Proc. Net.ObjectDays 2003, 22.-25. Okt. 2003, Erfurt, pp. 314-321

Struktur-, Verhaltens- und Ausgabeschemata für Multimedia-Daten in objektrelationalen Datenbanksystemen

Thomas Heimrich[1], Günther Specht[2]

[1] TU-Ilmenau, FG Datenbanken und Informationssysteme, 98684 Ilmenau
[2] Universität Ulm, Abteilung Datenbanken und Informationssysteme, 89069 Ulm

Abstract. Objektrelationale Datenbanken bieten sich heute für die Speicherung von unabhängigen Multimedia-Daten an. Die formalen Konzepte objektrelationaler Datenbanken berücksichtigen aber bisher keine spezielle Beziehungen (Substitutionsbeziehung, Synchronisationsbeziehung) zwischen Multimedia-Objekten. Wir stellen daher eine formale Erweiterung der Struktur- und Verhaltensvererbung vor, die auch die Substitutionsbeziehung berücksichtigt. Weiterhin wird ein Ausgabeschema als neues Konzept eingeführt, um die Synchronisationsbeziehungen für die Datenausgabe zu modellieren.

1 Einleitung

Multimedia-Datenbanksysteme sind meist keine völlig neu entwickelten Systeme. Man nutzt in der Regel bestehende relationale oder objektorientierte Datenbanksysteme, um Multimedia-Daten zu speichern und baut darüber eine entsprechende Multimediaschicht.
Relationale Datenbanken können Multimedia-Daten nur sehr rudimentär unterstützen. Dies liegt an deren sehr eingeschränkten und nicht erweiterbaren Typsystem. Um Multimedia-Daten zu speichern stehen nur die Datentypen Binary (bzw. Character) Large Objects (BLOB und CLOB) zur Verfügung. Sowohl Video, Audio und auch Bilder müssen als BLOB gespeichert werden. Die Information, welchen Medien-Typ ein BLOB speichert ist anhand des BLOB's nicht zu ermitteln. Relationale Datenbanken unterstützen weiterhin keine komplexen Objekte, die für zusammengesetzte Medien benötigt würden. Darüberhinaus gibt es keine Möglichkeit spezielle Präsentationsmethoden mit abzulegen. Vorteile der relationalen Datenbanken sind u.a. die deskriptive Anfragesprache (SQL) und die effiziente Verwaltung von Tabellen.
Objektorientierte Datenbanken dagegen besitzen ein erweiterbares Typsystem. Sie ermöglichen die Erzeugung spezieller Medien-Typen (z.B. Image, Video) und die Realisierung des Verhaltens in Form von Methoden. Medien-Typen können in einer Typhierarchie unter Nutzung von Vererbung modelliert werden. Selbstdefinierte Typen haben allerdings den Nachteil, dass sie für die Anfrageoptimierung und Indizierung nicht direkt genutzt werden können. Nachteilig ist auch, dass objektorientierte Datenbanken oft keine standardisierte ad-hoc Anfragesprache unterstützen, sowie die schlechte Skalierbarkeit der Systeme.

Objektrelationale Datenbanken versuchen, die Vorteile der beiden beschriebenen Ansätze zu vereinen. Dieses Papier untersucht, in wieweit objektrelationale Konzepte genutzt werden können, um Multimedia-Daten zu verwalten. Insbesondere wird ein Ausgabeschema neu eingeführt. Dieses soll Beziehungen während der Datenausgabe modellieren.
Abschnitt 2 beschreibt spezielle Anforderungen, die bei der Speicherung von Multimedia-Daten zu beachten sind. Abschnitt 3 gibt einen Überblick über die benötigte Funktionalität, die sich aus diesen Anforderungen ergibt. Dabei werden objektrelationale Konzepte vorgestellt und erweitert, die es ermöglichen, Struktur und Verhalten in einer objektrelationalen Datenbank zu modellieren. Abschnitt 4 fasst die Ergebnisse dieses Artikels zusammen.

2 Anforderungen an Multimedia-Datenbanksysteme

Bereits 1985 benannte Christodoulakis [ca85] die Verwaltung unformatierter Daten und die Verwaltung neuartiger Geräte zur Darstellung und Speicherung der unformatierten Daten als Aufgaben für ein Multimedia-Datenbanksystem.
In verschiedenen Arbeiten [mw03,sp98] findet man eine detaillierte Beschreibung aller Anforderungen an ein Multimedia-Datenbanksystem. Im Folgenden werden nur die wichtigsten davon genannt.

2.1 Speichern und Wiedergewinnen von Medienobjekten

Medienobjekte unterschiedlicher Medien (Audio, Video, Animationen, Untertitel, Volltexte usw.) müssen effizient als unterschiedliche Medientypen gespeichert werden können. Nur so ist die Überwachung struktureller und semantischer Integritätsbedingungen bereits innerhalb der Datenbank möglich.
Für die Wiedergewinnung von Medienobjekten wird heute insbesondere auch eine inhaltsorientierte Suche auf Medienobjekten gefordert. Dies erfordert eine automatische Analyse und Strukturierung der Medienobjekte bereits bei der Eingabe und die effiziente Ablage entsprechender Metadaten (Histogramme etc.) einschließlich oft mehrdimensionaler Indexstrukturen darauf. Die Art und Struktur von Metadaten wird z.B. durch den Standard MPEG-7 beschrieben [ms02].
Bei der Ausgabe von Medienobjekten müssen zeitliche und räumliche Beziehungen zwischen den Medienobjekten berücksichtigt werden. Insbesondere müssen multimediale Datenbanken daher Synchronisationsbedingungen zur Datenausgabezeit behandeln können.

2.2 Beziehungen zwischen Daten verschiedener Medien

Neben den herkömmlichen Beziehungen zwischen Datenbankentitäten (1:1, 1:n, n:m-Beziehung), unterscheidet man spezielle Beziehungen unter Beteiligung von Medienobjekten:

- Attributbeziehung: Die Daten einer Entität können oft durch Multimedia-Daten ergänzt werden. So kann z.B. in einer Personenrelation auch ein Bild zu jeder Person als Attribut gespeichert werden. Wichtig ist, dass das Bild die Person lediglich beschreibt, aber nicht Komponente (siehe unten) der Person ist.
- Komponentenbeziehung: Oft wird aus verschiedenen Medienobjekten ein zusammengesetztes Medienobjekt gebildet. Aus einem Video und einem Audio kann ein zusammengesetztes Medienobjekt (Film) erzeugt werden. Die beteiligten Medienobjekte beschreiben dann nicht eine Entität, sie sind vielmehr Bestandteil dieser Entität.
- Substitutionsbeziehung: Die gleiche Information kann in verschiedenen Medien dargestellt werden (z.B. Sprache und Text). Der Ersteller eines Medienobjektes kann eine bestimmte Darstellung favorisieren (z.B. Sprache), aber als Alternative auch eine andere Darstellung erlauben (z.B. Text).
- Synchronisationsbeziehung zwischen Medienobjekten: Oft sollen Medienobjekte in einer bestimmten Reihenfolge bzw. an einer bestimmten Position ausgegeben werden. Synchronisationsbeziehungen definieren den zeitlich/räumlichen Ablauf der Datenausgabe.
- Semantische Äquivalenzbeziehung: Oft werden Multimedia-Daten in verschiedenen Formaten oder mit unterschiedlichen Qualitätsparametern, z.B. in verschiedenen Auflösungen, gespeichert. Da es sich nur um verschiedene Varianten des selben Medienobjektes handelt, sind diese Daten semantisch äquivalent.

3 Formale Umsetzung von Struktur, Verhalten und Ausgabe

Grundlage jeder objektorientierten oder objektrelationalen Datenbank ist das Schema. Dieses hat entscheidenden Einfuss auf die Semantik. Das Schema kann in Struktur-, Verhaltens- und Ausgabeschema aufgeteilt werden. Das Strukturschema enthält nur strukturelle und Typinformationen. Das Verhalten, in Form von Methoden, wird durch das Verhaltensschema beschrieben. Das Ausgabeschema definiert die Synchronisation während der Datenausgabe. Alle drei Modelle sollen im Folgenden formal betrachtet werden. Anschließend wird beschrieben, wie ein Objektmodell für Multimedia-Daten aussehen muss.

3.1 Strukturschema

Um komplexe Strukturen (Objekte) beschreiben zu können, benötigt man ein entsprechendes *Typsystem (T)*. Ein mögliches Typsystem für eine Multimedia-Datenbank wäre:

(i) integer, string, float, boolean, image, video, audio, fulltext $\subseteq T$ (Basistypen. Die Typen image, video, audio und fulltext werden weiterhin auch als Multimedia-Datentypen bezeichnet.)

(ii) sind A_i verschiedene Attribute und $t_i \in T, 1 \le i \le n$, so ist $[A_1 : t_1,..., A_n : t_n] \in T$ (Tupel-Typ)

(iii) ist $t \in T$, so ist $\{t\} \in T$ (Mengen-Typ)

(iv) ist $t \in T$, so ist $\langle t \rangle \in T$ (Listen-Typ)

(v) $UDT \subseteq T$, wobei UDT eine Menge von Namen für nutzerdefinierte Typen (user defined types) ist

Multimedia-Datentypen müssen Basistypen des Typsystems sein (i). Nur so ist es möglich, dass die Datenbank effiziente Zugriffsmechanismen und Integritätsprüfungen für diese Typen anbietet.
Die in (v) definierten UDTs sind nutzerdefinierte Typen (meist Sammeltypen). Mit ihnen kann man Informationen von verschiedenen Typen heraus ansprechen. Die Domäne der UDTs sind dabei die Objekt-Identifikatoren (OID).
Die in Abschnitt 2.2 geforderte Attributbeziehung ist einfach zu realisieren, indem ein Attribut eines komplexen Typs einen Multimedia-Datentyp annimmt. Die Komponentenbeziehung kann auch umgesetzt werden, indem Typen zu Bestandteilen eines komplexen Typs werden. Die Unterscheidung in Attribut- und Komponentenbeziehung bezieht sich auf die Semantik, nicht auf die Implementierung.

3.2 Strukturvererbung

Von allen Typen können durch Ableitung Subtypen gebildet werden. Ein Typ *t'* ist Subtyp eines Typs *t*, wenn jede Instanz von *t'* auch eine Instanz von *t* ist. Die Subtyp-Relation „$\le$“, $\le \subseteq T \times T$ ist wie folgt definiert:

(i) $t \le t$ für jedes $t \in T$

(ii) $[A_1' : t_1',..., A_m' : t_m'] \le [A_1 : t_1,..., A_n : t_n]$ falls

(a) $m \ge n$ und

(b) $(\forall A_i, 1 \le i \le n)(\exists A_j', 1 \le j \le m) A_i = A_j' \wedge t_j' \le t_i$

(iii) $\{t'\} \le \{t\}$ falls $t' \le t$

(iv) $\langle t' \rangle \le \langle t \rangle$ falls $t' \le t$

Die gegebene Definition geht davon aus, dass man auf Typen eine feste Ordnung ($\le$) definieren kann (Bsp.: integer $\le$ float). Multimedia-Daten verfügen allerdings über eine Substitutionsbeziehung (Abschnitt 2.2), die nicht fest definiert werden kann. Hat man z.B. eine Attribut *Geschäftszahlen* als *fullText* deklariert, so kann es wünschenswert sein, in einem Subtyp dieses Attribut als *image* zu definieren, wenn man davon ausgeht, dass die gleiche Information einmal als Text und einmal als Bild darstellbar ist. Nach obiger Definition könnte man den gewünschten Subtyp nicht erzeugen. Die Subtyp-Relation berücksichtigt keine Substitutionsbeziehung. Die Subtyp-Relation müsste wie folgt ergänzt werden:

(i) $t \le t$ für jedes $t \in T$

(ii) $\left[A_1' : t_1', ..., A_m' : t_m'\right] \leq [A_1 : t_1, ..., A_n : t_n]$ falls

(a) $m \geq n$ und

(b) $(\forall A_i, 1 \leq i \leq n)(\exists A_j', 1 \leq j \leq m) A_i = A_j' \wedge$

$(t_j' \leq t_i \vee t_i$ *substituierbar durch* $t_j')$

(iii) $\{t'\} \leq \{t\}$ falls $t' \leq t \vee$ t substituierbar durch t'

(iv) $\langle t' \rangle \leq \langle t \rangle$ falls $t' \leq t \vee$ t substituierbar durch t'

3.3 Umsetzung der Struktur in Tabellen

Definiert man Tabellen über Typen, so spricht man von typisierten Tabellen. Untypisierte Tabellen entsprechen den klassischen Tabellen des relationalen Konzepts.
Die Tabellen zu einer Typhierarchie stehen untereinander in einer IS-A-Beziehung. Die Tabellenhierarchie bildet die Typhierarchie exakt nach. Bei der Tabellendefinition muss angegeben werden, ob eine Tabelle eine Subtabelle einer bereits existierenden ist. Ein Subtabelle erbt alle Eigenschaften der Obertabelle und kann diese Eigenschaften erweitern.
Für jedes Attribut eines komplexen Typs, das nicht von einen Basisdatentyp ist, wird in der Regel eine eigene Tabelle angelegt. In der Tabelle des komplexen Typs werden anstelle des strukturierten Attributes dann lediglich deren OID (Referenzen) gehalten. OID können auf verschiedene Arten implementiert werden. Einige Systeme bilden sie auf Primärschlüssel ab, müssen dann aber bei OID-Referenzen über Fremdschlüssel joinen, andere auf Pointer auf physische Speicheradresse, dann wird aber Pointer-Swizzeling notwendig. Wieder andere Systeme bilden OIDs auf Surrogate ab. Sie werden automatisch erzeugt und z.B. durch einen Index (z.B. Hash-Tabelle) auf physische Adressen abgebildet. Schließlich gibt es als vierte und letzte Möglichkeit noch getypte Surrogate, um über den Typ schneller auf die Klassenzugehörigkeit schließen zu können.

3.4 Verhaltensschema

Das Verhalten von Typen wird durch Methoden bestimmt. M ist eine endliche Menge von Methoden-Namen. Zu jeder Methode $m \in M$ gehört eine nichtleere Menge von Signaturen $sig(m) = \{s_1, ..., s_l\}, l \geq 1$. Jedes $s_i, 1 \leq i \leq l$ hat die Form:

$s_i : udt \times t_1 \times ... \times t_p \rightarrow t$, mit $udt \in UDT, t_1, ..., t_p, t \in T$

Die Methode wird dem Typ zugeordnet, der in der Methodensignatur deklariert ist (Für Basistypen existieren inhärente Methoden.). Auf diese Weise können jedem UDT beliebige Methoden zugeordnet werden. Die eigentliche Implementierung der Methode kann in einer Programmiersprache (z.B. C++, Java, VisualBasic) erfolgen oder direkt in SQL gegeben sein. Die folgende in SQL:99 verwendete Funktionsdeklaration ist eine direkte Umsetzung obiger formaler Signaturdefinition:

CREATE FUNCTION funktionsname (typ_1, …, typ_p)
RETURNS typ_output
AS
Dateiname oder SQL-Ausdruck;

3.5 Verhaltensvererbung

Auch das Verhalten der Typen muss von Supertyp auf Subtyp vererbbar sein. Bei der Definition einer Methode wird einem Methodennamen eine Signatur zugeordnet. Einem Methodennamen können je Typ unterschiedliche Signaturen zugeordnet werden. Auf diese Weise wird auch das Überladen von Methoden realisiert. Das kann dazu führen, dass sich die Signatur einer Methode des Supertyps völlig von der Signatur dieser Methode im Subtyp unterscheidet. Bei einem Methodenaufruf könnte der Subtyp dann nicht durch den Supertyp ersetzt werden.
In der Regel will man auch bei der Vererbung von Verhalten eine IS-A-Hierarchie der Typen verarbeiten können. Dazu ist es nötig, dass Methoden im Subtyp lediglich spezialisiert werden dürfen, d.h. ein Überladen nur mit kompatiblen Signaturen erfolgen darf. Formal würde dies wie folgt definiert (Kontravarianz):

Ist udt' IS-A udt und $s, s' \in sig(m)$ für $m \in M$ mit
$s: udt \times t_1 \times ... \times t_n \to t$, s': $udt' \times t_1' \times ... \times t_n' \to t'$, so gilt $t_i' \leq t_i$ für jedes $i, 1 \leq i \leq n$, und $t \leq t'$.
Für Multimedia-Daten muss man wieder fordern, dass die Kompatibilität zwischen Typen um die Substituierbarkeit erweitert wird.

3.6 Ausgabeschema

Die Ausgabe ist bei Multimedia-Daten erheblich komplexer als bei herkömmlichen alphanumerischen Daten. Es bestehen Synchronisationsbeziehungen (Abschnitt 2.2), die bei der Ausgabe berücksichtigt werden müssen. Während der Ausgabe muss auch eine bestimmte Datenqualität garantiert werden. Ansonsten können Multimedia-Daten semantisch verfälscht werden.
Bisher werden Multimedia-Datenbanksysteme oft zur Speicherung unabhängiger Multimedia-Daten benutzt. Es werden zwar Daten, aber wenig Beziehungen zwischen diesen Daten gespeichert. Man geht davon aus, dass der Nutzer bei der Anfrage die Multimedia-Daten zueinander in Beziehung setzt. Verbal könnte eine solche Anfrage wie folgt aussehen: „Video1 und Audio2 sollen gesucht und parallel ausgegeben werden." Entsprechende SQL-Erweiterungen sind leicht vorstellbar. Problematisch sind dabei folgende Punkte:

- Der Nutzer muss in der Datenbankanfrage eine komplexe Ausgabe beschreiben.
- Die Spezifikation der Ausgabe ist nur einmal verwendbar. Soll die selbe Ausgabe noch einmal erfolgen, muss die Spezifikation neu erstellt werden.

- Der Nutzer muss die zeitliche Semantik der Ausgabe bestimmen.
- In der Datenbank werden keine Synchronisationsbeziehungen gespeichert. Dadurch geht Semantik der Multimedia-Daten verloren.

Um Synchronisationsbeziehungen zwischen Multimedia-Daten speichern zu können, führen wir *Ausgabetypen* ein. Sie bilden eine Art Schablone für ein Datenausgabe. In dieser Arbeit werden Ausgabetypen nur für zeitliche Synchronisationsbeziehungen definiert. Man kann in einen Ausgabetyp aber auch Bedingungen zur Datenqualität mit aufnehmen. Der Ausgabetyp wird vom Nutzer einmal erstellt und von der Datenbank verwaltet. Bei Anfragen kann der Nutzer nun einfach den gewünschten Ausgabetyp, anstelle einer kompletten Spezifikation, mit angeben. Der Ausgabetyp ist wiederverwendbar und kann für Optimierungen bei der Datenausgabe benutzt werden. Im folgenden wird eine formale Definition eines *Ausgabetyp-Systems (AT)* gegeben:

(i) video, audio, imgae, fulltext $\subseteq AT$

(ii) Es seien at1, at2 $\in AT$, so sind auch at1 vor at2, at1 trifft at2, at1 überlappt at2, at1 während at2, at1 startet mit at2, at1 endet mit at2, at1 gleichzeitig at2 $\in AT$

Die in (ii) verwendeten zeitlichen Relationen sind die Allen-Relationen [al83]. Es können auch andere Konzepte zeitlicher Relationen verwendet werden. Wichtig ist, dass durch Kombination beliebig komplexe Ausgabetypen erzeugt werden können.
Es sollte möglich sein, Ausgabetypen in einer Typhierarchie (IS-A-Beziehung) anzuordnen. Man könnte Ausgabetypen so spezialisieren. Es könnte z.B. ein spezieller Ausgabetyp erzeugt werden, der at1 2 Sekunden vor at2 ausgibt. Durch Ableitung könnte auch die Substitutionsbeziehnung zwischen Multimedia-Daten unterstützt werden. Der Ausgabetyp „video gleichzeitig audio“ könnte spezialisiert werden zu „video gleichzeitig fulltext“. Dieser Subtyp erlaubt die parallele Ausgabe von Video und Audio und alternativ die parallele Ausgabe von Video und Fulltext.

4 Zusammenfassung

Objektrelationale Konzepte sind geeignet, um Multimedia-Daten zu verwalten. Besonders die direkte Unterstützung von Multimedia-Datentypen bringt Vorteile. Über Struktur- und Verhaltensvererbung wird in der Regel eine IS-A-Hierarchie erzeugt. Ebenso kann aber auch eine Generalisierungshierarchie verwendet werden. Bei der Spezialisierung von Typen und Methoden ist die Substituierbarkeit von Multimedia-Datentypen bisher nicht berücksichtigt. Wir haben die Subtyp-Relation so erweitert, dass ein Typ *t'* Subtyp eines Typs *t* sein kann, wenn *t* durch *t'* substituierbar ist.
Es wurde gezeigt, dass eine Modellierung der Datenausgabe für Multimedia-Daten nötig ist. Dazu wurde das Konzept der Ausgabetypen neu entwickelt. Durch Ausgabetypen können Synchronisationsbeziehungen zwischen Multimediaobjekten innerhalb der Datenbank modelliert werden. In objektrelationale Datenbanken kann dieses Konzept leicht integriert werden.

Literatur

[al83] Allen, J.F.: *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM, 26(11):832-843, 1983

[bk99] S. Boll, W. Klas, U. Westermann: *Exploiting ORDBMS Technology to Implement the ZyX Data Model for Multimedia Documents and Presentations.* BTW GI-Fachtagung, Freiburg, 1.-3. März 1999

[ca85] Christodoulakis S., Analyti A.: *Guest Editorial Special Issue on Multimedia Information Systems.* IS 20(6): 443-444 (1995)

[ms02] Herausgeber: Manjunath, B.S., P. Salembier, T. Sikora: *Introduction to MPEG-7 — Multimedia Content Desctiption Interface.* Wiley, 2002

[mw03] Meyer-Wegener K.: *Multimedia-Datenbanken – Einsatz von Datenbanktechnik in Multimedia-Systemen*. B.G. Teubner Stuttgart, 2003

[sp98] Specht G.: *Habilitationsschrift: Multimedia-Datenbanksysteme: Modelle – Architektur – Retrievel.* Technische Universität München, 1998

[st99] Stonebraker M.: *Objektrelationale Datenbanken – Die nächste große Welle.* Hanser Verlag München Wien, 1999

[tu99] Türker C.: *Integritätsbedingungen und Spezialisierung in Objektdatenbanken.* BTW GI-Fachtagung, Freiburg, 1.-3. März 1999, Seiten: 369-378