liegt. Zeitpunkte können miteinander addiert und subtrahiert werden.

Der unstrukturierte Datentyp *MedChunk* ermöglicht es, einen bestimmten Zeitbereich in einem Video genau festzulegen. Für diesen Typ existieren zusätzliche Funktionen, um beispielsweise zu bestimmen, ob sich zwei Zeitintervalle überlappen/aneinanderliegen oder welches Zeitintervall vor dem anderen liegt.

9.2 Speicherungs-Komponente

Die Speicherungs-Komponente hat zwei Aufgaben:

1. Die Einbindung externer Datenquellen wie z.B. physikalische Geräte (Videorekorder), Streaming Server (Realaudio) oder einfacher externer Dateien auf anderen Dateisystemen.

2. Abstraktion von unterschiedlichen Videoformaten

Die Zugriffsinformationen auf unterschiedliche Datenquellen werden in die Tabelle MedVSIRegister eingetragen. Dies beinhaltet Quellentyp, Zugriffsinformation (Benutzername, Passwort), Zugriffspfad, Herstellername und Version der Quelle. Videodaten innerhalb einer solchen Quelle werden mit dem unstrukturierten Datentyp *MedLoc* referenziert.

9.3 Das Videoschema

Um nun die beiden Konzepte der Speicherungs-Komponente und der Temporal-Komponente zu realisieren, wird mit dem Video Foundation Datablade ein umfangreiches Schema zur Videoverwaltung zur Verfügung gestellt. In diesem Schema existieren beispielsweise Entitäten für Codecs, Videos oder Audiodaten, welche von der Anwendung zur Ablage von Informationen verwendet werden können. Eine automatisierte Datenverwaltung wie etwa bei Bildern gibt es hier nicht, d.h., wenn ein neues Video in die Videotabelle eingefügt wird, so müssen vom Anwendungsprogrammierer alle Attribute des Videos angegeben werden, etwa Format, Framerate, Kodierung. Eine automatische Extraktion der Attribute oder inhaltsbasierte Suche auf Videodaten ist mit dem Video Foundation DataBlade nicht möglich.

9.4 Beispiel für die Integration eines externen Servers

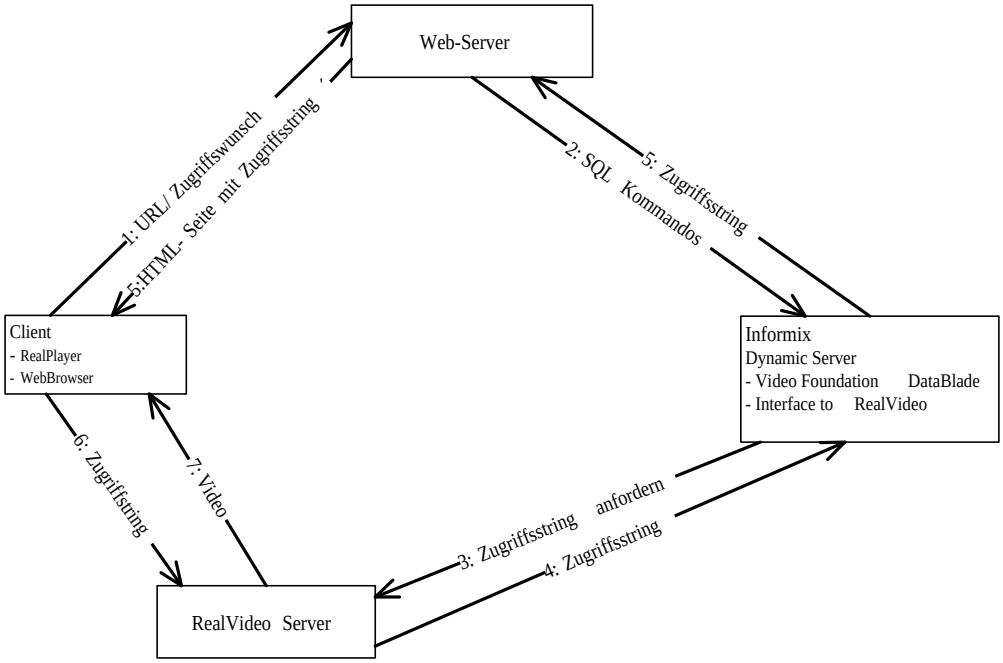

Abbildung 3: Zugriff auf ein Video, das auf einem externen Server gespeichert ist

Neben dem eigentlichen Video Foundation Datablade wurden von Informix auch einige Interfaces zu externen Servern entwickelt [INF98b], die auf dem Video Foundation Datablade aufbauen. Es existieren Interfaces zu Microsoft NetShow (eingestellt 1999), dem Hersteller EMC (nicht mehr kompatibel) und zu RealNetworks RealVideo.

Um eine Vorstellung davon zu bekommen, wie ein Client auf Videodaten des Datenbankservers (bzw. Referenzen auf Videodaten) zugreift, sei dies hier einmal am Beispiel von RealVideo demonstriert (Abbildung 3).

10 Zusammenfassung und Ausblick

Zusammenfassend lässt sich sagen, dass alle großen Hersteller von ORDBMS über Produkte verfügen, mit denen Audio-, Video- und Bilddaten in eine intelligente Art und Weise verwaltet werden können. Diese Produkte sind als Erweiterungsmodule realisiert und damit weitgehend in das entsprechende Datenbanksystem integriert.

Im Allgemeinen fällt auf, dass Audiodaten noch recht schlecht unterstützt werden. So können Audiodaten zwar in die DBS eingebracht und es kann auf einfache Attribute dieser Daten zugegriffen werden, aber komplexe Manipulationsfunktionen oder eine inhaltsbasierte Suche sind bei keinem der Hersteller möglich.

Videodaten werden von allen Systemen unterstützt, wobei auch hier keine Manipulationsfunktionen oder inhaltsbasierte Suchmöglichkeiten existieren. Mit Oracle und Informix ist es möglich, Video und Audiodaten von externen Multimediaservern einzubinden und an den Client weiterzuleiten, wobei das eigentliche Streaming oft von einem spezialisierten Multimediaserver durchgeführt wird. Der DB2 Video Extender verfügt über die Möglichkeit, eine Storyboard zu einem Video zu erstellen.

Die beste Unterstützung bieten die Hersteller sicher für Bilddaten, da diese in den täglichen Geschäftsprozessen auch am häufigsten auftauchen. Jeder Hersteller bietet die Möglichkeit der inhalts-

basieren Suche und der Bildmanipulation. Leider unterscheidet sich auch die Syntax der unterschiedlichen Erweiterungsmodule erheblich. In dem DB2 Image Extender wird für die Bildmanipulation eine Art Kommandostring verwendet, der Ähnlichkeit mit den Parametern hat, die bei einem Programmaufruf verwendet werden. Oracle interMedia verwendet ebenfalls eine einzige Funktion, die einen Kommandostring erhält, welcher natürlich nicht mit DB2 kompatibel ist. Das Informix Excalibur Image Datablade geht hier wieder einen anderen Weg und realisiert die Manipulationsfunktionen als orthogonal zueinander benutzbare Funktionen. Der in dem Informix Excalibur Datablade genommene Weg ist im Hinblick auf den kommenden SQL/MM Standard und auf die Bedienung hin der günstigere. Dabei muss aber bedacht werden, dass die Optimerung eines einzelnen Kommandostrings immer einfacher sein wird als die Optimierung orthogonaler Funktionen, deren Aufrufhierarchie ineinander verschachtelt sein kann.

Bei der inhaltsbasierten Suche auf Bildern zeichnet sich in Bezug auf die Syntax ein ähnliches Bild ab. Jeder Hersteller besitzt eine eigene kleine Abfragesprache. Erschwerend kommt hinzu, dass die Bedeutung des Schwellenwertes oder der Bewertungszahl nicht genau festgelegt ist. So ist nicht klar, welche Bilder bei einem Schwellenwert von 45, im Vergleich zu einem Schwellenwert von 28, zurückgeliefert werden. Auch ist ein solcher Schwellenwert von der eigentlichen Abfrage anhängig, d.h. hat bei einer Abfrage ein Schwellenwert von 20 zu guten Ergebnissen geführt, so kann bei einer anderen Abfrage ein Schwellenwert von 75 optimal sein. In [ORA01b] steht dazu: “Through trial and error, adjust the threshold to an appropriate value for your application.”

Inzwischen gibt es verschiedene Produkte, die auf der Basis der vorgestellten Erweiterungsmodule erweiterte Funktionalität anbieten. Der Hersteller Viisage Inc. bietet beispielsweise für Oracle und Informix ein Erweiterunsmodul für die Gesichtserkennung an. Auch Erweiterungsmodule im Medizinbereich oder für die Handschriftenerkennung sind in Zukunft denkbar.

Eine Lösung, die recht unterschiedlichen Syntaxgebilde für inhaltsbasierte Bildsuche und Bildmanipulation der verschiedenen Hersteller vereint, könnte der SQL/MM Standard sein. Die geforderte Funktionalität wird im Wesentlichen von allen Produkten unterstützt, wenn man von weniger sinnvollen Funktionen wie der Durchschnittsfarbe einmal absieht.

Jedes Produkt verfügt über die Möglichkeit, externe Daten in das Datenbanksystem einzubinden. Leider ist auch hier eine große Vielfalt an verschiedenen Ansätzen zu finden, welche sich sowohl von der Syntax, als auch durch Inhalt unterscheiden. Dies kann man auch innerhalb einer Datenbank vorfinden. So unterstützt IBM DB2 das DataLink-Konzept, eine Nutzung für externe Audio-, Video- oder Bilddaten ist jedoch nicht möglich. Hier könnte der kommende SQL/MED Standard eine gewisse Vereinheitlichung bewirken, wobei beachtet werden muss, dass Multimediadaten auf externen Spezialservern ein komplexeres Zugriffsmuster erfordern können.

Literatur

[STO02] Stolze, Knut: “Still Image Extensions in Database Systems - A Product Overview”, in Datenbank Spektrum 02/2002

[ORA02] Oracle Corporation: “Oracle 9i Data Catridge Developers Guide Release 1”, June 2002

[ORA01a] Oracle Corporation: “Oracle interMedia: Managing Multimedia Content” Whitepaper, March 2001

[ORA01b] Oracle Corporation: “Oracle interMedia User’s Guide and Reference” Release 9.0.1, June 2001

[ORA01c] Oracle Corporation: “Oracle9i Application Developer’s Guide - Large Objects (LOBs)” Release 9.0.1, June 2001

[INF99a] Informix Corporation: “Excalibur Image Datablade Module: User’s Guide” Version 1.2, March 1999

[INF00] Informix Corporation: “Informix Image Foundation Datablade Module: User’s Guide” Version 2.0, December 2000

[INF99b] Informix Corporation: “Informix Large Object Locator Datablade Module: User’s Guide” Version 1.2, July 1999

[INF98a] Informix Corporation: “Informix Video Foundation Datablade Module: User’s Guide” Version 1.2, September 1998

[INF98b] Informix Corporation: “Informix Video Foundation Interfaces Supplement” Version 1.2, November 1998

[IBM00a] IBM Corporation: “IBM DB2 Universal Database: Image, Audio und Video Extender Verwaltung und Programmierung”, April 2000

[IBM00b] IBM Corporation: “IBM DB2 Universal Database Application Developers Guide”, 2000

[TÜR01] Türker, C.: “Vorlesung Objektrelationale, erweiterbare Datenbanken - Kapitel 8 Erweiterbarkeit”, WS01/02, ETH Zürich