

# Fachhochschule Gelsenkirchen

*Fachbereich Informatik*

*Studiengang Angewandte Informatik*

## Diplomarbeit

**Thema:** Entwurf und Implementierung einer Datenbank zur Aufnahme von Binary  
Large Objects und ihre Anbindung ans Internet.

**vorgelegt von:** Markus Stamm

**betreut durch:** Prof. Dr. Klaus Drost (FH Gelsenkirchen) und  
Andreas Stark M.A. (Spinnrad GmbH)

---

Datum und Unterschrift des Betreuers

**Erklärung**

Hiermit versichere ich, die Arbeit selbständig angefertigt und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt zu haben.

Bottrop, 22. September 1998

\_\_\_\_\_

Markus Stamm

Diese Diplomarbeit ist ein Prüfungsdokument. Eine Verwendung zu einem anderen Zweck ist nur mit dem Einverständnis von Verfasser und Prüfern erlaubt.

## Inhaltsverzeichnis

<b>ERKLÄRUNG</b> .....	<b>2</b>
<b>INHALTSVERZEICHNIS</b> .....	<b>3</b>
<b>1 EINFÜHRUNG</b> .....	<b>6</b>
1.1 ARCHIVIERUNG MULTIMEDIALER DATEN.....	6
1.2 DIE SPINNRAD GMBH IN GELSENKIRCHEN .....	10
1.2.1 Das Unternehmen.....	10
1.2.2 Technische Voraussetzungen.....	10
<b>2 DATENBANKENTWURF</b> .....	<b>12</b>
2.1 KONZEPTIONELLES DATENMODELL ZUR AUFNAHME VON BLOBS .....	14
2.1.1 Domains .....	15
2.1.2 Binary Large Object (BLOB) .....	16
2.1.3 Inhaltstyp der Binärdaten .....	18
2.1.4 Der Entitätstyp <i>Bild</i> .....	19
2.1.5 Textdokumente und weitere Spezialisierungen von BLOBs.....	21
2.1.6 Das Produktfoto, eine besondere Art von Bild .....	22
2.1.7 Herkunft digitaler Daten: Digitalisiergerät.....	23
2.2 ZUSÄTZLICHE ENTITÄTEN FÜR DAS INFORMATION RETRIEVAL.....	25
2.2.1 Die Vorgehensweise beim Information Retrieval .....	25
2.2.2 Stichwort.....	26
2.2.3 Ersteller.....	27
2.2.4 Bildkategorie.....	28
2.3 GENERIERUNG DES PHYSIKALISCHEN DATENMODELLS.....	30
2.3.1 Einstellungen und Optionen des Power Designer Data Architect .....	30
2.3.2 Umwandlung von Entitätstypen .....	31
2.3.3 Umwandlung von Beziehungstypen.....	31
2.3.4 Probleme mit dem Datentyp Serial .....	32
2.4 ERSTELLEN DER DATENBANKTABELLEN.....	34
<b>3 ANBINDUNG DER DATENBANK AN DAS INTRANET</b> .....	<b>35</b>
3.1 AUFBAU DER SPRACHERWEITERUNG CFML.....	36
3.2 CFML-VARIABLEN .....	37
3.3 COLD FUSION TAGS .....	38
3.3.1 Datenbankabfrage mit <CFQUERY> .....	38
3.3.2 Ausgabe von Daten mit <CFOUTPUT>.....	40
3.3.3 Setzen von Variablen mit <CFSET>.....	40
3.3.4 Einfügen zusätzlicher Dateien in den Code mit <CFINCLUDE>.....	41
3.3.5 Umleitung der Ausgabe mit <CFLOCATION> .....	41
3.3.6 Schleifenprogrammierung mit <CFLOOP> .....	41
3.3.7 Bedingte Verzweigung mit <CFIF>, <CFELSEIF> und <CFELSE>.....	42
3.3.8 Dateioperationen mit <CFFILE> .....	42
3.4 FUNKTIONEN IN CFML.....	44
3.4.1 ParameterExists(Parameter).....	44
3.4.2 Chr(ASCII-Wert) .....	45
3.4.3 LCase(String).....	45
3.4.4 Left(String,n) .....	45
3.4.5 Len(String).....	45
3.4.6 PreserveSingleQuotes(String).....	45
3.4.7 ListContains(Liste,String,Trennzeichen) .....	45
3.4.8 ListContainsNoCase(Liste,String,Trennzeichen).....	46

3.5 SELBSTDEFINIERTER CFML-TAGS.....	46
3.5.1 Kopf und Fuß einer HTML-Seite festlegen mit <CF_HEAD> und <CF_FOOT> .....	47
3.5.2 Datenbankabfragen mit BLOBs (CFX_PUTIMAGE/CFX_GETIMAGE).....	49
3.5.3 Bildinformationen mit <CFX_IMGINFO> abfragen.....	50
<b>4 DATENBEARBEITUNGSFORMULARE UND ANWENDUNGSSEITEN.....</b>	<b>52</b>
4.1 DATENANZEIGEFORMULARE.....	55
4.2 ANZEIGE DER BINÄRDATEN.....	61
4.3 DATENEINFÜGEFORMULARE.....	63
4.4 DATENÄNDERUNGSFORMULARE.....	72
4.5 DATEN LÖSCHEN .....	75
4.6 SUCHFORMULAR.....	78
<b>5 SCHLUBBEMERKUNGEN.....</b>	<b>87</b>
<b>ANHANG.....</b>	<b>89</b>
<b>A DATENTYPEN IM POWER DESIGNER UND SYBASE ASE.....</b>	<b>89</b>
<b>B SQL-STATEMENTS ZUM ERZEUGEN DER DATENBANK .....</b>	<b>91</b>
<b>C COLD FUSION-DATENBEARBEITUNGSFORMULARE .....</b>	<b>95</b>
1 ARTIKEL_DELETE.CFM.....	95
2 ARTIKEL_FRM.CFM.....	95
3 ARTIKEL_UPDATE.CFM .....	95
4 ARTIKEL_UPDATE_FRM.CFM .....	95
5 ARTPRODFOTO_UPDATE.CFM.....	96
6 ARTPRODFOTO_UPDATE_FRM.CFM .....	96
7 BILD_DELETE.CFM.....	97
8 BILD_FRM.CFM.....	97
9 BILD_INSERT.CFM.....	97
10 BILD_INSERT_FRM.CFM.....	97
11 BILD_UPDATE.CFM .....	99
12 BILD_UPDATE_FRM.CFM.....	102
13 BILDKATEGORIE_DELETE.CFM.....	103
14 BILDKATEGORIE_FRM.CFM.....	104
15 BILDKATEGORIE_UPDATE.CFM .....	104
16 BILDKATEGORIE_UPDATE_FRM.CFM.....	104
17 BLOB_DELETE.CFM.....	105
18 BLOB_FRM.CFM.....	105
19 BLOB_INSERT.CFM.....	106
20 BLOB_INSERT_FRM.CFM.....	108
21 BLOB_UPDATE.CFM .....	108
22 BLOB_UPDATE_FRM.CFM .....	109
23 BLOB_VIEW.CFM .....	111
24 CONST.CFM .....	111
25 DATEITYP_FRM.CFM.....	111
26 DATEITYP_UPDATE.CFM .....	112
27 DATEITYP_UPDATE_FRM.CFM .....	112
28 DIGIGERAET_DELETE.CFM .....	113
29 DIGIGERAET_UPDATE.CFM.....	113
30 DIGIGERAET_UPDATE_FRM.CFM .....	113
31 DIGITALISIERGERAET_FRM.CFM.....	113
32 ERSTELLER_DELETE.CFM.....	114
33 ERSTELLER_UPDATE.CFM .....	114
34 ERSTELLER_UPDATE_FRM.CFM .....	114
35 FOOT.CFM.....	115
36 HEAD.CFM .....	115
37 PRODFOTO_UPDATE.CFM.....	115
38 PRODFOTO_UPDATE_FRM.CFM .....	115
39 PRODUKTFOTO_FRM.CFM .....	116
40 START.CFM .....	117
41 STICHWORT_DELETE.CFM.....	118

---

42 STICHWORT_FRM.CFM .....	118
43 STICHWORT_UPDATE.CFM .....	118
44 STICHWORT_UPDATE_FRM.CFM.....	118
45 SUCHE.CFM.....	119
46 SUCHE_FRM.CFM.....	119
<b>D INHALT DER BEIGEFÜGTEN CD-ROM.....</b>	<b>121</b>
<b>ABBILDUNGSVERZEICHNIS .....</b>	<b>122</b>
<b>ABKÜRZUNGSVERZEICHNIS .....</b>	<b>123</b>
<b>LITERATURVERZEICHNIS.....</b>	<b>124</b>

# 1 Einführung

## 1.1 Archivierung multimedialer Daten

Ein relationales Datenbank Management Systeme (RDBMS)<sup>1</sup> ist heutzutage in nahezu jedem Unternehmen das zentrale Element der DV-Infrastruktur. Längst haben RDBMS die dateiorientierte Datenhaltung abgelöst ([Klei97a]). Auch im Bereich der unternehmensweiten Datenhaltung in Zusammenhang mit integrierter Unternehmenssoftware wie SAP R/3 oder Baan sind sie eine wichtige Stütze des ganzen Systems. Aber auch in kleinen Firmen und Handwerksbetrieben sowie in Privathaushalten nimmt ihre Verbreitung in Form von Produkten wie z.B. Microsoft Access zu.

Eine vollkommen andere Technik hat in den letzten Jahren eine noch viel rasantere Entwicklung durchgemacht: das Internet. Es hat sich von einem reinen Forschungs- und Entwicklungsnetz zu einem Medium entwickelt, daß heute gerade im Unterhaltungs- und Informationssektor eine wichtige Rolle spielt. Speziell das multimediale Medium World Wide Web (WWW) als Teil des Internet hat diesen Trend geradezu boomartig ausgelöst. Längst sind die Benutzer des Internet nicht mehr ausschließlich Wissenschaftler, Studenten und Techniker, sondern oft Personen aus nicht-technischen Berufen. Fast jeder sechste Deutsche zwischen 14 und 59 Jahren hat heute einen Internet-Zugang. Ein Drittel aller Internet-Benutzer sind Frauen ([WWW05a]). Das Internet wird somit immer mehr ein Medium für den sogenannten „Otto Normalverbraucher“. Und deshalb ist es naheliegend, daß immer mehr Unternehmen in Deutschland die Möglichkeiten des WWW entdecken und versuchen, an diesem neuen Medium teilzuhaben.

Bestanden die ersten Angebote der Firmen im WWW zunächst aus reinen Informationen über das Unternehmen und deren Tätigkeitsfelder, so werden seit einiger Zeit auch immer mehr Artikel- und Zusatzinformationen publiziert und Produkte direkt zum Verkauf per WWW angeboten. Vorreiter sind hier die großen deutschen Handelsunternehmen wie Karstadt (<http://www.myworld.de>), der Otto-Versand (<http://www.otto.de> bzw. <http://www.shopping24.de>) oder Quelle

---

<sup>1</sup> Häufig auch „Relationale Datenbank Verwaltungssysteme“ (RDBVS) genannt. Im weiteren Verlauf wird aber die Abkürzung RDBMS oder DBMS als allgemeine Abkürzung für Datenbank Management Systeme verwenden.

(<http://www.quelle.de>). Auch international operierende Unternehmen wie der amerikanische Buchversand Amazon (<http://www.amazon.com> bzw. <http://www.amazon.de>), der seine Produkte ausschließlich über das Internet vertreibt, bestätigt diesen Trend. Sie sind nur einige Beispiele für Firmen, die das aufkommende E-Commerce-Geschäft<sup>2</sup> im Internet frühzeitig erkannt haben.

Für einen „Internet-Shop“ müssen neben den direkten Artikeldaten (Artikelnummer und -name, Verkaufspreis, Lagerbestand, Mehrwertsteuersatz usw.) auch unterschiedlichsten Medien zugehörige Informationen (z.B. Produktfoto, Anwendungsbild, Datenblätter, Handbücher oder Betriebsanleitungen, weiterführende Dokumentationen, Treiber-Software u.v.m.) elektronisch zur Verfügung stehen, verwaltet, aktualisiert und innerhalb des Internetangebots veröffentlicht werden.

Diese Tatsache führt zu den folgenden zwei Kernproblemen:

- Wie können verschiedenartigste multimediale Daten praktikabel und leicht wieder auffindbar archiviert werden?
- Wie können diese Archive an das Internet oder ein Intranet angebunden werden?

Für die Speicherung und Handhabung von großen Datenmengen werden gewöhnlich DBMS eingesetzt. Doch handelt es sich bei den Daten, die im WWW veröffentlicht werden sollen, nicht nur um reine Zahlen oder Bezeichnungstexte, sondern um umfangreiche multimediale Daten. Diese Daten zeichnen sich dadurch aus, daß sie nicht in den üblichen Standarddatentypen wie z.B. `integer`, `numeric` oder `char` vorliegen, sondern daß sie vielmehr aus einer Aneinanderreihung von Bytes oder Binärinformationen bestehen, die einer Interpretation bedürfen. Diese Interpretation übernehmen in der Regel Applikationen wie Textverarbeitungen, Bild- oder Grafikbearbeitungsprogramme, Video- und Audiorecorder oder entsprechende Anzeige- und Abspielprogramme ([Meye91a]).

Ein weiteres Charakteristikum ist die Größe der Daten. Werden für die Standarddatentypen in der Regel zwischen einem und 255 Byte Speicherplatz benötigt, so sind einfache Bilder, die in einer Webseite verwendet werden, meist zwischen einem und 200 KB groß. Im Bereich des professionellen Grafikdesigns findet man oft Bilddateien mit einer Datenmenge von zehn bis 500 MB. Auch

---

<sup>2</sup> E-Commerce wird hier als Oberbegriff für den Handel per Internet verwendet.

Audiodateien erreichen schnell eine Größe von mehreren Megabyte, während bei Videos schon die Gigabytegrenze überschritten wird.

Textdokumente bestehen aufgrund ihrer Speicherungsform in elektronischen Geräten zwar zunächst aus kleineren Datenmengen, doch hinterlegen professionelle Textverarbeitungsprogramme zusätzliche Layoutinformationen für den darzustellenden Text im Dokument. Außerdem ist es heutzutage oft üblich, in die Texte auch Bilder oder Audio- und Videodateien einzubinden. Dadurch kann der Umfang von diesen Dokumenten ebenfalls ganz erheblich anwachsen ([Meye91a], Kap. 3).

All diese verschiedenen Multimediadaten lassen sich aus Sicht eines DBMS abstrakt als binäre Datenobjekte (engl.: Binary Large Object = BLOB) zusammenfassen. Wie diese BLOBs in einer Datenbank gespeichert und wieder ausgelesen werden können, und wie ein entsprechender Datenbankentwurf aussehen kann, der es ermöglicht, gezielt ein BLOB wiederzufinden ist Gegenstand dieser Arbeit.

Der zweite Problembereich besteht in der Anbindung einer Datenbank an das Internet oder ein Intranet. Dies ist nicht unmittelbar über einen Webserver<sup>3</sup> möglich, sondern es wird ein zusätzliches Programm benötigt. Die Aufgabe von Webservern ist es, Anfragen eines Clients, die per HTTP-Request gestellt werden, zu beantworten ([RFC1945]). In einem HTTP-Request ist eine Art Adresse eines Netzwerkdokuments enthalten, der sogenannte URL (Uniform Resource Locator). Diese Adresse verweist auf ein Dokument, daß der Webserver an den Client übertragen soll. Der URL beinhaltet, neben dem zu verwendenden Übertragungsprotokoll (i.d.R. HTTP), den Namen eines Servers sowie den Verzeichnispfad und den Dokumentennamen.

Soll nun statt auf ein statisches, auf einem Filesystem gespeichertes Dokument ein Datenbankzugriff erfolgen und das Ergebnis dieses Zugriffs an den Client übertragen werden, so benötigt man ein Gatewayprogramm, über das der Webserver die Datenbankanfrage weiterleiten kann ([Gree98a]). Ein Beispiel für eine entsprechende Vorgehensweise wird in Kapitel 3 gegeben.

---

<sup>3</sup> Webserver wird in dieser Arbeit als Oberbegriff für Server verwendet, die mittels Hypertext-Transfer-Protocol (HTTP) gestellte Anfragen entsprechend bearbeiten und beantworten ([RFC1945], Kap. 1.2).

Sowohl der Datenbankentwurf als auch die Anbindung der Datenbank an einen Webserver wurde in Zusammenarbeit mit der Firma Spinnrad in Gelsenkirchen umgesetzt. Diese hat im Rahmen ihrer Neu- und Umstrukturierung der Datenverarbeitungsinfrastruktur sowie der Entwicklung eines umfangreichen Internetangebots mit integriertem E-Commerce-System die Notwendigkeit eines Archivierungssystems für Binary Large Objects erkannt. Vor allem im Zusammenhang mit der geplanten virtuellen Verkaufsstelle im WWW ([Belo98a]) ist es naheliegend, neben den Artikelinformationen auch die Produktfotos mit Hilfe einer Datenbank zu verwalten. Weitere Einsatzfelder können dann noch die Unterstützung bei der Erstellung von Katalogen, Broschüren, Informationsblättern usw. sein. Der Entwurf und die Umsetzung dieser Printmedien wird von professionellen Designer und Werbeagenturen durchgeführt. Dabei hat sich jedoch herausgestellt, daß vor allem die Informationsbeschaffung und das Zusammenstellen von immer wieder benötigten Angaben zu Produkten oder Produktgruppen besonders zeitaufwendig ist. Auch hier ist es naheliegend, alle vorhandenen Informationen so zu archivieren, daß sie leicht wieder auffindbar sind.

Zusammenfassend läßt sich sagen, daß eine Datenbank benötigt wird, die binäre Datenobjekte wie Bilder (speziell Produktfotos) und andere aufnehmen kann. Diese anderen Objekte sind zum Zeitpunkt der Entwicklung noch nicht unbedingt genau spezifiziert und teilweise sogar unbekannt. Denkbar sind alle möglichen multimedialen Daten. Der Zugriff auf die Daten in der Datenbank muß über das firmeneigene Intranet erfolgen. Außerdem sollen bestimmte Daten auch für Benutzer des WWW zugänglich sein.

## **1.2 Die Spinnrad GmbH in Gelsenkirchen**

### 1.2.1 Das Unternehmen

Die Spinnrad GmbH ist ein Einzelhandelsunternehmen, daß Drogerieartikel wie Kosmetik, Parfüms und Reinigungsmittel sowie Haus- und Gartenprodukte, Geschenkartikel, Nahrungsergänzungsmittel, Lebensmittel, Bücher u.v.m. verkauft. Dabei steht der Leitsatz „Für Umwelt und soziale Verantwortung“ bei der Auswahl der Produkte immer im Mittelpunkt und hebt Spinnrad von anderen Drogerien und Geschenkartikelläden ab. Das Produktsortiment umfaßt ca. 5000 Artikel, wovon fast 3500 dauerhaft im Programm sind. Die anderen Artikel sind zum Großteil Saisonartikel und Produkte, die aus verschiedenen Gründen zwischenzeitlich nicht verkauft werden.

In Deutschland gibt es derzeit ca. 200 Verkaufsstellen, die von der Zentrale in Gelsenkirchen aus beliefert werden. Im Verwaltungsbereich der Zentrale arbeiten ca. 120 Mitarbeiter.

Seit 1997 baut Spinnrad ein umfangreiches Internet-Angebot auf. Parallel dazu soll auch die interne Datenverarbeitungs- und Kommunikationsstruktur auf die Internettechnologie umgestellt werden. Dazu wurde in einen Ausbau der DV-Infrastruktur investiert.

### 1.2.2 Technische Voraussetzungen

Das Herzstück der neuen DV-Anlage der Firma Spinnrad bildet das RDBMS Sybase Adaptive Server Enterprise (ASE) Version 11.5 unter dem Betriebssystem Sun Solaris 2.5.1. Dieses ist auf einer Sun E3000 mit zwei Prozessoren, einem Gigabyte Hauptspeicher und ca. 60 GB Plattenspeicher auf insgesamt zehn Festplatten installiert. Der Server wird die Datenbanken für das Warenwirtschaftssystem, die Lagerverwaltung und das BLOB-Archivierungssystem aufnehmen. Letzteres wird aufgrund der zu erwartenden Datenmenge nach der Testphase auf ein eigenes externes RAID-Subsystem ausgelagert. Auf einem weiteren Server, einer Sun E450 mit einem Prozessor und ca. 12 GB Plattenspeicher, läuft ein E-Commerce-System der Intershop AG, das auf einem Sybase Adaptive Server 11.0 basiert ([Belo98a]). Dieser Rechner dient zusätzlich über das Programmpaket Netscape Suitespot Pro und

dem darin enthaltenen Netscape Enterprise Server als Webserver. Über diesen Webserver kann wiederum mittels des Programmes Cold Fusion Application Server der Firma Allaire auf die Sybase Datenbanken zugegriffen werden. Die Funktionsweise eines Application Servers und das Zusammenspiel von Webserver, Application Server und Datenbankserver wird in Kapitel 3 erklärt.

Der Sybase ASE stellt die Datenbanksprache Transact-SQL zur Verfügung, die kompatibel zum SQL/92-Standard ist ([Date93a]) und darüber hinaus noch Erweiterungen bietet ([Syba97d]).

Als Clients werden IBM-kompatible PCs mit dem Betriebssystem Microsoft Windows 95 eingesetzt. Sie sind über ein 10- bzw. 100-Mbit/s Ethernet mit den Servern verbunden und verwenden das TCP/IP-Protokoll zur Datenübertragung. Als grafische Benutzeroberfläche für den Datenbankzugriff dient der Webbrowser Netscape Navigator 4.x.

Folgende Programme stehen für die Entwicklung der Datenbank und der Datenerfassungs- und -bearbeitungsformulare zur Verfügung:

- Das CASE-Tool „Power Designer Data Architect 6.0“ von Powersoft/Sybase
- Der SQL-Abfrageeditor „SQL-Advantage 11.5“ von Sybase
- Das Datenbankverwaltungsprogramm „Sybase-Central 2.3.01“ von Sybase
- Der HTML- und CFML-Editor „Cold Fusion Studio 3.1“ von Allaire
- Der Application Server „Cold Fusion Engine 3.1.0.0 Professional für Windows 95/NT“ von Allaire
- ODBC-Treiber von Intersolv zur Verbindung des lokalen Cold Fusion Application Servers mit den Sybase Datenbanken

Parallel zu dieser Arbeit wird bei der Firma Spinnrad sowohl das Warenwirtschaftssystem als auch die Lagerverwaltung von einer dateiorientierten Datenbank auf das Sybase RDBMS umgestellt.

## 2 Datenbankentwurf

Obwohl ein konzeptioneller Datenbankentwurf theoretisch hardware- und plattform-unabhängig ist, sind schon beim Entwurf einige Vorüberlegungen zum später genutzten DBMS anzustellen. Die zu entwerfende Datenbank wird auf dem Sybase Adaptive Server Enterprise 11.5 implementiert, der neben den Standarddatentypen auch einige spezielle SQL-Datentypen zur Verfügung stellt. Das zum Entwurf verwendete grafische CASE-Tool Power Designer Data Architect verwendet dagegen einige Datentypen, die sich von denen des Sybase-DBMS unterscheiden<sup>4</sup>. Deshalb wurde darauf geachtet, im Modellentwurf nur Datentypen zu verwenden, denen ein passender Typ im Adaptive Server zugeordnet werden kann. Auf diese Problematik wird bei der Umsetzung vom konzeptionellen in das physikalische Datenmodell im Kapitel 2.3 noch genauer eingegangen.

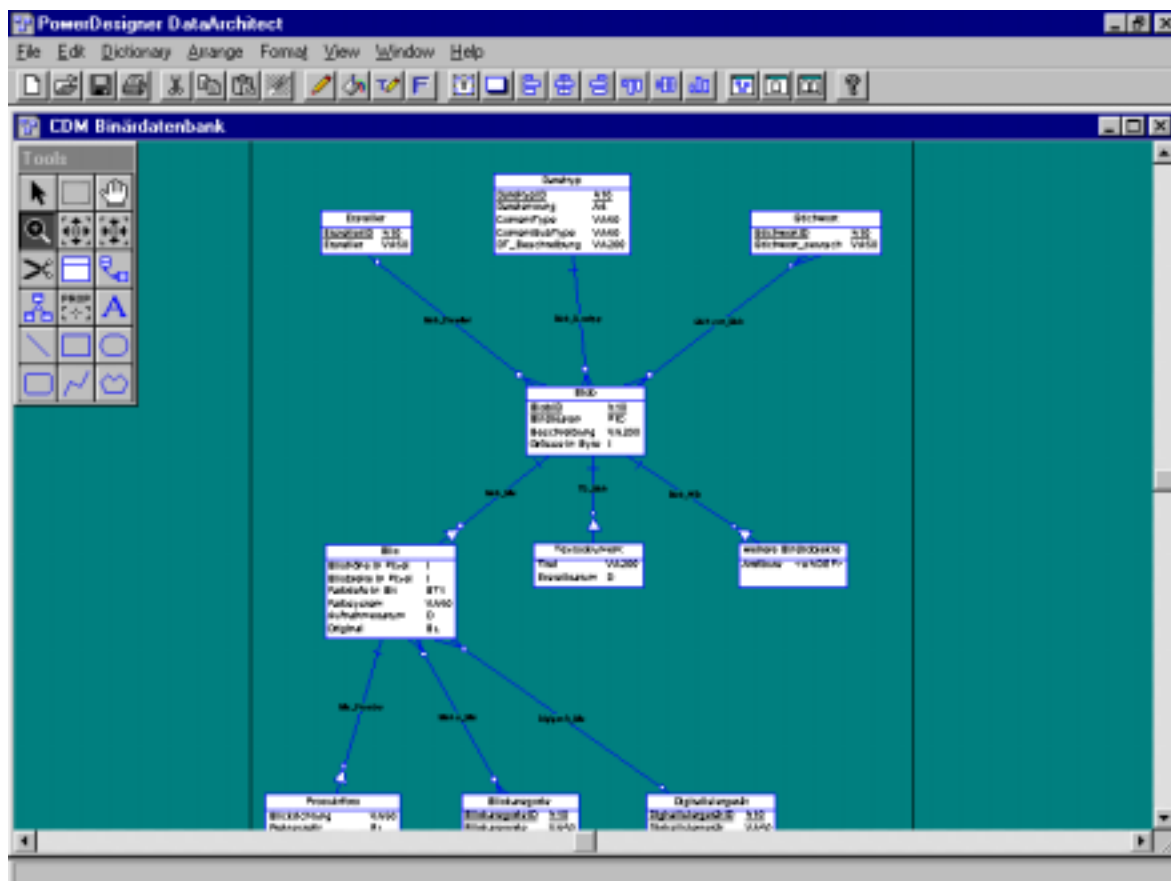
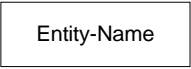
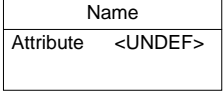
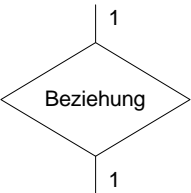
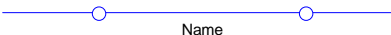
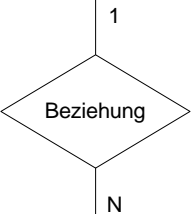
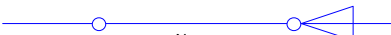
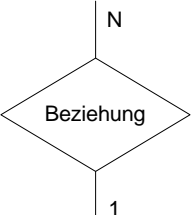

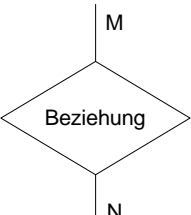



Abbildung 2-1: Grafische Benutzeroberfläche des Power Designer Data Architect

Mit dem Power Designer Data Architect kann man das Datenmodell grafisch aufbereiten, indem man die verschiedenen Symbole für Entitätstypen auf dem

Arbeitsblatt zusammenstellt und die Relationen durch Verbinden der Entitäten erzeugt. Weitere Informationen zu den diversen Objekten werden dann in speziellen Eingabefenstern festgelegt. Der Power Designer hält sich nur unzureichend an die Notation für Entity-Relationship-Modelle nach [Chen91a]. Die entsprechenden Symbole werden in der folgenden Tabelle kurz erläutert:

Bezeichnung	Symbol im E-R-Modell	Symbol im Power Designer
Entitätstyp		
1:1-Beziehung		
1:n-Beziehungstyp		
n:1-Beziehungstyp		
m:n-Beziehungstyp		

<sup>4</sup> Siehe dazu Anhang A, Powerdesigner- und Sybasedatentypen.

Bezeichnung	Symbol im E-R-Modell	Symbol im Power Designer
Existenzabhängiger Beziehungstyp und ein schwacher Entitätstyp		
Attribut einer Entität		
Attribut einer Relation		nicht möglich!

Aus dem fertigen Datenmodell kann automatisch ein physikalisches Datenmodell erzeugt werden. Dazu muß zunächst das DBMS festgelegt werden, auf dem die Datenbank später implementiert werden soll. Anschließend kann das Programm entweder direkt die Datenbank auf dem Server generieren oder eine Datei mit SQL-Anweisungen erstellen, die dann wiederum zum Erzeugen der Datenbank dienen.

Weitere Datenbanken werden nach der Umstellung des Warenwirtschaftssystems und der Lagerhaltung auf dem selben Server vorhanden sein. Somit wird es möglich sein, Verknüpfungen zwischen der Multimediatendbank und Tabellen in anderen Datenbanken herzustellen. So kann man beispielsweise einem Artikel Bilder zuordnen, um neben dem Bild auch Informationen wie Verkaufspreise, Lagerbestände usw. mit auszugeben. Auf diese Möglichkeiten und den Grund ihrer Ausnutzung wird im weiteren Verlauf noch näher einzugehen sein.

### 2.1 Konzeptionelles Datenmodell zur Aufnahme von BLOBs

Das konzeptionelle Datenmodell wird alle Entitätstypen mit den darin vorkommenden Attributen und deren Datentypen sowie die Relationstypen mit den entsprechenden Kardinalitäten enthalten. Alle Objekte im Power Designer haben einen Namen und einen Code, der ein aus dem Namen generierter Bezeichner ohne

unerlaubte (Sonder-) Zeichen ist<sup>5</sup>. Dieser Bezeichner wird später sowohl im physikalischen Datenmodell als auch in der Datenbank verwendet ([Powe97a]).

### 2.1.1 Domains<sup>6</sup>

Der Power Designer Data Architect erlaubt es, eine Art benutzerdefinierte Datentypen anzulegen, die in dem Programm als „Domains“ bezeichnet werden. Neben dem eigentlichen Datentyp können für jede Domain weitere Integritätsbedingungen festgelegt werden. Dies sind unter anderem: Standardwert, Minimum- und Maximumwerte oder eine Liste mit gültigen Werten, außerdem eine Maßeinheit und das Ausgabeformat, sowie Regeln, auf die ein Attribut überprüft werden soll.

Die Domains ermöglichen es, häufig verwendeten Attributen gleichen Typs schnell und einfach Integritätsbedingungen zuzuweisen, da sich Änderungen der Eigenschaften einer Domain auf alle Objekte auswirken, die dieser Domain zugeordnet sind.

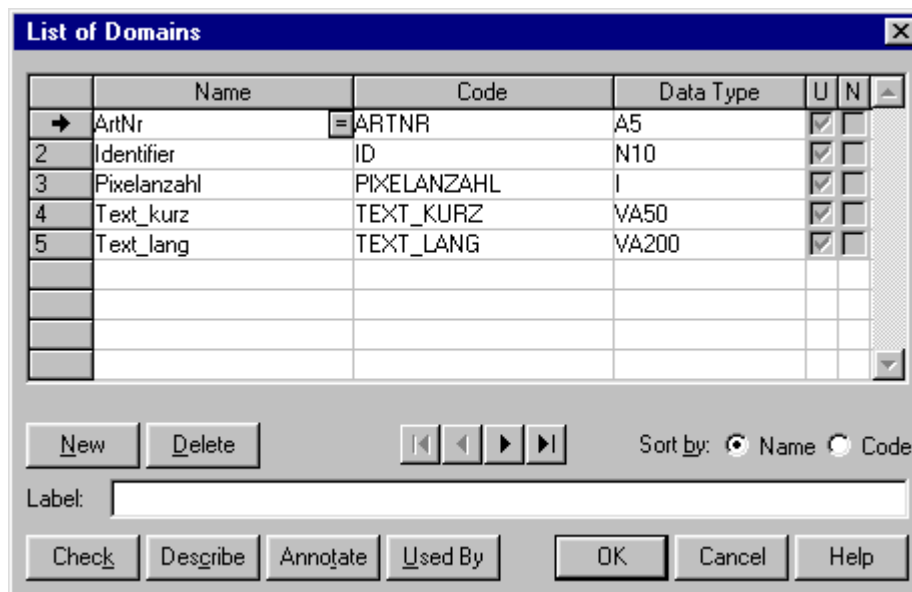


Abbildung 2-2: Liste der Domains im Power Designer Data Architect

Für die Artikelnummer wird die Domain *ArtNr* definiert. Der Typ wird entsprechend dem Datentyp im bestehenden Warenwirtschaftssystem bei der Firma Spinnrad auf fünf alphanumerische Zeichen festgelegt. Obwohl die bisher vergebenen Artikelnummern aus einer fünfstelligen Zahl - also ohne Buchstaben - bestehen, wird dieser

<sup>5</sup> Für den Code erlaubte Zeichen sind alle Buchstaben von A bis Z ('a'-'z', 'A'-'Z') ohne Umlaute und ohne 'ß', die Ziffern ('0'-'9') und der Unterstrich ('\_').

Typ verwendet, damit eine direkte Verknüpfung mit Artikeln in der Warenwirtschaft ohne vorherige Typkonvertierung möglich ist. Integritätsbedingungen werden keine festgelegt.

Die Domain *ArtNr* wird im gesamten Modell zwar nur einmal verwendet, doch erscheint es sinnvoll, für die Artikelnummer eine eigene Domain einzuführen, da die geplante Umstellung des Warenwirtschaftssystems wahrscheinlich auch eine Änderung des Datentyps für die Artikelnummer mit sich bringen wird. Eine entsprechende Änderung ist dann schnell umsetzbar.

Eine weitere Domain wurde für die Attribute eingeführt, die als Identifikationsnummer (ID) dienen werden. Diese ID dient als eine Art künstlicher Schlüssel, der eventuell aus mehreren Attributen zusammengesetzte Schlüssel ersetzt und so die Handhabung der einzelnen Datensätze deutlich vereinfacht. Als Datentyp wird für die Domain *ID* eine 10stellige Zahl fester Länge ohne Nachkommastellen festgelegt (`numeric(10,0)`). Dieser Datentyp entspricht dem Typ *Serial*, dem bei der Umsetzung vom konzeptionellen in ein physikalisches Datenmodell eine besondere Bedeutung zukommt. Mehr dazu im Kapitel 2.3.4.

Für die Aufnahme von Bezeichnern, Namen und kurzen Beschreibungen wurden die beiden Domains *Text\_kurz* und *Text\_lang* eingeführt. Attribute, denen diese Domains zugeordnet sind, können bis zu 50 bzw. 200 alphanumerische Zeichen aufnehmen.

Aufgrund des häufigen Vorkommens von Angaben in Pixeln (z.B. Bildhöhe und -breite) wird hierfür ebenfalls eine eigene Domain *Pixelanzahl* definiert. Entsprechende Attribute erhalten den Datentyp *Integer*. Über die Integritätsbedingungen wird festgelegt, daß der Wert größer oder gleich null ( $\geq 0$ ) sein muß, da negative Längenangaben nicht sinnvoll sind.

### 2.1.2 Binary Large Object (BLOB)

Der zentrale Entitätstyp des Datenmodells ist das „Binary Large Object“. Die später daraus resultierende Tabelle wird die einzige sein, die alle in der Datenbank gespeicherten binären Objekte, enthält. Dabei ist es beliebig, ob es sich um Bilder, Textdokumente, Video, Audio oder sonstige Arten multimedialer Daten handelt. Die

---

<sup>6</sup> Auf die Übersetzung des Begriffes „Domain“ wird an dieser Stelle verzichtet, da es kein deutsches

Entität besteht aus den eigentlichen binären Daten, einem beschreibenden Namen, der Anzahl der für die Speicherung benötigten Bytes (Größe in Byte) sowie einem künstlichen Schlüssel, der BLOB-ID.

Blob	
BlobID	N10
Binärdaten	PIC
Beschreibung	VA200
Grösse in Byte	1

Abbildung 2-3: Die Entität *BLOB*

Für die Aufnahme der Binärdaten wird der Power Designer-Datentyp `Picture` verwendet, der im physikalischen Modell und im Sybase ASE als `image` bezeichnet wird. Während die Datentypen `binary` und `varbinary` nur maximal 255 Byte aufnehmen können, stellt der Typ `image` bis zu 2 GB an Speicherplatz pro Attributwert zur Verfügung. In den Datenfeldern vom Typ `image` können nicht nur Bilder gespeichert werden, sondern jede Art von binären Daten. Eine Verwendung in SQL-Statements ist genauso wie bei Spalten anderer Datentypen möglich. Allerdings können `image`-Spalten für folgende Zwecke nicht eingesetzt werden:

- In `order by`-, `compute`-, `group by`- und `union`-Klauseln
- In einem Index
- In Unterabfragen und Joins
- In einer `where`-Klausel, außer mit dem Schlüsselwort `like`
- Mit dem Verkettungsoperator `+`

Die *BLOBID* wird der Domain *ID* zugeordnet, die *Beschreibung* der Domain *Text\_lang*. Die *Größe in Byte* ist vom Datentyp `Integer` mit der Integritätsbedingung „Mindestwert = 0“ (d.h. der Wert muß größer oder gleich Null sein).

Zur genaueren Spezifikation, um welche Art der Daten es sich in dem Feld *Binärdaten* handelt, wird im Kap. 2.1.3 die Entität *Dateityp* eingeführt. Außerdem werden weitere Entitäten benötigt, die je nach Art der BLOBs entsprechende Zusatzinformationen aufnehmen können. D.h. BLOBs, die als Bilder oder Texte interpretiert werden können, haben weitere grundlegende Eigenschaften, die zu den Objekten abgespeichert werden müssen. Beispiele dafür sind die Entitäten *Bild* (Kap. 2.1.4) und *Textdokument* (Kap. 2.1.5).

### 2.1.3 Inhaltstyp der Binärdaten

Wie schon erwähnt lassen sich die eigentlichen Daten, die in einem BLOB gespeichert sind, nicht ohne weitere Informationen interpretieren. Erst durch das Wissen, von welchem Typ sie sind, kann auch eine Interpretation erfolgen und so aus dem binären Datenstrom z.B. ein Bild werden. Wenn dem System also beispielsweise bekannt ist, daß ein BLOB ein im „Graphics Interchange Format“ (GIF) ([Comp87a]) gespeichertes Bild ist, so kann ein entsprechendes Programm oder ein passender Filter verwendet werden, um die Bildinformationen richtig zu interpretieren und das Bild anzuzeigen.

Die Information über den Typ einer Datei werden in MS-DOS- und Windows-basierenden Betriebssystemen in der Endung des Dateinamens festgelegt. Dafür stehen bis zu drei Buchstaben zur Verfügung. In Unix-Betriebssystemen dagegen gibt es zwar die Möglichkeit, Dateien eine Endung beliebiger Länge anzuhängen, allerdings ist die Nutzung eher freiwillig. Die Endung wird nicht von allen Programmen benutzt oder interpretiert.

Endungen für Bilddateien in MS-DOS-Systemen sind z.B. GIF für „Graphics Interchange Format“ oder BMP für das „Windows-Bitmap-Format“.

Der Typ einer Datei, die von einem Webserver übertragen wird, wird in einem speziellen Feld im HTTP-Header angegeben: dem „Contenttype“-Feld, das den Inhaltstyp und einen Untertyp (Contentsubtype) enthält. Der Typ wird in Klartext in der Form „Contenttype/Contentsubtype“ übertragen. ([RFC1945])

In den Betriebssystemen Windows 95 und NT gibt es in der Registrierdatenbank für jeden bekannten Contenttype eine Eintragung mit der zugehörigen Dateiendung und einer Applikation, die standardmäßig zum Anzeigen oder Bearbeiten der Daten verwendet werden soll.

Der Contenttype für GIF-Bilder lautet z.B. „image“ und der Contentsubtype „gif“. Für Windows-Bitmap-Bilder ist der Contenttype ebenfalls „image“, der Contentsubtype heißt dann „x-MS-bmp“. Unbekannte Typen sollen vom Webserver laut [RFC1945] mit „application/octet-stream“ spezifiziert werden. Der empfangende Client muß dann mittels der Dateiendung im URL oder durch Benutzerbefragung klären, wie er die Daten interpretieren soll.

Dateityp	
DateitypID	N10
Dateiendung	A4
ContentType	VA50
ContentSubType	VA50
DT_Beschreibung	VA200

Abbildung 2-4:

Die Entität *Dateityp*

Die Entität *Dateityp* erhält die Attribute *Dateiendung*, *ContentType* und *ContentSubType* sowie ein Textfeld, in dem eine kurze Beschreibung des Dateityps hinterlegt werden kann. Für die *Dateiendung* wird der Datentyp auf vier alphanumerische Zeichen festgelegt, da sich drei-

bzw. vierbuchstabile Endungen als Standard etabliert haben. *ContentType* und *ContentSubType* wird die Domain *Text\_kurz* zugeordnet. Somit können diese Attribute bis zu 50 alphanumerische Zeichen aufnehmen. Entsprechend wird das Beschreibungsfeld (*DT\_Beschreibung*) der Domain *Text\_lang* zugeordnet.

Da diese Entität zwei mögliche Schlüsselkandidaten hat - nämlich zum einen *Dateiendung* und zum anderen die Attribute *Contenttype* und *Contentsubtype* - wird außerdem noch ein künstlicher Entitätsschlüssel eingefügt. *DateitypID* wird natürlich die Domain *ID* zugeordnet.

Die Relation *BLOB\_Dateityp* zwischen den BLOBs und den Dateitypen ist eine 1:n-Relation. Zu jedem BLOB gehört genau ein Dateityp, während ein Dateityp zu beliebig vielen BLOBs gehören kann. Wichtig dabei ist, daß es ein BLOB ohne einen Dateityp nicht geben kann und darf, da das Wissen über die Art der Daten und damit jede Möglichkeit der Interpretation verloren geht.

#### 2.1.4 Der Entitätstyp *Bild*

Prinzipiell gibt es zwei verschiedene Arten von elektronisch verarbeitbaren Bildern, die heutzutage Verwendung finden. Dies sind Rasterbilder und Vektorgrafiken ([Meye91a], Kap. 3.2 und 3.3). Während bei Rasterbildern die Bildinformationen für jeden Bildpunkt hinterlegt werden (z.B. durch Angabe der Farbwerte für jedes Pixel), beschreiben Vektorgrafiken die einzelnen Elemente eines Bildes durch eine Art Vektor. Für beide Arten von Computerbildern gibt es inzwischen unzählige verschiedene Formate. Die meistbenutzten Rasterbildtypen sind das „Graphics Interchange Format“ (GIF) der Firma CompuServe und „Joint Photographic Experts Group Format“ (JPEG), die vor allem im WWW verwendet werden, sowie das „Tag Image File Format“ (TIFF) und das „Adobe Photoshop Format“ (PSD), die im professionellen Grafikdesign eingesetzt werden. Aus der großen Palette der Vektorgrafikformate werden bei der Firma Spinnrad vor allem das „Corel Draw Format“ (CDR) und das „Encapsulated Postscript Format“ (EPS) eingesetzt.

Unabhängig von dem jeweiligen Algorithmus haben die unterschiedlichen Formate einige gemeinsame Eigenschaften, aus denen sich folgende Attribute ableiten lassen:

Attribut	Datentyp/Domain	
Bildhöhe in Pixel	Domain: <i>Pixelanzahl</i> Datentyp: Integer IB <sup>7</sup> : >= 0	Die vertikale und horizontale Ausdehnung eines Bildes wird am Besten in Pixeln angegeben. Eine Größenangabe in Zentimetern - wie es viele Bildbearbeitungsprogramme machen - ist wenig sinnvoll, da dieser Wert relativ von der Auflösung (Pixel/cm oder Pixel/inch) der Bildausgabe abhängt.
Bildbreite in Pixel	Domain: <i>Pixelanzahl</i> Datentyp: Integer IB: >= 0	Mit dem Datentyp <code>integer</code> steht ein Wertebereich von $-2^{32}$ bis $2^{32}$ zur Verfügung. Negative Werte werden über die Integritätsbedingungen ausgeschlossen.
Farbtiefe in Bit	Datentyp: Byte IB: keine	Farbtiefe und verwendetes Farbsystem eines Bildes sind eng miteinander verknüpft. Die Angabe der Farbtiefe legt fest, aus wie vielen Farben ein Bild besteht ( $2^{\text{Farbtiefe}}$ ). Für einige Farbsysteme ist die Farbtiefe genau festgelegt.
Farbsystem	Domain: <i>Text_kurz</i> Datentyp: Variable characters (50) IB: keine	Übliche Werte für die Farbtiefe sind 1 (Schwarz/Weiß-Grafiken), 2-8 (Webgrafiken mit festgelegter Palette), 8 (256 Graustufen), 24 (RGB) und 32 (CMYK). Zwischenwerte sind zwar unüblich aber denkbar.
Aufnahmedatum	Datentyp: Date IB: keine	Dieses Attribut nimmt das Aufnahmedatum eines Bildes auf. In dem Datentyp <code>Date</code> werden Tag, Monat und Jahr gespeichert.
Original	Datentyp: Boolean IBen: nicht null	Ist das gespeicherte Bild bisher unbearbeitet und in dem Zustand, in dem es digitalisiert wurde, ist es original, also wird dieses Attribut mit dem Wert <code>true</code> gefüllt.

Die Angabe einer Bildgröße bei Vektorgrafiken gestaltet sich als schwierig, da die Bilder durch die Art der Speicherung prinzipiell auf beliebige Größe skaliert werden können. Trotzdem ist es sinnvoll, hier Werte einzutragen, weil die meisten

Vektorgrafiken von den Designern von vornherein für eine bestimmte Größe angelegt werden und eine Skalierung in den seltensten Fällen durchgeführt wird. Man könnte dann z.B. für ein im Vektorgrafik-Format gespeichertes Werbeposter (DIN A3) die Größe aus den Seitenlängen (42 cm x 29,7 cm) und der Druckauflösung (z.B. 300 dpi  $\approx$  118 Pixel/cm) berechnen. Sind die endgültigen Maße eines solchen Bildes nicht bekannt, sollten die Werte für Bildhöhe und -breite nicht gesetzt werden.

Die bisher beschriebenen Attribute haben sich bei Gesprächen mit den späteren Benutzern dieser Datenbank herauskristallisiert. Da diese Personen aber bisher keinerlei Erfahrung mit einem solchen System haben, ist es möglich, daß im späteren Betrieb des Archivierungssystems weitere Attribute gebraucht werden, die jetzt noch nicht notwendig erscheinen. Das nachträgliche Einfügen eines Attributes gestaltet sich bei RDBMS aber sehr einfach und ist jederzeit möglich. Um die prinzipielle Machbarkeit und Funktionsweise zeigen zu können und um das System sinnvoll einzusetzen, sind die oben genannten Attribute zunächst vollkommen ausreichend.

Der Entitätstyp *Bild* ist abhängig vom Entitätstyp *BLOB*, d.h. eine *Bild*-Entität kann in der Datenbank nicht ohne die zugehörige *BLOB*-Entität existieren. Außerdem stehen *BLOB* und *Bild* in einer 1:1-Beziehung zueinander, denn jedes *BLOB* kann genau ein oder kein *Bild* sein.

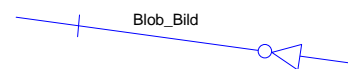


Abbildung 2-5: Symbol für die *BLOB-Bild*-Relation

### 2.1.5 Textdokumente und weitere Spezialisierungen von BLOBs

Analog zum *Bild* können bei Bedarf weitere Entitätstypen eingeführt werden, die zusätzliche Informationen zu den in *BLOB* gespeicherten Daten enthalten. Dabei können die zu ergänzenden Objekte mit Ausnahme ihrer Beziehung zu den *BLOBs* vollkommen unabhängig von den bestehenden Entitätstypen sein oder ebenfalls Beziehungen zu ihnen haben.

Als Beispiel wurde ein Textdokument mit den Attributen *Titel* und *Erstelldatum* in das konzeptionelle Modell eingefügt. Darüber könnten beliebige Textdokumente aus verschiedenen Textverarbeitungsprogrammen - wie z.B. Microsoft Word oder Word Perfect - in die Datenbank eingefügt werden. Zur Nutzung dieses Entitätstyps werden

<sup>7</sup> IB = Integritätsbedingung

dann - je nach Art der Verwendung - noch weitere Attribute notwendig sein. An dieser Stelle wird aber auf eine genauere Analyse der Anforderungen verzichtet, da dadurch keine weiteren generellen Erkenntnisse über die Speicherung von BLOBs in einer Datenbank gewonnen werden.

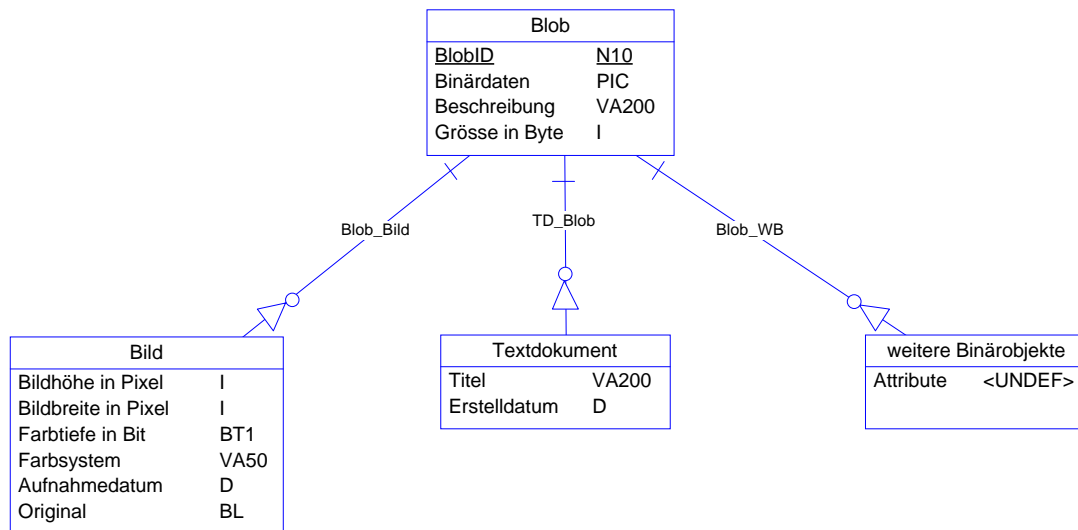


Abbildung 2-6: Entitätstypen *BLOB*, *Bild*, *Textdokument* und weitere binäre Objekte

### 2.1.6 Das Produktfoto, eine besondere Art von Bild

Den Produktfotos kommt aufgrund ihres häufigen Vorkommens in der Praxis eine besondere Rolle in diesem Modell zu. Zu den Produktfotos gehören die Bilder, auf denen ein einzelnes Produkt oder eine Gruppe von Produkten abgebildet sind. D.h. die Produkte stehen oder liegen meist vor einem einfarbigen Hintergrund oder vor einem Hintergrund mit Farbverlauf. Dieser Hintergrund wird dann häufig mit einem Bildbearbeitungsprogramm entfernt und das Bild wird freigestellt. Die Abbildungen der Produkte werden dann z.B. in Katalogen oder Broschüren mit anderen Grafiken und Hintergründen zu neuen Bildern kombiniert.

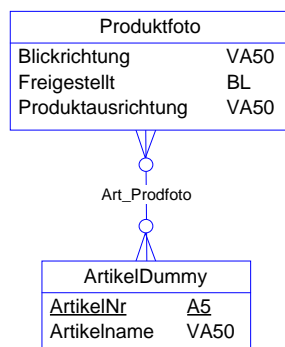


Abbildung 2-7: *Produktfoto*

Der Entitätstyp *Produktfoto* steht in einer existenzabhängigen Beziehung zum *Bild*, da ein *Produktfoto* immer auch ein *Bild* ist (und damit auch ein *BLOB*). Außerdem besteht eine m:n-Beziehung zwischen *Produktfoto* und den Produkten, da auf einem Produktfoto ein oder mehrere Artikel abgebildet sind.

Die Informationen über die verschiedenen Artikel werden bereits in einer anderen Datenbank gespeichert, so daß an dieser Stelle eigentlich

keine neue Entität *Artikel* eingeführt werden müßte. Aufgrund der Umstellung des bestehenden DBMS kann die entsprechende Beziehung aber noch nicht hergestellt werden. Darum wurde die Entität *ArtikelDummy* erzeugt. Diese enthält als Attribute nur die Artikelnummer (*ArtikelNr*) - als eindeutigen Schlüssel für die Artikel - und den *Artikelnamen* (siehe Abb. 2-7). Im späteren regulären Betrieb kann dann die Beziehung durch eine gleichartige Beziehung mit der richtigen Artikeltabelle ersetzt werden.

Zur Entität *Produktfoto* gehören die Attribute *Blickrichtung* (Domain *Text\_kurz*), *Produktausrichtung* (Domain *Text\_kurz*) und *Freigestellt* (Datentyp *Boolean*). Im Feld *Blickrichtung* soll mit wenigen Worten erklärt werden, aus welcher Richtung das Produkt aufgenommen wurde. Der Benutzer kann hier also z.B. „von vorne“ oder „von rechts oben“ usw. eintragen, um das Bild genauer zu spezifizieren. Analog dazu kann das Attribut *Produktausrichtung* Beschreibungen wie „stehend“ oder „liegend“ usw. enthalten. Diese Felder werden für das Wiederauffinden eines Bildes verwendet und später noch genauer erklärt (Kap. 2.2.1).

### 2.1.7 Herkunft digitaler Daten: Digitalisiergerät

Digitale Bilder können auf unterschiedlichste Weise entstehen. Neben den verschiedensten Arten von Scannern (z.B. Flachbettscanner, Trommelscanner usw.) setzen sich derzeit vermehrt Digitalkameras und digitale Fotosysteme durch. Je nach verwendeter Technik können die digitalisierten Bilder sehr unterschiedliche Eigenschaften haben, die nicht nur durch das Bildformat und die Auflösung beschrieben werden können. Die Information über das Digitalisiergerät, mit dem ein Bild erzeugt wurde, hilft einem Benutzer, der die Eigenschaften der verschiedenen Digitalisiergeräte und -techniken kennt, Rückschlüsse auf die Qualität eines Bildes zu ziehen.

Der dafür eingeführte Entitätstyp *Digitalisiergerät* erhält die Attribute *Digitalisiergerät* als Bezeichnungsfeld (Domain *Text\_kurz*) und den künstlichen Schlüssel *DigitalisiergerätID* (Domain *ID*).

Zusammenfassend stellt sich somit das konzeptionelle Datenmodell wie folgt dar:

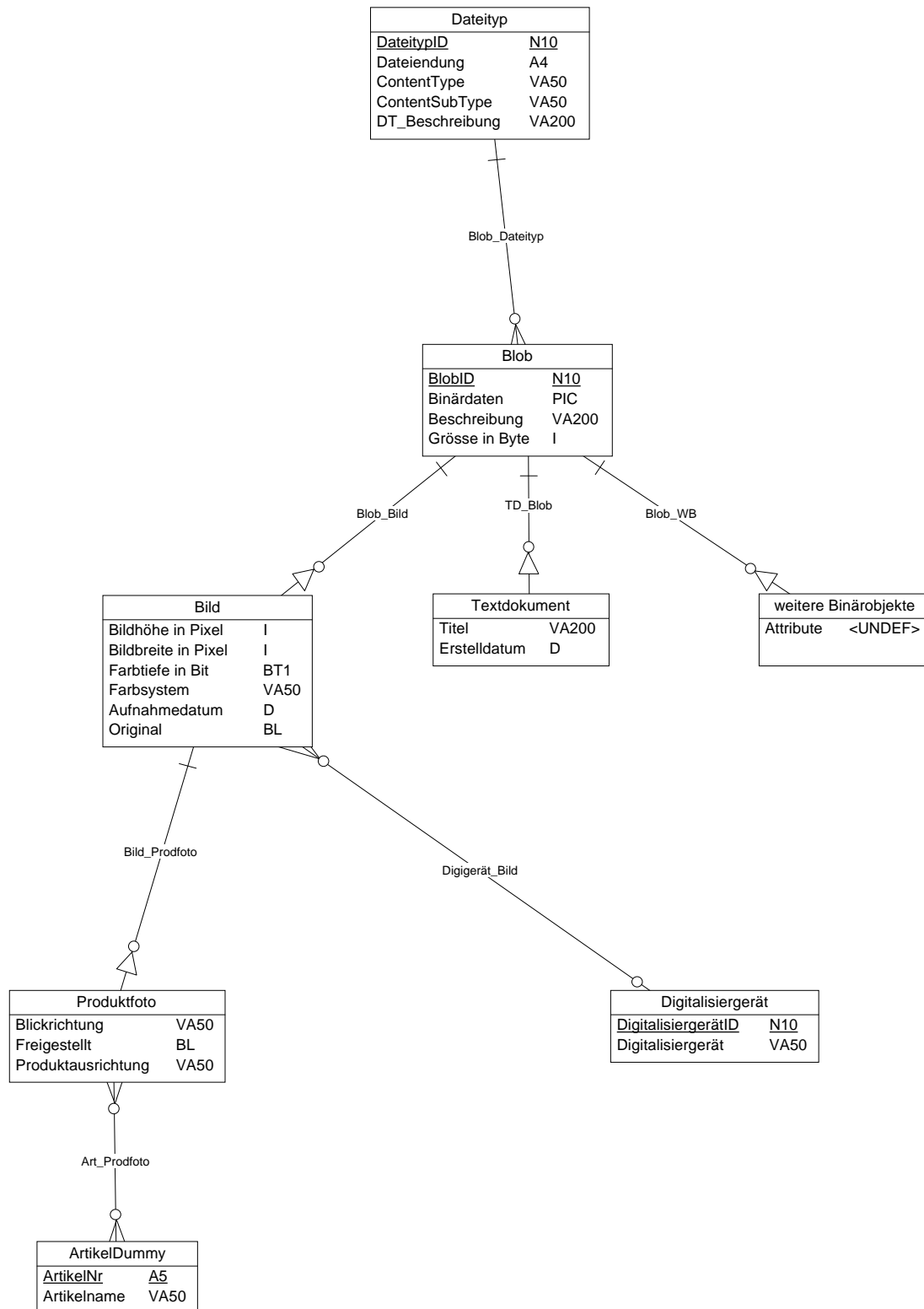


Abbildung 2-8: Konzeptionelles Datenmodell zur Archivierung von BLOBs

## **2.2 Zusätzliche Entitäten für das Information Retrieval**

Der Vorteil eines Archivierungssystems liegt darin, daß die gespeicherten Daten schnell wieder auffindbar sind. Die Suche und das Auffinden von Daten in Datenbanken nennt man Information Retrieval (IR). Mit den bisher erstellten Entitäten ist ein effizientes Information Retrieval bisher noch nicht möglich, denn kaum jemand sucht und findet ein Bild anhand seiner Größe in Byte oder in Pixeln. Darum werden im Folgenden weitere Entitätstypen vorgestellt, mit deren Hilfe es sehr einfach ist, ein Bild oder andere BLOBs wieder aufzufinden.

### **2.2.1 Die Vorgehensweise beim Information Retrieval**

Für die Suche in einer großen Menge von Bildern gibt es zwei verschiedene Motivationen. Zum einen möchte der Benutzer ein ganz bestimmtes Bild wiederfinden, das er kennt und über das ihm einige Informationen vorliegen (z.B. weil er dieses Bild selber angelegt hat oder von einem Mitarbeiter weiß, daß dieser es erstellt und in der Datenbank archiviert hat). Andererseits kommt es sehr häufig vor, daß ein Designer kein konkretes Bild sucht, sondern nur eine gewisse Vorstellung von einem Bild hat, daß er beispielsweise in ein zu entwerfendes Printmedium (Katalog, Broschüre, Infoblatt usw.) einbauen möchte. Je nachdem wie genau die Vorstellung des Designers zu dem gesuchten Motiv ist, gibt er mehr oder weniger Kriterien für die Suche an. Diese treffen in der Regel nicht nur auf ein Bild zu, sondern auf mehrere. Dem Benutzer muß es dann ermöglicht werden, die Ergebnismenge Element für Element zu durchsuchen, um daraus schließlich ein Bild auswählen zu können (dieser Vorgang wird auch „browsen“ genannt [Mey91a]).

Um diesen beiden verschiedenen Motivationen zur Bildsuche gerecht zu werden, muß dem Benutzer des Archivierungssystems freigestellt werden, nach welchen Eigenschaften eines BLOBs gesucht werden soll. Prinzipiell kann jede in der Datenbank gespeicherte Information über ein BLOB auch zum Wiederauffinden verwendet werden.

Die Entitäten *Stichwort* (Kap. 2.2.2), *Ersteller* (Kap. 2.2.3) und *Dateityp* werden für die Suche eines *BLOBs* verwendet, die Attribute *Beschreibung* und *Größe in Byte* sollten dann als zusätzliche Informationen beim Browsen angezeigt werden und so die Auswahl eines *BLOBs* unterstützen.

Bei den Bildern kommt die *Bildkategorie* und bei den Produktfotos die zugeordneten Artikel für die Suchanfrage hinzu. Die Attribute von *Bild* und *Produktfoto* sind dann die Zusatzinformationen, die beim Browsen mit angezeigt werden sollten.

### 2.2.2 Stichwort

Den Inhalt eines Bildes mit Worten zu erklären ist sehr schwierig. Wie das Sprichwort „Ein Bild sagt mehr als tausend Worte“ andeutet, ist es kaum möglich, den gesamten Inhalt eines Bildes ausreichend zu beschreiben. Die Beschreibung eines Bildes ist außerdem meist sehr subjektiv, da ein Betrachter nur die Dinge beschreibt, die er für wichtig hält, obwohl ein Bild aus Sicht einer anderen Person oft noch mehr aussagt. Noch viel komplizierter wird es bei Audio- oder Videodaten sowie Textdokumenten mit eingebetteten multimedialen Elementen. Hinzu kommt, daß eine Beschreibung für das Wiederauffinden der Daten eher hinderlich ist, da umgangssprachlicher Text von kaum einem System geeignet ausgewertet - geschweige denn interpretiert - werden kann. Somit bleibt nur die Möglichkeit, diese Interpretation einem Benutzer des Systems zu überlassen, der sich alle Beschreibungen durchlesen müßte, um zu entscheiden, ob er die gesuchten Multimediadaten vorliegen hat. Mit zunehmender Datenmenge wird dies nahezu unmöglich.

Eine andere Vorgehensweise ist, mit Hilfe einer Volltextsuche in den Beschreibungstexten Bilder wiederzufinden. Dieser Weg ist jedoch unpraktikabel. Zum einen bietet eine Volltextsuche keine gute Performance. Andererseits ist es nicht notwendig, vollständige beschreibende Sätze in die Datenbank aufzunehmen. Hierdurch würden viel zu viele Wörter gespeichert, die für einen korrekten Satzaufbau benötigt werden, nach denen aber niemand suchen würde.

Zum Beispiel kommen in dem Satz „Das Bild zeigt ein Lavendelfeld in Frankreich.“ (siehe Abbildung 2-9) nur zwei Wörter vor, die sich auch als Suchwort eignen. Fünf der Wörter sind nur notwendig, um einen vollständigen Satz zu bilden. Wenn man von den speicherplatzrelevanten Buchstaben ausgeht, sind ca. 50% überflüssig (22 Buchstaben in den Wörtern „Lavendelfeld“ und „Frankreich“ gegenüber 23 Buchstaben und Leerzeichen im Rest des Satzes).



Abbildung 2-9: Lavendelfeld in Frankreich

Besser ist es, wenn man einem BLOB stattdessen eine Reihe von Stichwörtern zuordnet, ähnlich wie die Schlagwortzuordnung in Bibliothekskatalogen. Das Problem der Subjektivität wird dadurch zwar nicht gelöst, aber dafür verringert sich der Aufwand für die Dateneingabe und es wird weniger Speicherplatz benötigt.

Zur Aufnahme der Schlagwörter wird der Entitätstyp *Stichwort* eingeführt. Dieser enthält zunächst nur das Attribut *Stichwort\_deutsch* (*Text\_kurz*) und das Schlüsselattribut *StichwortID* (*ID*). Wie der Name des Attributs vermuten läßt, enthält diese Entität zunächst nur Stichwörter in deutscher Sprache, was für Anwendungen im Intranet vorerst ausreichend ist. Falls das System später auch über das Internet zugänglich sein soll, können jederzeit weitere Attribute zugeordnet werden, die z.B. Übersetzungen der deutschen Stichwörter enthalten.

Die Relation *Stichwort\_BLOB* ist eine m:n-Relation.

### 2.2.3 Ersteller

In mittelständischen Unternehmen, in denen sich die potentiellen Benutzer eines BLOB-Archivierungssystems (bei Spinnrad z.B. Mitarbeiter aus den Organisationseinheiten Marketing, Design und Internet-Entwicklung) persönlich kennen, ist es oft so, daß ein

Ersteller	
<u>ErstellerID</u>	N10
Ersteller	VA50

Abbildung 2-10:  
*Ersteller*

Mitarbeiter bei der Suche nach einem Bild nicht nur nach dem Bildinhalt sucht, sondern auch nach dem Ersteller eines Bildes, weil er beispielsweise mit diesem über

ein Motiv gesprochen hat. Außerdem ist es für eventuelle Rückfragen, z.B. nach Hintergrundinformationen zu dem Motiv, hilfreich, den Ersteller zu kennen.

Ähnliches gilt natürlich auch für andere Arten von multimedialen Daten, also für alle BLOBs. Deshalb ist es naheliegend, Informationen über den Ersteller eines BLOBs zu speichern. Dafür wird der Entitätstyp *Ersteller* geschaffen. Dieser enthält ein Textfeld (*Ersteller*), in dem der Name eines Mitarbeiters oder einer externen Agentur eingetragen werden kann und ein Schlüsselattribut (*ErstellerID*). An dieser Stelle ist es sinnvoll, über den Ersteller eine Verknüpfung zu einer Personaldatenbank und zu Geschäftspartnern (Agentur) herzustellen, sobald solche Daten im System vorhanden sind.

*BLOB* und *Ersteller* stehen in einer 1:n-Beziehung zueinander, da ein BLOB in der Regel nur einen Ersteller hat, jeder Ersteller aber mehrere BLOBs erstellen kann.

#### 2.2.4 Bildkategorie

Neben der Zuordnung eines Stichwortes kann man ein Bild auch in verschiedene Kategorien einteilen. Wie in Kapitel 2.2.1 erläutert, sucht ein Designer oder Internet-Entwickler oft nicht nach einem konkreten Bild, sondern nach einer Art oder Kategorie von Bildern. Wird beispielsweise eine Veröffentlichung zum Thema Lavendel erstellt und benötigt der Designer ein geeignetes Bild, dann kann er die Auswahl der Bilder dadurch einschränken, indem er neben dem Stichwort „Lavendel“ auch nach Bildern aus der Kategorie „Landschaftsaufnahmen“ sucht. Ermöglicht wird dies durch die Einführung der Entität *Bildkategorie*, die in einer m:n-Beziehung zu *Bild* steht.

Als Attribute enthält sie *Bildkategorie* (Domain *Text\_kurz*) und *BildkategorieID* (Domain *ID*).

Für den späteren Betrieb des Archivierungssystems ist es wichtig festzulegen, wer diese Bildkategorien nach welchen Kriterien anlegt, da es in den seltensten Fällen in einem Unternehmen Bibliotheks- oder Dokumentationsabteilungen gibt, in denen Personen mit einer entsprechenden Ausbildung im Bereich der Katalogisierung und Kategorisierung arbeiten.

Das konzeptionelle Datenmodell zur Archivierung von BLOBs mit den Ergänzungen für ein geeignetes Information Retrieval sieht wie folgt aus:

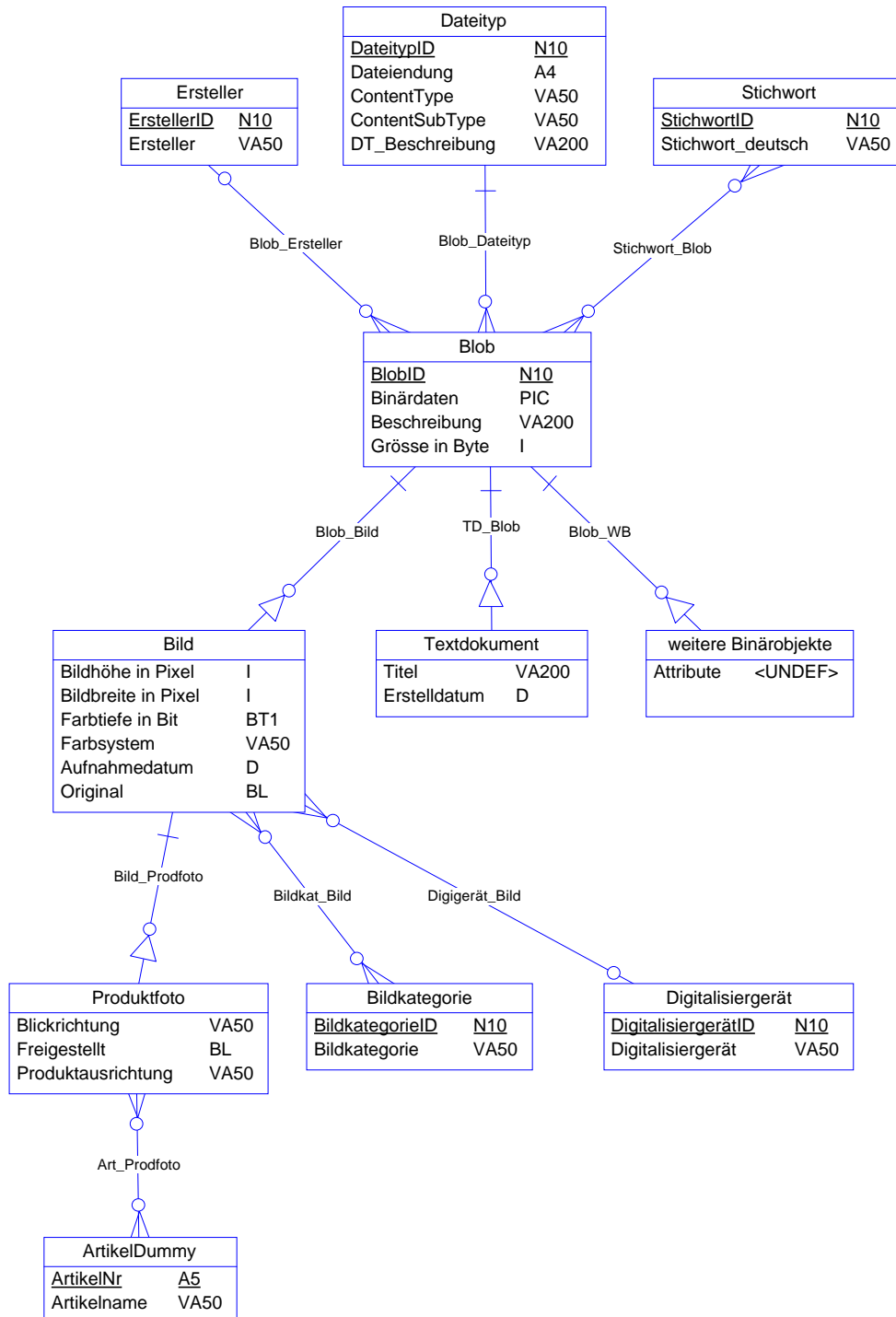


Abbildung 2-11: Konzeptionelles Datenmodell zur Archivierung von BLOBs inklusive der Entitäten für ein geeignetes Information Retrieval.

## 2.3 Generierung des physikalischen Datenmodells

### 2.3.1 Einstellungen und Optionen des Power Designer Data Architect

Die Generierung eines physikalischen Datenmodells aus dem konzeptionellen Datenmodell wird vom Power Designer Data Architect automatisch durchgeführt. Um den Vorgang zu beeinflussen, kann der Benutzer einige Parameter einstellen. In Abbildung 2-12 sieht man das Formular zum Einstellen dieser Parameter.

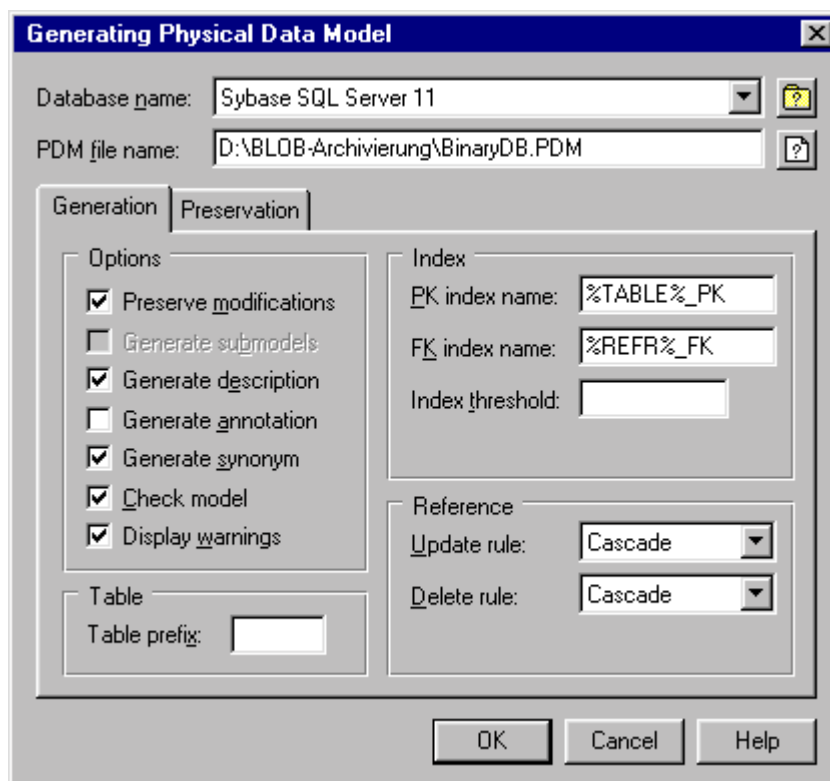


Abbildung 2-12: Parameterformular für die Generierung eines physikalischen Datenmodells

Die Eingabefelder „Database name“ und „PDM file name“ müssen ausgefüllt werden. Alle anderen Angaben sind optional. Im Feld „Database name“ wird das Ziel-DBMS festgelegt. Dieses entscheidet dann zum Beispiel wie die Datentypen umgewandelt werden. Für den ASE wird hier „Sybase SQL Server“ eingestellt. Im Feld „PDM file name“ wird der Dateiname für die Speicherung des physikalischen Datenmodells festgelegt.

Die Einstellungen im Bereich „Options“ wurden wie vom Programm vorgegeben übernommen. Sie wirken sich hauptsächlich auf Beschriftungen und Kommentare aus. Der Parameter „Check model“ bewirkt eine Prüfung des Modells auf Richtigkeit. Wenn „Display warnings“ aktiviert ist, werden nicht nur aufgetretene

Fehler, sondern auch Warnungen angezeigt. Unter dem Punkt „Index“ werden die Namen für Indizes eingestellt. Zu allen Primärschlüsseln und Fremdschlüsseln werden wie in [Syba97d] empfohlen automatisch Indizes erstellt. Als Namen voreingestellt sind „%TABLE%\_PK“ für Primärschlüssel (Primary keys) und „%REFR%\_FK“ für Fremdschlüssel (Foreign keys). Die in Prozentzeichen eingeschlossenen Bezeichner sind Platzhalter. An ihrer Stelle wird bei Primärschlüsseln der Tabellename eingesetzt. Bei Fremdschlüsseln wird für %REFR% der Name der Beziehung eingesetzt.

Im Abschnitt „Reference“ wird festgelegt, ob referentielle Integritätsbedingungen erzeugt werden sollen und welcher Art sie sind. Durch den Eintrag „Cascade“ wird dafür gesorgt, daß die referentielle Integrität hergestellt wird.

### 2.3.2 Umwandlung von Entitätstypen

TABELLE		
SPALTE	int	null

Abbildung 2-13:  
Symbol einer Tabelle

Bei der automatischen Umwandlung des konzeptionellen in ein physikalisches Datenmodell mit den in Kapitel 2.3.1 genannten Einstellungen werden alle Entitätstypen in Tabellen übertragen. Jedes Attribut einer Entität wird als eine Spalte dieser Tabelle übernommen. Für die Namen der Tabellen und Spalten wird im physikalischen Modell der von Sonderzeichen bereinigte Code der Entitäten bzw. Attribute anstelle der Namen verwendet. Die Abbildungen der Tabellen im physikalischen Modell unterscheiden sich nur unwesentlich von denen der Entitäten (siehe Abbildung 2-1). Neben dem Namen der Spalte und dem Datentyp wird zusätzlich angezeigt, ob die Spalte Nullwerte zuläßt (null) oder nicht (not null). Außerdem werden Primärschlüssel unterstrichen dargestellt.

### 2.3.3 Umwandlung von Beziehungstypen

Für die Umwandlung von Beziehungstypen gibt es je nach Art der Relation verschiedene Vorgehensweisen. Existenzabhängige Beziehungen führen dazu, daß der Primärschlüssel der nicht-abhängigen Tabelle auch Primärschlüssel der abhängigen Tabelle wird. Die Tabelle *BILD* erhält somit als Primärschlüssel *BLOBID* aus der Tabelle *BLOB*.

Liegt eine 1:n-Relationen vor, so wird der Primärschlüssel der n-Seite als Fremdschlüssel in die Tabelle der 1-Seite aufgenommen. Beispielsweise enthält die

Tabelle *BLOB* den Fremdschlüssel *DATEITYPID* aus der Tabelle *DATEITYP*. Durch den Eintrag eines Wertes in dieser Spalte kann dem BLOB genau ein Dateityp zugeordnet werden.

M:n-Beziehungen werden in eine eigene Tabelle umgewandelt, die die Primärschlüssel der an der Beziehung beteiligten Tabellen als Schlüssel enthält. Der Tabellename wird aus dem Code des Beziehungstypen generiert. Die Beziehung zwischen *Bild* und *Bildkategorie* (*Bildkat\_Bild*) wird beispielsweise zur Tabelle *BILDKAT\_BILD* mit den beiden Spalten *BILDKATEGORIEID* und *BLOBID*.

#### 2.3.4 Probleme mit dem Datentyp `Serial`

Für künstliche Schlüsselattribute bietet es sich eigentlich an, den Power Designer Datentyp `Serial n` zu verwenden. Dieser Datentyp wird bei der Generierung eines physikalischen Datenmodells für den Sybase Adaptive Server in den Typ `numeric(n,0)` umgewandelt. Zusätzlich erhält die so erzeugte Spalte die Integritätsbedingung `identity`. Dadurch wird der ASE veranlaßt, diese Spalte beim Einfügen eines Datensatzes in die Tabelle automatisch mit einem Wert zu füllen. Der Wert ist jeweils um 1 höher als der zuletzt eingefügte Wert. Jede Tabelle kann allerdings nur eine `identity`-Spalte enthalten. Legt man die Domain *ID* (Kap. 2.1.1) im Powerdesigner mit dem Datentyp `Serial n` an, so erhalten alle Spalten, die dieser Domain zugeordnet sind auch die Eigenschaft `identity`. Treffen nun in einer Tabelle zwei Spalten aufeinander, die der Domain *ID* angehören (z.B. Primärschlüssel *BLOBID* und Fremdschlüssel *DATEITYPID* in der Tabelle *BLOB*), so enthält die Tabelle auch zwei Spalten mit der Eigenschaft `identity`. Dies führt zu einer Fehlermeldung, da der Power Designer an dieser Stelle nicht automatisch erkennt, daß der Fremdschlüssel aus einer anderen Tabelle kommt und nicht die Eigenschaft `identity` übernehmen darf. Deshalb wurde in der Domain *ID* der Datentyp mit `Number(10)` und nicht mit `Serial` festgelegt. Damit die Primärschlüsselspalten im späteren Betrieb des Systems trotzdem automatisch mit Werten gefüllt werden, werden nun im physikalischen Modell alle ID-Spalten auf `identity` geändert.

Das generierte physikalische Datenmodell mit diesen Änderungen wird in Abbildung 2-14 gezeigt.

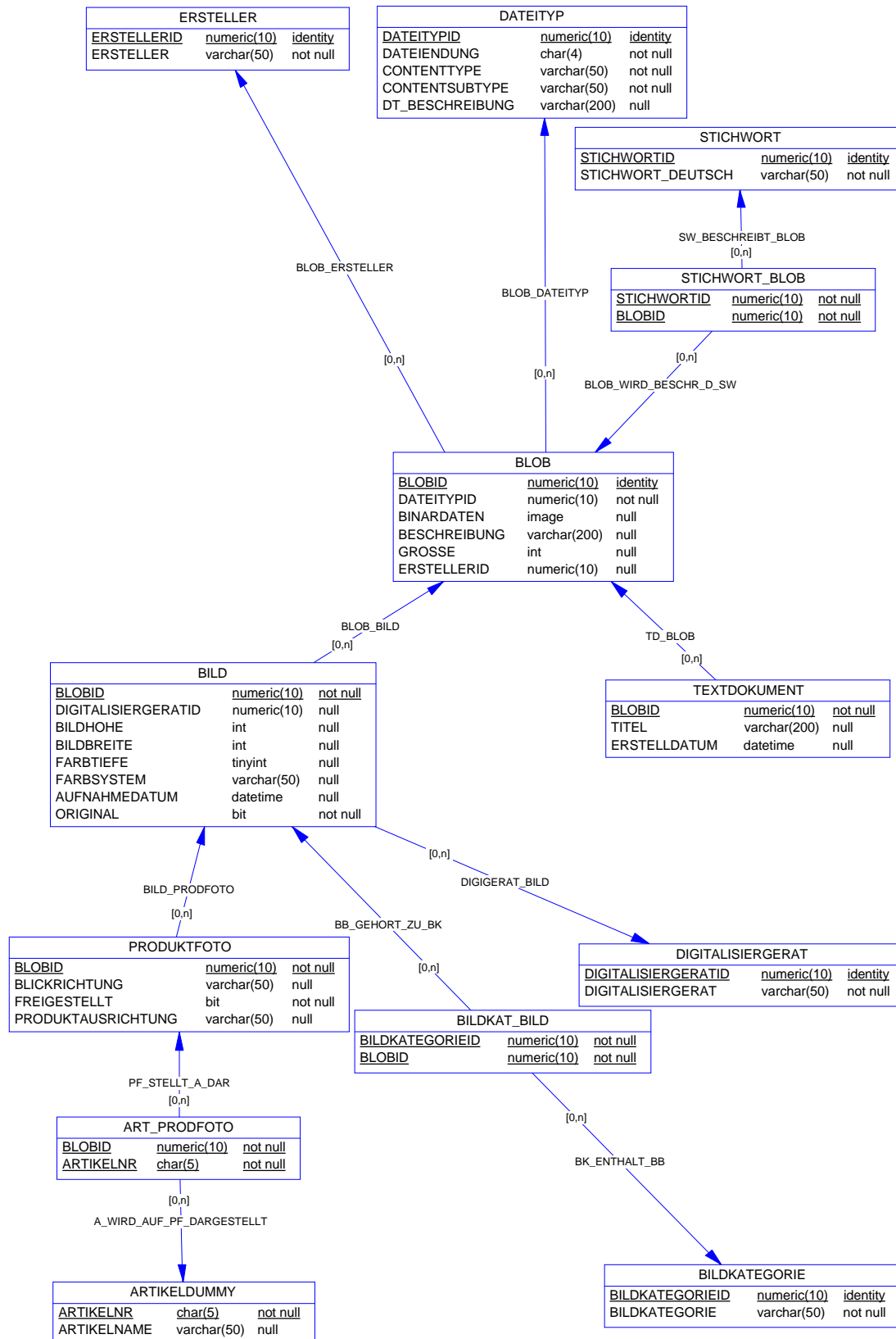


Abbildung 2-14: Physikalisches Datenmodell zur Archivierung von BLOBs

## 2.4 Erstellen der Datenbanktabellen

Um aus dem fertigen physikalischen Datenmodell die entsprechenden Tabellen, Indizes und Integritätsbedingungen zu erstellen, kann der Power Designer eine Reihe von SQL-Statements generieren. Diese können mit dem Tool SQL-Advantage, das per ODBC-Verbindung<sup>8</sup> mit dem Sybase ASE verbunden ist, ausgeführt werden. Der komplette SQL-Code ist dem Anhang B zu entnehmen. Beispielhaft wird hier nur der SQL-Code zur Generierung der Tabelle *BLOB* gezeigt:

```

/* ===== */
/* Table: BLOB */
/* ===== */
create table BLOB
(
  BLOBID          T_ID          identity,
  DATEITYPID     T_ID          not null,
  BINARDATEN     image         null   ,
  BESCHREIBUNG  T_TEXT_LANG   null   ,
  GROSSE        int           null
  constraint CKC_GROSSE_BLOB check (GROSSE >= 0),
  ERSTELLERID   T_ID          null   ,
  constraint PK_BLOB primary key (BLOBID)
)

```

Mit diesem SQL-Befehl wird die Tabelle *BLOB* erstellt (`create table`). Sie enthält die Spalten:

- *BLOBID*, die automatisch mit Werten gefüllt wird (`identity`) - Datentyp *T\_ID*
- *DATEITYPID*, die keine Nullwerte enthalten darf (`not null`) - Datentyp *T\_ID*
- *BINARDATEN* (Name von Sonderzeichen befreit) - Datentyp `image`
- *BESCHREIBUNG*, Datentyp *T\_TEXT\_LANG*
- *GROSSE* (Name von Sonderzeichen befreit) - Datentyp `int`
- *ERSTELLERID* - Datentyp *T\_ID*

Für die Spalte *GROSSE* wird die Integritätsbedingung (`constraint`) `GROSSE>=0` festgelegt. Der Primärschlüssel der Tabelle heißt *PK\_BLOB* und beinhaltet die Spalte *BLOBID*.

<sup>8</sup> Open Database Connectivity ([WWW01a])

### 3 Anbindung der Datenbank an das Intranet

Um über das Internet oder ein Intranet auf eine Datenbank zugreifen zu können, muß der entsprechende Webserver an diese Datenbank angebunden werden. Dazu benötigt man ein Gateway-Programm oder Application Server. Diese Programme können mit dem Webserver entweder über das „Common Gateway Interface“ (CGI, [WWW03a]) oder spezielle „Application Programming Interfaces“ (APIs) Daten austauschen. Standardisierte APIs gibt es unter anderem von Netscape (NSAPI)<sup>9</sup> und Microsoft (ISAPI). Diese APIs können im Gegensatz zu CGI nur genutzt werden, wenn der Webserver die entsprechende Schnittstelle unterstützt. Der Nachteil von CGI ist allerdings eine schlechtere Performance und der hohe Ressourcenverbrauch (vgl. [WWW02b] und [WWW03a]).

Bei der Firma Spinnrad wird der Applikation Server Cold Fusion 3.1 von der Firma Allaire mit der in Abbildung 3-1 dargestellten Konfiguration verwendet. Der Webserver von Netscape und der Cold Fusion Application Server sind zusammen auf einer Maschine installiert. Sie nutzen zur Verständigung das NSAPI. Der Application Server greift über eine ODBC-Verbindung auf das DBMS zu, das auf einem zweiten Rechner installiert ist.

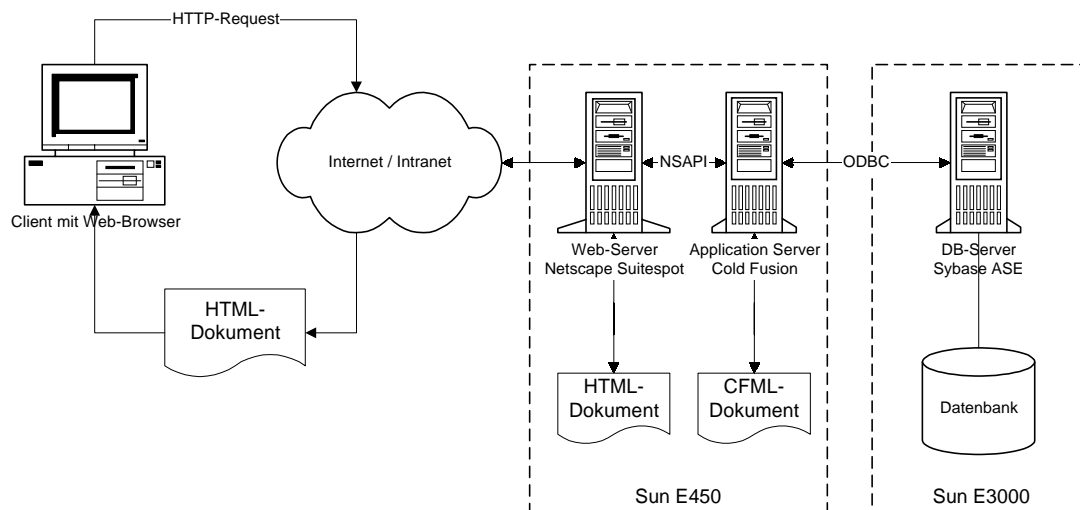


Abbildung 3-1: Serverkonfiguration bei der Firma Spinnrad

Die Datenbankabfragen werden in speziellen Cold Fusion Dokumenten gespeichert. Ein solches Dokument enthält Befehle der Spracherweiterung „Cold Fusion Markup Language“ (CFML), mit der u.a. die SQL-Anfragen an die Datenbank formuliert

werden können, sowie normalen Text und HTML-Befehle. Wird dieses Dokument von einem Client angefordert, so gibt der Webserver die Anforderung an den Application Server weiter. Dieser führt dann den CFML-Code aus und erzeugt mit den Daten, die er von der Datenbank zurück bekommt ein reines HTML-Dokument, das anschließend über den Webserver an den Client gesendet wird.

### **3.1 Aufbau der Spracherweiterung CFML**

Die Cold Fusion Markup Language ist in ihrem Aufbau sehr ähnlich zur Hypertext Markup Language. Die Syntax der Befehle ist genauso wie die HTML-Syntax aufgebaut. CFML-Tags<sup>10</sup> (siehe Kap. 3.3) beginnen immer mit den Buchstaben *CF*. Jedes Tag kann eine bestimmte Anzahl von Parametern enthalten. Außerdem gibt es zu jedem einleitenden CFML-Tag ein abschließendes CFML-Tag, das wie bei HTML-Tags mit einem Schrägstrich (/) beginnt. Zwischen dem einleitenden und abschließenden Tag können dann z.B. SQL-Abfragen formuliert werden oder die Ergebnisse der Datenbankabfrage ausgegeben werden. Innerhalb der CFML-Tags können verschiedene Variablen benutzt werden, die bei der Ausgabe dann durch ihren Wert ersetzt werden. So ist es z.B. möglich, Schleifen über alle Datensätze einer Abfrage zu programmieren und dabei ausgewählte Spalten mittels dieser Variablen auszugeben. Variablennamen werden in die Escape-Zeichen Raute bzw. Doppelkreuz (#) eingeschlossen (siehe Kapitel 3.2).

Ein drittes Sprachelement von CFML stellen die Funktionen dar. Alle verwendeten Funktionen werden in Kap. 3.4 erläutert.

Durch den einfachen und an HTML orientierten Aufbau der Sprache wird einem HTML-Programmierer das Erlernen von CFML erleichtert und ein strukturierter Aufbau von CFML-Dokumenten ermöglicht.

Im weiteren Verlauf dieses Abschnitts werden die in Kapitel 3 benötigten CFML-Tags und der Umgang mit CFML-Variablen erläutert. Als Beispiel dienen die Datenbearbeitungsformulare in Kapitel 3 und im Anhang C.

---

<sup>9</sup> Netscape Server Application Programming Interface ([WWW01b], [WWW02a] und [WWW02b])

<sup>10</sup> Tags sind die in den Escape-Zeichen „<“ und „>“ eingeschlossenen HTML- und CFML-Befehle.

### 3.2 CFML-Variablen

CFML-Variablen können auf sehr unterschiedliche Weise erzeugt werden. Neben den durch eine Abfrage oder mit CFSET erzeugten Variablen, können Daten auch von einer Anwendungsseite an eine andere übergeben werden. Außerdem gibt es verschiedene Variablen, die für die Laufzeit des Application Servers oder einer Verbindung zwischen Client und Webserver gültig sind, sowie Client- und Cookievariablen, in denen Informationen über den Client gespeichert sind.

Variablen, die durch eine SQL-Abfrage erzeugt wurden sind nur innerhalb des CFOUTPUT-Tags, das zu der Abfrage gehört, gültig und enthalten bei jedem Durchlauf jeweils den Wert der entsprechenden Spalte eines Datensatzes. Sie werden über den Namen der Spalte angesprochen (z.B. #Spalte#). Zur eindeutigen Referenzierung einer Abfragevariablen kann als Präfix der Abfrage name angegeben werden. Der Variablenname folgt dann mit einem Punkt vom Präfix getrennt.

Mit CFSET gesetzte Variablen werden durch das Präfix `variables` gekennzeichnet. Sie sind von der Initialisierung mit CFSET an bis zum Ende der Anwendungsseite gültig (siehe Kap. 3.3.3).

Der gebräuchlichste Weg zum Übergeben von Variablen an eine Anwendungsseite sind Formularfelder. Die in ein Formularfeld eingegebenen Daten werden in Variablen mit dem Namen #form.NameDesFormularfeldes# an die Anwendungsseite übergeben, wenn das Formular durch Betätigen des Absende-Buttons (Submit-Button) an den Server übertragen wurde.

URL-Parameter wie in „`http://www.spinnrad.de/seite.cfm?parameter=wert`“ können über die Variablen #url.parameter# abgefragt werden. Diese Variablen können nicht nur aus einem Dokument heraus erzeugt werden, sondern auch vom Benutzer durch Eingabe im Adressfeld des Browsers angegeben werden. Abfrage-, URL- und Form-Variablen sind Nur-Lese-Variablen, d.h. ihre Werte können innerhalb einer Anwendungsseite nicht geändert werden.

Weitere Arten von Variablen werden in [Alla98a] erläutert. Da sie zur Umsetzung des BLOB-Archivierungssystems nicht verwendet wurden, wird auf eine genauere Erläuterung an dieser Stelle verzichtet.

Wird kein Gültigkeitsbereich mit dem Präfix festgelegt, versucht Cold Fusion, die Variablen zunächst als Abfragevariable auszuwerten. Anschließend wird versucht, sie als normale Variable, URL-Variable und schließlich als Formularvariable auszuwerten. Um die Verarbeitungsgeschwindigkeit zu erhöhen, sollte außer bei Abfragevariablen, immer das Präfix verwendet werden.

Eine CFML-Variable kann ein String-Wert, Zahlenwert, boolescher Wert oder Datums-/Uhrzeitwert sein. Außerdem gibt es die zusammengesetzten Datentypen Array und Liste. Die Typkonvertierung zwischen den einfachen Datentypen übernimmt der Cold Fusion Server selbständig bei der Auswertung eines Ausdrucks. Für den Zugriff auf Elemente einer Liste oder eines Arrays gibt es spezielle Funktionen. Die Arbeitsweise der verwendeten Funktionen wird in Kap. 3.4 erläutert.

Cold Fusion-Variablen können nur innerhalb der eigentlichen Tagdefinition zwischen „<CFTAG“ und „>“ sowie innerhalb des CFOUTPUT-Tags zwischen <CFOUTPUT . . . > und </CFOUTPUT> verwendet werden. Soll in diesem Bereich die Raute („#“) verwendet werden, so muß sie zweimal eingegeben werden („##“), damit sie vom Cold Fusion Server nicht als einleitendes Symbol für eine Variable angesehen wird.

### 3.3 Cold Fusion Tags

#### 3.3.1 Datenbankabfrage mit <CFQUERY>

Das CFQUERY-Tag übergibt beliebige SQL-Anweisungen per ODBC an eine Datenbank. Die Syntax ist wie folgt aufgebaut:

```
<CFQUERY
  NAME="Abfragename"
  DATASOURCE="DSN"
  USERNAME="Benutzername"
  PASSWORD="Passwort"
  MAXROWS="Zahl"
  TIMEOUT="ms"
  DEBUG>
SQL-Anweisungen
</CFQUERY>
```

Für eine Abfrage müssen die Attribute NAME und DATASOURCE angegeben werden. Alle anderen Attribute sind optional. Mit NAME wird der Abfrage zur späteren Referenzierung ein eindeutiger Bezeichner zugewiesen. DATASOURCE gibt die ODBC-Datenquelle an, aus der die Daten bezogen werden sollen. Die ODBC-Datenquelle wird z.B. in Windows 95 mit dem ODBC-Administrator festgelegt (Abbildung 3-2). Im Administrationstool für den Cold Fusion Application Server kann man dann der Datenquelle zusätzlich einen Login-Namen und ein Paßwort geben, mit dem sich der Application Server in die Datenbank einloggt (Abbildung 3-3). Werden diese Angaben nicht gemacht, so müssen die Attribute USERNAME und PASSWORD im CFQUERY-Tag angegeben werden, falls die entsprechende Datenbank diese zum Einloggen benötigt.



Abbildung 3-2: Einstellungen für die BLOB-Datenbank im ODBC-Administrator ...

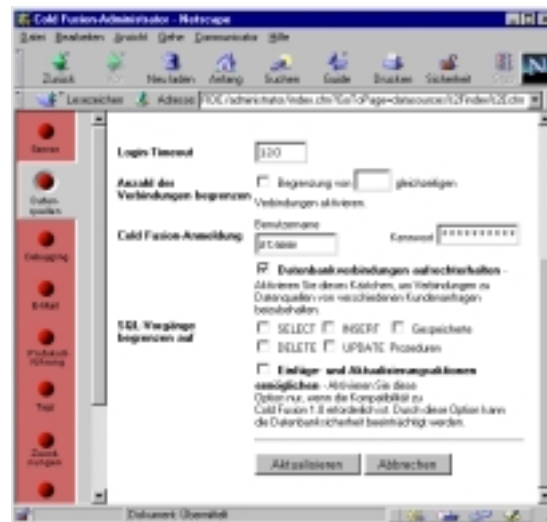


Abbildung 3-3: ... und im Cold Fusion Administrator

Mit dem Parameter MAXROWS kann die maximale Anzahl der Datensätze, die in der Ergebnisgruppe ausgegeben werden sollen, beschränkt werden. Mit TIMEOUT kann eine maximale Zeitdauer in Millisekunden festgelegt werden, nach der die ODBC-Verbindung mit einer Fehlermeldung abgebrochen wird. Allerdings unterstützen nicht alle ODBC-Treiber diesen Parameter.

Durch die Angabe des Wortes DEBUG werden neben der eigentlichen Ausgabe auch Debug-Informationen angezeigt. Sie beinhalten die tatsächlich ausgeführte SQL-Abfrage, die davon betroffenen Datensätze und die für die Abfrage benötigte Zeit.

Die Variable `Abfragename.RecordCount` enthält nach der Durchführung der Abfrage die Anzahl der Datensätze, die in der Ergebnismenge enthalten sind.

### 3.3.2 Ausgabe von Daten mit `<CFOUTPUT>`

Syntax:

```
<CFOUTPUT
    QUERY="Abfragename"
    MAXROWS="maximale Zeilenausgabe"
    GROUP="Parameter"
    STARTROW="Startzeile">
CFML-Tags, Cold Fusion-Variablen, HTML-Tags und/oder Text
</CFOUTPUT>
```

Das `CFOUTPUT`-Tag dient zur Ausgabe von CFML-Variablen und von Abfrageergebnissen. Innerhalb dieses Tags können CFML-Variablen verwendet werden. Wird der Parameter `QUERY` angegeben, so wird der Code innerhalb des Tags für jeden Datensatz, der in der Ergebnismenge der Abfrage vorkommt, genau einmal ausgeführt. Mit dem Parameter `MAXROWS` kann die Ausgabe auf eine maximale Anzahl von Datensätzen beschränkt werden. `STARTROW` legt die Nummer des ersten Datensatzes fest. Mit diesen beiden Attributen ist es z.B. möglich, große Tabellen so aufzuteilen, daß immer nur eine bestimmte Anzahl Datensätze angezeigt werden und man mittels eines Buttons oder Links die nächsten Datensätze auswählt.

Mit dem Parameter `GROUP` können die Datensätze ähnlich wie mit dem `GROUP BY`-Statement in SQL gruppiert werden. Wird dieser Parameter verwendet, ist es möglich das `CFOUTPUT`-Tag zu verschachteln, was ansonsten nicht erlaubt ist.

### 3.3.3 Setzen von Variablen mit `<CFSET>`

Mit `<CFSET>` können benutzerdefinierte Variablen erzeugt und mit Werten gesetzt werden. Die Syntax lautet: `<CFSET VarName="Wert">`. Falls die Variable `VarName` noch nicht existiert, wird sie neu angelegt. Ansonsten wird der bestehende Inhalt mit `Wert` überschrieben. Eine mit dem `CFSET`-Tag gesetzte Variable ist nur innerhalb einer Anwendungsseite gültig. Soll sie an eine weitere Anwendungsseite übergeben werden, so kann dies mit einem Formularfeld oder einem URL-Parameter geschehen.

### 3.3.4 Einfügen zusätzlicher Dateien in den Code mit <CFINCLUDE>

Das CFINCLUDE-Tag ermöglicht es dem Programmierer, in eine Datei weiteren Code zu importieren. Mit dem einzigen Parameter `TEMPLATE="Datei.cfm"` wird eine Datei angegeben, die an der Stelle des CFINCLUDE-Tags eingefügt wird. Dadurch können in Abhängigkeit von bestimmten Variablen unterschiedliche HTML-Anweisungen in den Code eingefügt werden, um so z.B. das Aussehen zu beeinflussen, oder zusätzliche Gestaltungselemente in eine Webseite zu integrieren.

### 3.3.5 Umleitung der Ausgabe mit <CFLOCATION>

Mit dem CFLOCATION-Tag kann von der gerade bearbeiteten Anwendungsseite aus eine andere CFML- oder HTML-Seite geöffnet und an den Client übertragen werden. Dadurch ist es möglich, nach der Abarbeitung von CFML-Befehlen in einer Anwendungsseite, eine andere Seite aufzurufen, die dann z.B. ein Hauptmenü oder eine Übersicht anzeigt. Mit dem Parameter `URL="url"` wird die zu öffnende Seite festgelegt.

### 3.3.6 Schleifenprogrammierung mit <CFLOOP>

In der Cold Fusion Markup Language werden fünf verschiedene Arten von Schleifen bereitgestellt. Alle Schleifen werden mit dem CFLOOP-Tag definiert. Je nach Art der Schleife werden unterschiedliche Parameter benötigt, die die Anzahl der Schleifendurchläufe oder die Abbruchbedingung bestimmen.

Neben der für das BLOB-Archivierungssystem verwendeten Schleife über eine Liste gibt es noch Indexschleifen (FOR-Schleifen), Bedingungsschleifen (WHILE-Schleifen), Schleifen über eine Abfrage und Schleifen über eine COM-Sammlung (siehe dazu [Alla97a]).

Das Ausführen einer Schleife über eine Liste ermöglicht es, jeden Wert in der Liste genau einmal zu durchlaufen. Dabei enthält die mit dem Parameter `INDEX="Indexname"` festgelegte Variable jeweils einen Wert aus der Liste. Die zugrundeliegende Liste wird mit dem Parameter `LIST="Liste"` festgelegt. `DELIMITERS="Trennzeichen"` gibt an, welche Zeichen als Trennzeichen in der Liste interpretiert werden sollen.

### 3.3.7 Bedingte Verzweigung mit <CFIF>, <CFELSEIF> und <CFELSE>

Mit Hilfe von CFIF-, CFELSE- und CFELSEIF-Tags kann die Bearbeitung einer Anwendungsseite durch Bedingungen gesteuert werden. Die Funktionalität entspricht der von IF-Anweisungen in konventionellen Programmiersprachen.

Syntax:

```
<CFIF Ausdruck1>
    HTML- und CFML-Tags
<CFELSEIF Ausdruck2>
    HTML- und CFML-Tags
<CFELSE>
    HTML- und CFML-Tags
</CFIF>
```

Die CFELSEIF und CFELSE-Anweisungen können auch weggelassen werden, sie sind optional. Außerdem kann CFELSE auch direkt auf CFIF folgen. Die Ausdrücke im CFIF- und CFELSEIF-Tag werden zu einem booleschen Wert ausgewertet und die Verzweigung erfolgt dann entsprechend.

Zur Ermittlung des booleschen Wertes von Ausdruck1 und Ausdruck2 sind unter anderem folgende Vergleichsoperatoren zugelassen:

Operator	Vergleich
IS	Gleichheit
IS NOT	Ungleichheit
GREATER THAN	größer als
LESS THAN	kleiner als

Außerdem können Vergleiche mit AND, OR, XOR, EQV (Äquivalenz) und IMP (Implikation) verknüpft werden. Die Auswertungsreihenfolge kann durch Klammerung mit runden Klammern - „(“ und „)“ - beeinflusst werden.

### 3.3.8 Dateioperationen mit <CFFILE>

Mit dem CFFILE-Tag können Interaktionen mit Dateien durchgeführt werden. Der Parameter ACTION legt dabei die Aktion fest, die mit einer Datei durchgeführt werden soll. Weitere Parameter sind dann von dem Wert des Parameters ACTION

abhängig. Mit CFFILE können Dateien verschoben, umbenannt, kopiert, gelöscht, gelesen, geschrieben und Inhalt angehängt werden. Außerdem gibt es die Möglichkeit, eine Datei vom Client auf den Server hochzuladen.

Für die Datenverarbeitungsformulare und -applikationen in Kapitel 4 wird das Hochladen (upload) und Löschen (delete) von Dateien benötigt.

Die Syntax des CFFILE-Tags zum Hochladen einer Datei:

```
<CFFILE ACTION="upload"
  FILEFIELD="Formularfeld"
  DESTINATION="Verzeichnis"
  NAMECONFLICT="Verhalten"
  ACCEPT="Dateinamenerweiterungen"
  MODE="Berechtigung"
  ATTRIBUTES="Dateiattribute">
```

Der Parameter FILEFIELD ist erforderlich und gibt den Namen eines Formularfeldes an, das zum Auswählen einer Datei verwendet wurde. Das entsprechende Formularfeld muß mit dem HTML-Tag `<input type="file" name="Formularfeld">` erzeugt worden sein.

Mit DESTINATION wird das Verzeichnis oder der Dateiname der neu zu erstellenden Datei festgelegt. Wird nur ein Verzeichnis angegeben, behält die Datei ihren bisherigen Namen bei. Existiert in dem angegebenen Verzeichnis bereits eine Datei mit dem selben Namen, entscheidet der Parameter NAMECONFLICT darüber, wie verfahren wird. Wird NAMECONFLICT auf „Error“ gesetzt, wird die Verarbeitung der Anwendungsseite abgebrochen und eine Fehlermeldung ausgegeben. Mit „Skip“ wird die Datei weder gespeichert noch ein Fehler ausgegeben. Dies dient dazu, ein selbstdefiniertes Verhalten zu programmieren, indem nach dem CFFILE-Tag Variablen mit dem Präfix „file“ abgefragt werden. Durch die Angabe von NAMECONFLICT="Overwrite" kann das Überschreiben der bestehenden Datei erzwungen werden und durch Makeunique erzeugt der Cold Fusion Application Server einen eindeutigen Dateinamen um den Konflikt zu beheben.

Der Parameter `ACCEPT` dient der Einschränkung der akzeptierten Dateitypen. Die Dateitypen werden durch ihren Typ und Subtyp festgelegt und als kommagetrennte Liste angegeben (z.B. `ACCEPT="image/gif,application/msword"`).

Mit den Parametern `MODE` bzw. `ATTRIBUTES` können die Berechtigungen (Application Server unter Solaris) bzw. die Dateiattribute (Application Server unter Windows 95/NT) für die hochgeladene Datei festgelegt werden.

Nach dem Datei-Upload stehen unter anderem folgende Variablen zur Verfügung:

Variable	Beschreibung
<code>file.ServerDirectory</code>	Verzeichnis der Datei, die auf dem Server gespeichert wurde
<code>file.ServerFile</code>	Dateiname der Datei, die auf dem Server gespeichert wurde
<code>file.ServerFileExt</code>	Erweiterung der Datei, die auf dem Server gespeichert wurde
<code>file.ContentType</code>	MIME-Inhaltstyp (Contenttyp) der gespeicherten Datei
<code>file.ContentSubType</code>	MIME-Inhaltssubtyp (Contentsubtype) der gespeicherten Datei
<code>file.FileSize</code>	Größe der auf dem Server gespeicherten Datei in Byte

Um eine Datei vom Server zu löschen wird das `CFFILE`-Tag mit dem Parameter `ACTION="Delete"` verwendet. Der zweite Parameter `FILE="Dateiname"` beinhaltet den Verzeichnispfad und Namen der zu löschenden Datei.

### 3.4 Funktionen in CFML

Die Cold Fusion Markup Language stellt in der Version 3.1 über 170 verschiedene Funktionen zur Verfügung. Im Folgenden werden nur die für das BLOB-Archivierungssystem verwendeten Funktionen vorgestellt.

#### 3.4.1 `ParameterExists(Parameter)`

- **Parameter:** Name der zu überprüfenden Variable
- **Rückgabewert:** Boolean; `TRUE`, wenn die Variable `Parameter` bereits initialisiert wurde; `FALSE`, wenn die Variable nicht existiert

### 3.4.2 Chr(ASCII-Wert)

- ASCII-Wert: Zahl zwischen 0 und 255, für die das entsprechende ASCII-Zeichen zurückgegeben werden soll.
- Rückgabewert: Buchstabe mit dem entsprechenden ASCII-Wert

### 3.4.3 LCase(String)

- String: beliebiger Text
- Rückgabewert: Der mit String festgelegte Text, in dem alle Großbuchstaben in Kleinbuchstaben umgewandelt wurden.

### 3.4.4 Left(String,n)

- String: beliebiger Text
- n: Anzahl der Zeichen

Rückgabewert: Die ersten n Zeichen des mit String angegebenen Textes

### 3.4.5 Len(String)

- String: beliebiger Text

Rückgabewert: Die Länge des mit String angegebenen Textes.

### 3.4.6 PreserveSingleQuotes(String)

- String: beliebiger Text
- Rückgabewert: String

Wird String ohne diese Funktion verwendet, werden aus String eventuell vorhandene Anführungszeichen entfernt, da sie sonst zu unerwarteten Ausgaben führen könnten. PreserveSingleQuotes() verhindert, daß Cold Fusion Anführungszeichen automatisch löscht.

### 3.4.7 ListContains(Liste,String,Trennzeichen)

- Liste: Liste, in der gesucht wird
- String: Text, der in der Liste gesucht wird
- Trennzeichen: Das Trennzeichen, das die verschiedenen Listenelemente voneinander trennt.

Rückgabewert: Index des ersten Elements von `Liste`, das den mit `String` angegebenen Text enthält. Bei dem Vergleich der Texte wird Groß- und Kleinschreibung beachtet.

#### 3.4.8 `ListContainsNoCase(Liste, String, Trennzeichen)`

- `Liste`: Liste, in der gesucht wird
- `String`: Text, der in der Liste gesucht wird
- `Trennzeichen`: Das Trennzeichen, das die verschiedenen Listenelemente voneinander trennt.

Rückgabewert: Index des ersten Elements von `Liste`, das den mit `String` angegebenen Text enthält. Bei dem Vergleich der Texte wird Groß- und Kleinschreibung **nicht** beachtet.

### 3.5 **Selbstdefinierte CFML-Tags**

Die Entwicklung selbstdefinierter Tags mit CFML bietet die Möglichkeit, einen besser strukturierten und lesbaren Code zu entwickeln. Außerdem kann so die Wiederverwendbarkeit des Codes erhöht werden.

Selbstdefinierte CFML-Tags bestehen wie normale Anwendungsseiten aus CFML-Tags, HTML-Tags und normalem Text und werden mit

```
<CF_TagName Parameter1="Wert1" Parameter2="Wert2" ...>
```

aufgerufen. Das Tag muß unter dem Namen `TagName.cfm` gespeichert werden. Lokale Variablen innerhalb des Tags können die gleichen Namen wie die einer aufrufenden Anwendungsseite verwenden, da für die Speicherung unterschiedliche Speicherbereiche verwendet werden. Sollen die Variablen der aufrufenden Seite explizit angesprochen werden, muß das Präfix `Caller` verwendet werden (z.B. `#Caller.VarName#`). Die an ein selbstdefiniertes Tag übergebenen Parameter werden mit dem Präfix `Attributes` angesprochen (z.B. `#Attributes.Parameter1#`).

Neben den in CFML und HTML programmierten Tags, gibt es zusätzlich die Möglichkeit, mit Hilfe einer Cold Fusion API erweiterte Cold Fusion-Tags in C++ zu schreiben. Die kompilierten Programmbibliotheken müssen vor ihrer Verwendung

mit Hilfe des Cold Fusion Administrations-Tools registriert werden. Erweiterte Cold Fusion-Tags werden mit

```
<CFX_TagName Param1="Wert1" Param2="Wert2" ...>
```

aufgerufen. In der Allaire-Taggallery ([WWW04a]) stehen eine Vielzahl von Tags zum Download zur Verfügung. Für das BLOB-Archivierungssystem wurden von dort die drei Tags <CFX\_PUTIMAGE>, <CFX\_GETIMAGE> und <CFX\_IMGINFO> heruntergeladen und eingesetzt (Kap. 3.5.2 und 3.5.3).

### 3.5.1 Kopf und Fuß einer HTML-Seite festlegen mit <CF\_HEAD> und <CF\_FOOT>

Um für alle Applikationen ein einheitliches Aussehen zu schaffen, wurden die beiden selbstdefinierten Tags <CF\_HEAD Title="Titeltext"> und <CF\_FOOT> definiert. Das CF\_HEAD-Tag gibt die für eine HTML-Seite nötigen einleitenden Tags (HEAD, TITLE und BODY, [Münz96a]) aus. Dabei wird der mit dem Parameter Title übergebene Text in die Überschrift für die Seite übernommen. Außerdem wird im oberen Bereich der Seite das Firmenlogo und eine Überschrift angezeigt.

#### HEAD.CFM

```
<head>
<title>
    Spinnrad-Intranet: <cfoutput>#Attributes.title#</cfoutput>
</title>
</head>
<body TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#990000" VLINK="#006600"
    ALINK="#FF0000">
<table cellspacing="2" cellpadding="2" width="600" border="0">
<tr>
    <td align="left" width="25%">
        
    </td>
    <td align="center" width="50%">
        <b>Diplomarbeit:</b> Binäre Datenobjekte
    </td>
    <td align="right" width="25%">
        Markus Stamm<br>Juli-September 1998
    </td>
</tr>
</table>
<hr align="CENTER" color="Green" size="2">
```

Analog zum CF\_HEAD-Tag wurde ein CF\_FOOT-Tag angelegt, daß neben den seitenabschließenden HTML-Tags (/BODY, /HTML) einen Navigationsbereich ausgibt, der es ermöglicht, von jeder Seite aus auf die verschiedenen Datenbearbeitungsformulare zu wechseln. Parameter werden an dieses Tag nicht übergeben.

#### FOOT.CFM

```
<p><p>
<hr align="CENTER" size="2" color="Green">
<table border="0" cellpadding="2" cellspacing="2">
<tr>
  <td align="center">
    [<a href="Suche_frm.cfm">Suche</a>]
    [<a href="Start.cfm">Start-Seite</a>]
  </td>
</tr>
<tr><td align="center"><br></td></tr>
<tr>
  <td align="center">
    [<a href="Bild_frm.cfm">Bild</a>]
    [<a href="Text_frm.cfm">Text</a>]
    [<a href="Blob_frm.cfm">BLOB</a>]
    [<a href="Ersteller_frm.cfm">Ersteller</a>]
  </td>
</tr>
<tr>
  <td align="center">
    [<a href="Digitalisiergeraet_frm.cfm">Digitalisiergerät</a>]
    [<a href="Stichwort_frm.cfm">Stichwort</a>]
    [<a href="Bildkategorie_frm.cfm">Bildkategorie</a>]
    [<a href="Dateityp_frm.cfm">Dateityp</a>]
    [<a href="Artikel_frm.cfm?RequestTimeout=120">Artikel</a>]
  </td>
</tr>
</table>
</body>
</html>
```

### 3.5.2 Datenbankabfragen mit BLOBs (CFX\_PUTIMAGE/CFX\_GETIMAGE)

Die beiden Tags <CFX\_PUTIMAGE> und <CFX\_GETIMAGE><sup>11</sup> aus der Cold Fusion-Taggallery ermöglichen das Lesen und Schreiben von BLOBs aus einer und in eine Datenbank.

Die Syntax für das CFX\_PUTIMAGE-Tag, mit dem binäre Daten in eine Datenbank geschrieben werden können, lautet wie folgt:

```
<CFX_PUTIMAGE
  DATASOURCE="ODBC-Datenquelle"
  USER="Benutzer"
  PASSWORD="Passwort"
  SQL="SQL-Abfrage"
  INPUT="Dateiname">
```

Die im Parameter SQL="SQL-Abfrage" enthaltene Abfrage könnte z.B. so lauten:

```
UPDATE tabelle SET binaer_spalte = ? WHERE ...
```

oder

```
INSERT INTO tabelle VALUES ('einWert',?, 'andererWert')
```

Das Fragezeichen legt dabei fest, an welcher Stelle die binären Daten eingefügt werden sollen. Die Daten werden aus der im Parameter INPUT="Dateiname" angegebenen Datei gelesen.

Die Parameter DATASOURCE, USER und PASSWORT bestimmen die ODBC-Datenquelle sowie einen Login-Namen und -Passwort für die entsprechende Datenbank.

Zum Auslesen der Daten aus der Datenbank und Abspeichern in eine Datei wird das CFX\_GETIMAGE-Tag mit folgender Syntax verwendet:

```
<CFX_GETIMAGE
  DATASOURCE="ODBC-Datenquelle"
  USER="Benutzer"
```

---

<sup>11</sup> Während der Entwicklung des BLOB-Archivierungssystems lagen die beiden Tags CFX\_GETIMAGE und CFX\_PUTIMAGE nur in einer Version für Windows 95/NT vor. Nach einer entsprechenden Anfrage an den Entwickler dieser Tags, Jukka Manner aus Finnland, hat dieser den kompletten Quellcode zur Verfügung gestellt. Eine entsprechende Portierung auf das Sun Solaris Betriebssystem wäre damit möglich, würde allerdings den Rahmen dieser Arbeit sprengen.

```
PASSWORD="Passwort "  
SQL="SQL-Abfrage "  
OUTPUT="Dateiname ">
```

Die SQL-Abfrage könnte z.B.

```
SELECT spalte FROM tabelle WHERE ...
```

lauten. Dabei muß beachtet werden, daß die Abfrage nur eine Spalte zurückgeben darf. Das Datenfeld in der ersten Zeile dieser Spalte wird dann in der Datei abgespeichert, die mit OUTPUT="Dateiname" angegeben wurde.

Die anderen Parameter haben die gleiche Funktion wie beim CFX\_PUTIMAGE-Tag.

### 3.5.3 Bildinformationen mit <CFX\_IMGINFO> abfragen

Das CFX\_IMGINFO-Tag<sup>12</sup> liefert Informationen über Bilddateien im JPEG- oder GIF-Format.

Syntax:

```
<CFX_IMGINFO FILENAME="Dateiname" NAME="Name">
```

Mit dem Parameter NAME="Name" wird das Präfix für die Variablen festgelegt, die nach dem Aufruf von <CFX\_IMGINFO> zur Verfügung stehen. Handelt es sich bei der im Parameter FILENAME="Dateiname" angegebenen Datei um ein Bild im JPEG- oder GIF-Format, so enthält die Variable Name.RecordCount den Wert 1. Ansonsten enthält sie den Wert 0. Außerdem stehen folgende Variablen zur Verfügung:

Name.Type	Typ der Bilddatei als String: „GIF“ oder „JPEG“.
Name.Height	Höhe der Bilddatei in Bildpunkten/Pixeln
Name.Width	Breite der Bilddatei in Bildpunkten/Pixeln
Name.Compression	Art der Bildkompression im Klartext (z.B. „Baseline“ oder „GIF89a“)

---

<sup>12</sup> Das CFX\_IMGINFO-Tag liegt ebenfalls nur in einer Windows 95/NT-Version vor. Da mit dem Entwickler kein Kontakt aufgenommen werden konnte, muß mit einer Portierung abgewartet werden, bis dieses oder ein vergleichbares Tag für Sun Solaris in der Cold Fusion-Taggallery ([WWW04a]) zur Verfügung gestellt wird.



## 4 Datenbearbeitungsformulare und Anwendungsseiten

Ein vollständiges System zur Bearbeitung von Daten in einer Datenbank benötigt für jede als eigene Tabelle vorliegende Entität vier verschiedene Aktionen: Einfügen, Ändern, Löschen und Ausgeben. Außerdem muß es möglich sein, jede Relationen zwischen den Entitäten herstellen, löschen und ansehen zu können. Bei der Erstellung der Datenbearbeitungsformulare und -applikationen wurde darauf geachtet, alle oben genannten Aktionen umzusetzen. Dabei sind die in der folgenden Tabelle aufgeführten Cold Fusion Dokumente entstanden. Die Spalte *Entität* enthält den Namen der Entität zu der das entsprechende Formular gehört. Die Spalte *Relationen* listet alle Entitäten auf, die in einer Beziehung zu der Entität stehen und in dem Formular angezeigt werden bzw. dort zu ändern sind. Die Spalte *Aktionen* gibt an, welche Tätigkeit mit den Attributen der entsprechenden Entität und mit den Relationen durchgeführt werden können.

Dokument	Entität	Relationen	Aktion
Blob_frm.cfm	BLOB	Dateityp, Ersteller, Stichwort	Anzeigen (ohne Binärdaten), Löschen
Blob_view.cfm	BLOB		Anzeige der eigentlichen Binärdaten mit dem jeweils zugehörigen Programm
Blob_insert_frm.cfm	BLOB	Ersteller, Stichwort	Einfügen
Blob_update_frm.cfm	BLOB	Ersteller, Stichwort	Ändern
Dateityp_frm.cfm	Dateityp	BLOB	Anzeigen
Dateityp_update_frm.cfm	Dateityp		Ändern
Bild_frm.cfm	Bild	BLOB, Bildkategorie, Dateityp, Ersteller, Digitalisiergerät, Produktfoto, Artikel	Anzeigen, Löschen
Bild_insert_frm.cfm	Bild	BLOB, Bildkategorie, Ersteller, Digitalisiergerät, Stichwort	Einfügen
Bild_update_frm.cfm	Bild	BLOB, Bildkategorie, Ersteller, Digitalisiergerät, Stichwort, Dateityp (nur Anzeige)	Ändern
Ersteller_frm.cfm	Ersteller	BLOB	Anzeigen, Löschen
Ersteller_update_frm.cfm	Ersteller		Ändern
Digitalisiergeraet_frm.cfm	Digitalisiergerät	Bild	Anzeigen, Löschen

Dokument	Entität	Relationen	Aktion
Digitalisiergeraet_update_frm.cfm	Digitalisier- gerät		Ändern
Stichwort_frm.cfm	Stichwort	BLOB	Anzeigen, Löschen
Stichwort_update_frm.cfm	Stichwort		Ändern
Bildkategorie_frm.cfm	Bild- kategorie	Bild	Anzeigen, Löschen
Bildkategorie_update_frm.cfm	Bild- kategorie		Ändern
Artikel_frm.cfm	Artikel- dummy	Produktfoto	Anzeigen, Löschen
Artikel_update_frm.cfm	Artikel- dummy		Ändern

Das Löschen von Datensätzen in der Tabelle *Dateityp* kann nicht direkt vom Benutzer ausgelöst werden, da ansonsten die referentielle Integrität verletzt wird. Die Datensätze werden automatisch gelöscht, wenn das letzte BLOB mit dem entsprechenden Dateityp gelöscht wird (Kap. 4.5).

Entitäten, die keine eigenen Einfüge-Formulare haben, werden über die Formulare der Entitäten eingefügt, zu denen sie in einer Beziehung stehen.

Für eine effektive Nutzung als BLOB-Archivierungssystem ist es darüber hinaus wichtig, mit Hilfe der dafür vorgesehenen Daten die verschiedenen BLOBs (also Bilder, Texte und andere) in der Datenbank wieder aufzufinden. Dazu wird in Kap. 4.6 das entsprechende Suchformular vorgestellt.

In Abbildung 4-1 wird das Übersicht-Menü für das BLOB-Archivierungssystem gezeigt, wie es im Netscape Navigator 4.x dargestellt wird. Es enthält Links zu neun verschiedenen Datenbearbeitungsformularen und zu dem Suchformular. Das Formular für die Produktfotos kann nur von dem Formular für die Bilder ausgewählt werden, da ein Produktfoto eine Sonderform eines Bildes ist und nur im Zusammenhang damit Änderungen vorgenommen werden können.

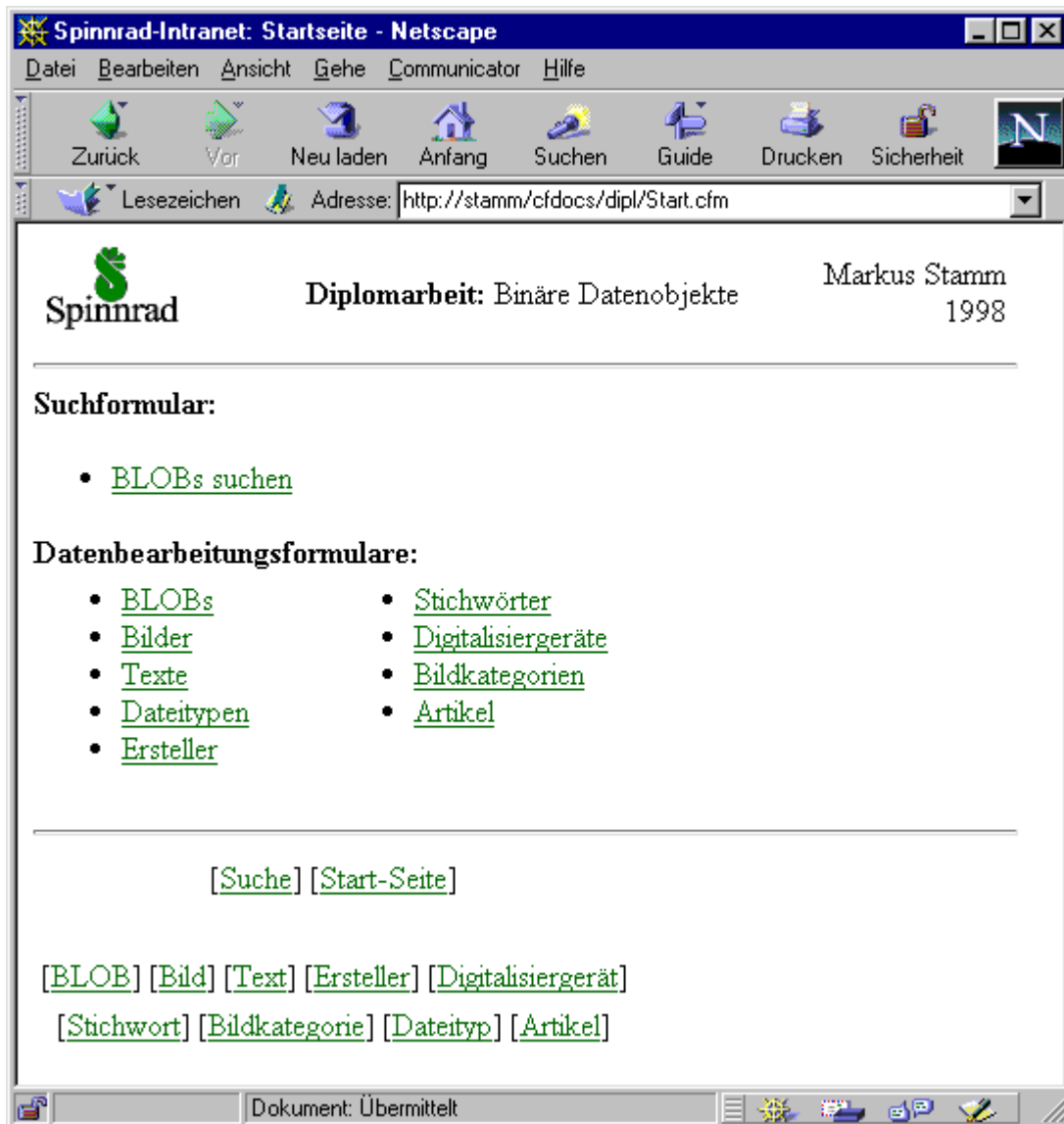


Abbildung 4-1: Übersicht-Menü des BLOB-Archivierungssystems

Hinter jedem Link verbirgt sich das jeweilige Formular zum Anzeigen der entsprechenden Entität wie in der Tabelle oben aufgeführt. Innerhalb der Anzeigeformulare (Kap. 4.1) gibt es wiederum Links zum Aufruf von Einfügeformularen (Kap.4.3), Änderungsformularen (Kap. 4.4) und zum Löschen eines Datensatzes (Kap. 4.5).

### 4.1 Datenanzeigeformulare

In Abbildung 4-2 wird das Datenanzeigeformular für die Entität *Stichwort* gezeigt. In der ersten Spalte wird die ID des jeweiligen Stichwortes angezeigt. Die zweite Spalte enthält die Spalte *STICHWORT\_DEUTSCH*.

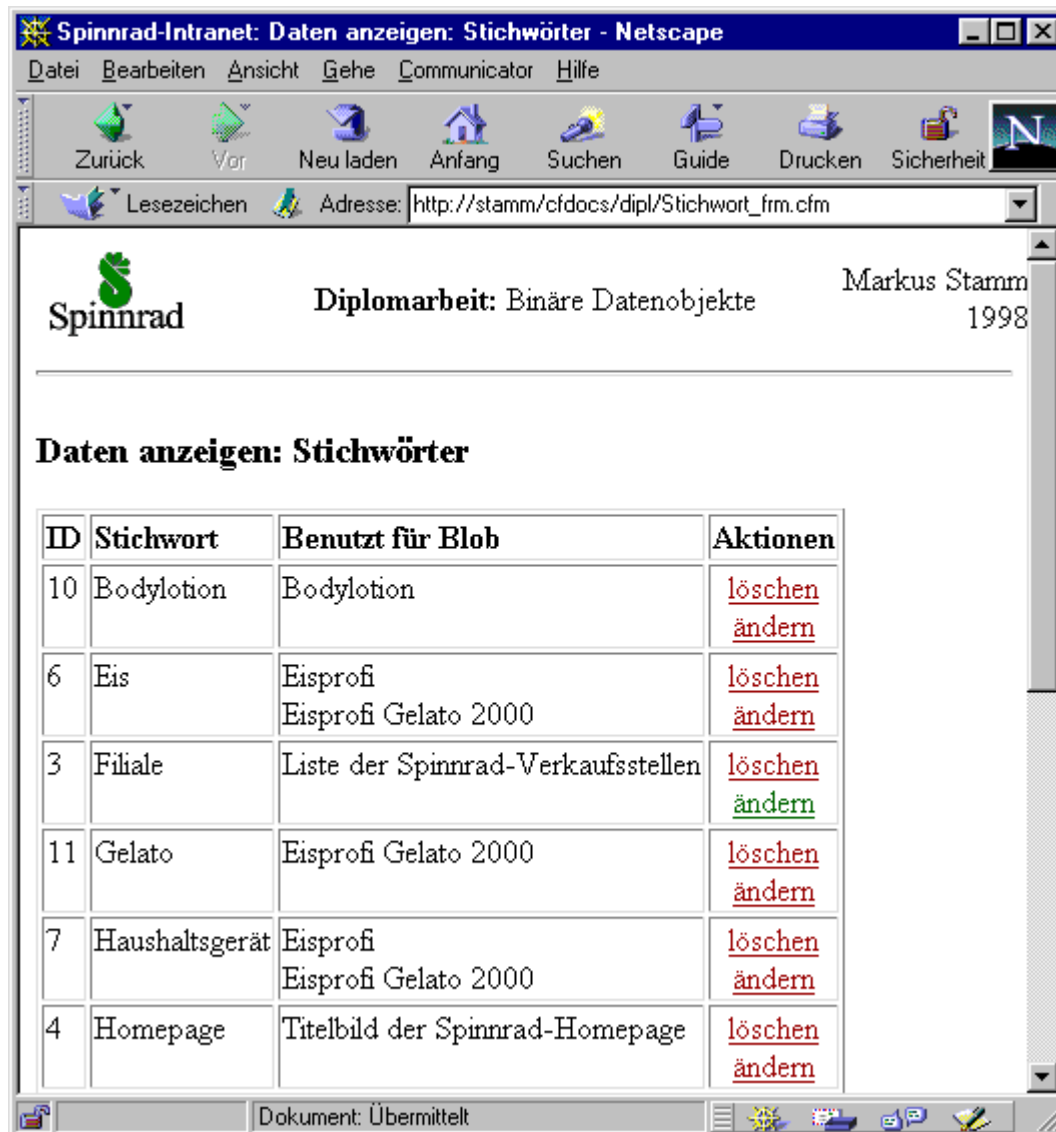


Abbildung 4-2: Datenanzeigeformular für Stichwörter

Die dritte Spalte zeigt die BLOBS an, die in einer Beziehung zu dem jeweiligen Stichwort stehen. Zu jedem Datensatz können in der vierten Spalte die Aktionen „Ändern“ und „Löschen“ ausgewählt werden. Das Formular wurde mit dem folgenden CFML-Code erzeugt:

## Stichwort\_frm.cfm

```

(1)<cfquery name="Stichwort" datasource="SunDB">
(2)   select STICHWORT.*,BLOB.BESCHREIBUNG
(3)       from STICHWORT,STICHWORT_BLOB,BLOB
(4)       where STICHWORT.STICHWORTID*=STICHWORT_BLOB.STICHWORTID
(5)           and STICHWORT_BLOB.BLOBID*=BLOB.BLOBID
(6)       order by STICHWORT_DEUTSCH
(7)</cfquery>

(8)<cf_head title="Daten anzeigen: Stichwort">

(9)<table border="1">
(10)<tr>
(11)   <td valign="top"><b>ID</b></td>
(12)   <td valign="top"><b>Stichwort</b></td>
(13)   <td valign="top"><b>Benutzt für BLOB</b></td>
(14)   <td valign="top"><b>Aktionen</b></td>
(15)</tr>
(16)<cfoutput query="Stichwort" group="STICHWORTID">
(17)<tr>
(18)   <td valign="top">#STICHWORTID#</td>
(19)   <td valign="top">#STICHWORT_DEUTSCH#</td>
(20)   <td valign="top">
(21)       <cfoutput>#BESCHREIBUNG#<br></cfoutput>
(22)   </td>
(23)   <td align="center" valign="top">
(24)       <a href="Stichwort_delete.cfm?ID=#STICHWORTID#">löschen</a><br>
(25)       <a href="Stichwort_update_frm.cfm?ID=#STICHWORTID#">ändern</a><br>
(26)   </td>
(27)</tr>
(28)</cfoutput>
(29)</table>

(30)<cf_foot>

```

Mit dem CFQUERY-Tag in den Zeilen (1) bis (7) wird die SQL-Abfrage definiert, die alle in der Datenbank gespeicherten Stichwörter zurückliefert. Dabei wird die Tabelle Stichwort über einen Left-Outer-Join<sup>13</sup> mit der Tabelle BLOB verknüpft. Die Verwendung dieses Joins stellt sicher, daß auch die Stichwörter in der Ergebnismenge vorkommen, die mit keinem BLOB verknüpft sind.

<sup>13</sup> (Left-) Outer-Joins sind eine Erweiterung von normalen Joins. Sie liefern zusätzlich noch die Zeilen einer Tabelle (der linken Tabelle), die nicht mit einer Zeile der anderen Tabelle (rechten Tabelle) verknüpft werden können ([Klei97a]).

Die Zeile (8) dient dann der Ausgabe von einleitenden HTML-Tags und der Überschrift für dieses Formular. Die Tabelle und die erste Zeile, die die Überschriften enthält, wird in den Zeilen (9) bis (15) definiert und in Zeile (29) abgeschlossen.

Alle HTML- und CFML-Befehle zwischen den Zeilen (16) und (28) werden nun für jeden Datensatz in der Ergebnismenge, bei dem sich das Feld STICHWORTID unterscheidet, genau einmal ausgeführt. Dies wird durch das CFOUTPUT-Tag in Zeile (16) durch die Angabe des Parameters `query="Stichwort"` und `group="STICHWORTID"` verursacht. Der zweite Parameter bewirkt eine Gruppierung über die Spalte STICHWORTID. In Zeile (21) werden dann durch das geschachtelte CFOUTPUT-Tag jeweils alle Beschreibungen der BLOBs für jedes Stichwort ausgegeben (siehe Abbildung 4-2, Stichwort „Eis“).

Mit dem CF\_FOOT-Tag in Zeile (30) werden die benötigten HTML-Anweisungen ausgegeben um das Dokument korrekt abzuschließen.

Die Formulare für die Entitäten *Dateityp*, *Ersteller*, *Digitalisiergerät* und *Bildkategorie* sind genauso aufgebaut und werden im Anhang C abgebildet. Die Anzeigeformulare für BLOBs und Bilder sind dagegen umfangreicher, da sie mehr Informationen über die in Beziehung zu ihnen stehenden Entitäten ausgeben. Beispielhaft wird im Weiteren das Formular für die Anzeige von Bildern erläutert.

ID	Beschreibung	Bildkategorie	Dateigröße (in Byte)	Bildgröße E x H (in Pixel)	Farbtiefe (in Bit)	Farbsystem	Aufnahmedatum	Original	Endung Dateityp	Ersteller	Digitalisiergerät	Stichworte	Aktionen
5	Bodylotion	<a href="#">Produktfoto</a>	10507	71 x 200	8	Palette	13.05.1998	nein	gif image/gif	Christian Mark	Sony Marvica	Körperpflege Bodylotion	<a href="#">anzeigen/speichern</a> <a href="#">löschen</a> <a href="#">ändern</a>
4	Eisprofi	<a href="#">Produktfoto</a>	14103	115 x 68	8	Palette	19.08.1998	nein	gif image/gif	Christian Mark	Sony Marvica	Eis Haushaltsgerät Maschine	<a href="#">anzeigen/speichern</a> <a href="#">löschen</a> <a href="#">ändern</a>
6	Eisprofi Gelato 2000	<a href="#">Produktfoto</a>	10477	147 x 200	8	Palette	12.04.1998	ja	gif image/gif	Christian Mark	Sony Marvica	Eis Haushaltsgerät Maschine Gelato	<a href="#">anzeigen/speichern</a> <a href="#">löschen</a> <a href="#">ändern</a>
3	Titelbild der Spinnrad_Homepage	Kollage	45086	x	8	Palette		nein	gif image/gif	Christian Mark	Verschiedene	Homepage Titelbild	<a href="#">anzeigen/speichern</a> <a href="#">löschen</a>

Abbildung 4-3: Datenanzeigeformular für Bilder

Das in Abbildung 4-3 gezeigte Formular wird durch den folgenden Cold Fusion-Code erzeugt:

### Bild\_frm.cfm

```
(1)<cfquery name="Bilder" datasource="SunDB">
(2)   select BILD.*,BLOB.BESCHREIBUNG,BLOB.GROSSE,ERSTELLER,
      DATEITYP.*,DIGITALISIERGERAT
(3)   from BLOB, BILD, DATEITYP, ERSTELLER, DIGITALISIERGERAT,
      PRODUKTFOTO, ART_PRODFOTO
(4)   where BLOB.DATEITYPID=DATEITYP.DATEITYPID
(5)   and BLOB.ERSTELLERID*=ERSTELLER.ERSTELLERID
(6)   and BILD.DIGITALISIERGERATID*=DIGITALISIERGERAT.DIGITALISIERGERATID
(7)   and BLOB.BLOBID=BILD.BLOBID
(8)   and BLOB.BLOBID*=PRODUKTFOTO.BLOBID
(9)   and BLOB.BLOBID*=ART_PRODFOTO.BLOBID
(10)  order by BLOB.BESCHREIBUNG
(11)</cfquery>

(12)<cf_head title="Daten anzeigen: Bilder">

(13)<table border="1">
(14)<tr>
(15)  <td valign="top"><b>ID</b></td>
(16)  <td valign="top"><b>Beschreibung</b></td>
(17)  <td valign="top"><b>Bildkategorie</b></td>
(18)  <td valign="top" align="center"><b>Dateigröße</b><br>(in Byte)</td>
(19)  <td valign="top" align="center"><b>Bildgröße</b><br>B x H<br>(in Pixel)</td>
(20)  <td valign="top" align="center"><b>Farbtiefe</b><br>(in Bit)</td>
(21)  <td valign="top" align="center"><b>Farbsystem</b></td>
(22)  <td valign="top" align="center"><b>Aufnahmedatum</b></td>
(23)  <td valign="top" align="center"><b>Original</b></td>
(24)  <td valign="top" align="center"><b>Endung<br>Dateityp</b></td>
(25)  <td valign="top"><b>Ersteller</b></td>
(26)  <td valign="top"><b>Digitalisiergerät</b></td>
(27)  <td valign="top"><b>Stichworte</b></td>
(28)  <td valign="top"><b>Aktionen</b></td>
(29)</tr>
(30)<cfloop query="Bilder">
(31)  <tr>
(32)  <cfoutput>
(33)    <td valign="top">#BLOBID#</td>
(34)    <td valign="top">#BESCHREIBUNG#&nbsp;</td>
(35)  </cfoutput>
(36)  <td valign="top">
(37)    <cfquery name="GetBildKat" datasource="SunDB">
(38)      select BILDKATEGORIE,BLOBID
```

```
(39)         from BILDKAT_BILD,BILDKATEGORIE
(40)         where BILDKAT_BILD.BILDKATEGORIEID=BILDKATEGORIE.BILDKATEGORIEID
(41)         and BILDKAT_BILD.BLOBID=#BLOBID#
(42)     </cfquery>
(43)     <cfoutput query="GetBildKat">
(44)         <cfif #BILDKATEGORIE# is 'Produktfoto' >
(45)             <a href="Produktfoto_frm.cfm?id=#BLOBID#">#BILDKATEGORIE#</a>
(46)         <cfelse>
(47)             #BILDKATEGORIE#
(48)         </cfif>
(49)         <br>
(50)     </cfoutput>
(51)     &nbsp;
(52) </td>
(53) <cfoutput>
(54)     <td valign="top" align="center">#GROSSE#&nbsp;</td>
(55)     <td valign="top" align="center">#BILDBREITE# x #BILDHOHE#</td>
(56)     <td valign="top" align="center">#FARBTIEFE#&nbsp;</td>
(57)     <td valign="top" align="center">#FARBSYSTEM#&nbsp;</td>
(58)     <td valign="top" align="center">#AUFNAHME DATUM#&nbsp;</td>
(59)     <td valign="top" align="center">
(60)         <cfif #ORIGINAL#>
(61)             ja
(62)         <cfelse>
(63)             nein
(64)         </cfif>
(65)     </td>
(66)     <td valign="top" align="center">
(67)         #DATEIENDUNG#<br>#CONTENTTYPE#/#CONTENTSUBTYPE#
(68)     </td>
(69)     <td valign="top">#ERSTELLER#&nbsp;</td>
(70)     <td valign="top">#DIGITALISIERGERAT#&nbsp;</td>
(71) </cfoutput>
(72) <td valign="top">
(73) <cfquery name="GetStichwort" datasource="SunDB">
(74)     select STICHWORT.STICHWORT_DEUTSCH
(75)     from STICHWORT,STICHWORT_BLOB
(76)     where STICHWORT.STICHWORTID=STICHWORT_BLOB.STICHWORTID
(77)     and STICHWORT_BLOB.BLOBID=#BLOBID#
(78) </cfquery>
(79) <cfoutput query="GetStichwort">
(80)     #STICHWORT_DEUTSCH#<br>
(81) </cfoutput>
(82) &nbsp;
(83) </td>
(84) <td align="center" valign="top">
(85) <cfoutput>
(86)     <a href="Blob_view.cfm?ID=#BLOBID#">anzeigen/speichern</a><br>
(87)     <a href="Bild_delete.cfm?ID=#BLOBID#">löschen</a><br>
```

```
(88)         <a href="Bild_update_frm.cfm?ID=#BLOBID#">ändern</a><br>
(89)     </cfoutput>
(90)     </td>
(91)</tr>
(92)</cfloop>
(93)</table>
(94)<p>
(95)Falls es sich bei einem Bild um ein Produktfoto handelt, können Sie weitere Daten
    durch Klicken auf den Link 'Produktfoto' einsehen und ändern.
(96)<p>
(97)<a href="Bild_insert_frm.cfm">Bild hinzufügen.</a>
(98)<cf_foot>
```

In den Zeilen (1) bis (11) wird die SQL-Abfrage *Bilder* definiert. Sie führt eine Verknüpfung der Tabellen *BLOB*, *Bild*, *Ersteller*, *Digitalisiergeraet*, *Produktfoto* und *Art\_Prodfoto* durch. Dabei werden die vier letztgenannten Tabellen über ein Outer Join verbunden, damit auch die Bilder in der Ergebnismenge vorkommen, denen kein Ersteller oder Digitalisiergerät zugeordnet ist oder die kein Produktfoto sind.

Die Zeilen (12) bis (29) dienen der Ausgabe von einleitenden Tags und der Kopfzeile der Tabelle.

Zwischen den Zeilen (30) und (92) wird eine Schleife über jeden Datensatz der oben definierten Abfrage ausgeführt. Bei jedem Schleifendurchlauf wird genau eine Zeile der auszugebenden Tabelle durch das `<TR>` und `</TR>`-Tag in den Zeilen (31) und (91) erzeugt. Darin werden zunächst die Spalten *BLOBID* und *BESCHREIBUNG* ausgegeben (Zeilen (32) bis (35)). Anschließend werden mit einer weiteren Abfrage die zugeordneten Bildkategorien ermittelt und ausgegeben ((37) bis (50)). Falls die Bildkategorie „*Produktfoto*“ heißt, wird dieses Wort als Link auf die Seite *Produktfoto\_frm.cfm* mit dem URL-Parameter *id=#BLOBID#* dargestellt (45), wobei die Variable *BLOBID* die ID des gerade angezeigten Datensatzes enthält.

Mit den Zeilen (53) bis (71) werden wieder verschiedene Spalten mit Daten aus der Abfrage *Bilder* erzeugt. Dabei wird der boolesche Wert der Spalte *ORIGINAL* mit Hilfe der IF-Abfrage in den Zeilen (60) bis (64) als Wort „Ja“ bzw. „Nein“ ausgegeben.

Die Abfrage mit der zugehörigen Ausgabe in den Zeilen (73) bis (81) gibt alle Stichwörter aus, die dem jeweils angezeigten Datensatz zugeordnet sind. Schließlich

werden in den Zeilen (85) bis (89) noch die Links auf die unterschiedlichen Datenbearbeitungsformulare erzeugt. Die ID des jeweiligen Bildes wird dabei als URL-Parameter an die Anwendungsseiten übergeben.

Die Zeilen (93) bis (98) enthalten abschließende Tags für die Tabelle und das HTML-Dokument, sowie erklärenden Text und einen Link auf das Formular zum Einfügen von Bildern in die Datenbank (siehe Kap. 4.3).

## 4.2 Anzeige der Binärdaten

Für die Anzeige der Binärdaten wird die Eigenschaft von Webbrowsern ausgenutzt, daß sie zum einen Bilder direkt oder mit Hilfe von Plug-Ins im Browser anzeigen können und zum anderen die Applikationen, die mit dem jeweiligen Dateityp verknüpft sind, auch starten können. Dadurch ist es nicht nötig, eigene Anzeigeprogramme für die verschiedenen Arten der Binärdaten zu programmieren. Außerdem können die so angezeigten Bilder oder anderen Binärdaten mit Hilfe der Speicherfunktion des Browsers oder der zugehörigen Applikation auf dem jeweiligen Client zur späteren Weiterverwendung gespeichert oder gedruckt werden. Hierin liegt auch der große Vorteil des Webbrowsers als Frontend gegenüber „klassischen“ Client-Applikationen.

Damit der Webbrowser die Binärdaten vom Server herunter laden kann, müssen sie dort in Form einer Datei auf dem Filesystem zur Verfügung stehen. Mit der Cold Fusion Anwendungsseite `blob_view.cfm` werden die Daten eines BLOBs in eine temporäre Datei auf dem Server gespeichert. Anschließend wird dann der Webbrowser dazu veranlaßt, die temporäre Datei vom Server anzufordern und herunter zu laden.

### Blob\_view.cfm

```
(1)<cfinclude template="const.cfm">

(2)<cfquery name="HoleDateiTyp" datasource="SunDB">
(3)    select DATEITYP.DATEIENDUNG
(4)        from BLOB,DATEITYP
(5)        where BLOB.DATEITYPID=DATEITYP.DATEITYPID
(6)            and BLOB.BLOBID=#url.id#
(7)</cfquery>
```

```
(8)<cfoutput query="HoleDateiTyp">
(9)   <cfset Endung=#LCase(DATEIENDUNG)#>
(10)</cfoutput>

(11)<CFX_GETIMAGE DATASOURCE="SunDB" USER="stamm" PASSWORD="frusips"
(12)   SQL="select BINARDATEN
(13)         from BLOB
(14)         where BLOBID=#URL.ID#"
(15)   OUTPUT="#wwwroot##wwwsubdir#temp#url.ID#. #Endung#">

(16)<cflocation url="/#wwwsubdir#temp#url.id#. #Endung#">
```

Mit der als URL-Parameter übergebenen *BLOBID* wird zunächst in den Zeilen (2) bis (7) der Dateityp des BLOBs bestimmt. Für die Dateiendung der temporären Datei wird in den Zeilen (8) bis (10) die Variable *Endung* mit dem entsprechenden Wert aus der Tabelle *Dateityp* gesetzt.


In den Zeilen (11) bis (15) werden mit Hilfe des *CFX\_GETIMAGE*-Tags die Binärdaten aus der Datenbank ausgelesen und auf dem Server gespeichert. Dabei wird der Name des Verzeichnisses zum Speichern aus den Variablen *wwwroot* und *wwwsubdir* zusammengesetzt. Diese Variablen werden in der Datei *const.cfm* gesetzt, die in Zeile (1) in das Dokument eingebunden wurde. Der Dateiname wird aus dem Wert „temp“, der ID-Nummer des BLOBs und der passenden Endung zusammengesetzt.

Zeile (16) veranlaßt den Webbrowser letztendlich dazu, die zuvor auf dem Server gespeicherte Datei zu laden. Die weitere Vorgehensweise des Browsers ist nun von der Dateiendung - also vom Typ der Datei - abhängig.

Die Datei auf dem Server kann an dieser Stelle noch nicht wieder gelöscht werden, da bis zur Anfrage des Browsers und des anschließenden Herunterladens eine nicht vorhersagbare Zeitdauer liegen kann. Dadurch können sich je nach Intensität der Nutzung des Systems mit der Zeit viele temporäre Dateien in dem entsprechenden Verzeichnis ansammeln. Mit Hilfe einer Anwendungsseite und dem Cold Fusion Scheduler-Mechanismus kann aber dafür gesorgt werden, daß das Verzeichnis regelmäßig geleert wird ([Alla98a]).

### **4.3 Dateneinfügeformulare**

Zum Einfügen von Daten in die Datenbank werden jeweils zwei Cold Fusion-Dokumente benötigt. Das erste stellt ein Formular zur Dateneingabe und einige Auswahlfelder zur Verfügung, das zweite verarbeitet dann nach dem Betätigen des Absende-Buttons die Eingaben in den Formularfeldern und fügt die Daten in die Datenbank ein. Im Folgenden wird eine Bildschirmkopie des entsprechenden Formulars (Abbildung 4-4) und anschließend der Quellcode der Anwendungsseite, die die Daten verarbeitet, gezeigt. Der Quellcode wurde zur besseren Nachvollziehbarkeit in kleinere Blöcke unterteilt und jeder Block wird dann direkt erläutert.



Diplomarbeit: Binäre Datenobjekte

Markus Stamm  
1998

---

**Daten einfügen: Bild**

Bitte wählen Sie eine Bilddatei aus, die Sie in der Datenbank speichern möchten.

Geben Sie der Datei einen Titel oder kurze Beschreibung:

Bildbreite\* (in Pixel):

Bildhöhe\* (in Pixel):

\*Falls es sich bei dem Bild um ein Computerze Bild (gif) oder JPEG-Bild (jpg) handelt, brauchen Sie die Felder 'Bildhöhe' und 'Bildbreite' nicht ausfüllen.

Farbtiefe (in Bit):

Farbsystem:

Aufnahmedatum (tt.mm.jjjj): ..19

Original (oder bearbeitet):

Ersteller:

Christian Mark

Markus Stamm

Sybase Inc.

Neuer Ersteller:

Digitalisiergerät,  
mit dem das Bild erstellt wurde:

Sony Mavica

Verschiedene

Neues Digitalisiergerät:

Stichworte:

Bodylotion

Eis

Filiale

Gelato

Haushaltsgerät

Neue Stichwörter  
(mehrere Stichwörter mit Semikolon getrennt eingeben):

Bildkategorie:

Kollage

Produktfoto

Neue Kategorie  
(mehrere Kategorien mit Semikolon getrennt eingeben):

Der Hochladevorgang kann je nach Größe der Binärdatei bis zu 5 Minuten dauern.

---

[\[Suche\]](#) [\[Start-Seite\]](#)

[\[BLOB\]](#) [\[Bild\]](#) [\[Text\]](#) [\[Ersteller\]](#) [\[Digitalisiergerät\]](#)

[\[Stichwort\]](#) [\[Bildkategorie\]](#) [\[Dateityp\]](#) [\[Artikel\]](#)

Abbildung 4-4: Formular zum Einfügen von Bildern (ohne Netscape-Rahmen)

## Bild\_insert.cfm

```
(1)<cfinclude template="const.cfm">
(2)<!-- Datei vom Client auf Server übertragen --->
(3)<cffile action="upload" filefield="BINARDATEN" destination="#wwwroot##wwwsubdir#"
      nameconflict="Makeunique" attributes="temporary">
(4)<cfset Dateiname = #file.ServerDirectory# & "\" & #file.ServerFile#>
```

Nachdem in Zeile (1) wieder die Datei `const.cfm` mit den Konstanten eingebunden wurde, wird die in dem Formular ausgewählte Datei vom Client auf den Server geladen. Anschließend stehen verschiedene Variablen mit dem Präfix `file` zur Verfügung.

```
(5)<cfquery name="InsertDateityp" datasource="SunDB">
(6)   insert into DATEITYP (DATEIENDUNG,CONTENTTYPE,CONTENTSUBTYPE)
(7)     select '#file.ServerFileExt#', '#file.ContentType#',
(8)         '#file.ContentSubType#'
(9)       where not exists
(10)         (select *
(11)           from DATEITYP
(12)            where CONTENTTYPE='#file.ContentType#'
(13)              and CONTENTSUBTYPE='#file.ContentSubType#'
(14)              and DATEIENDUNG='#file.ServerFileExt#')
(14)</cfquery>
```

Mit der SQL-Abfrage in den Zeilen (5) bis (14) wird der Dateityp der Datei angelegt, soweit er noch nicht in der Tabelle existiert. Dateiendung sowie Contenttype und Contentsubtype können direkt aus den `file`-Variablen ermittelt werden.

```
(15)<!-- aus Liste gewählter Ersteller --->
(16)<cfif ParameterExists(form.Ersteller)>
(17)   <cfset insErst=form.Ersteller>
(18)<cfelseif form.Ersteller_neu is "">
(19)   <!-- kein Ersteller --->
(20)   <cfset insErst="null">
(21)<cfelse>
(22)   <!-- neuen Ersteller in die DB einfügen --->
(23)   <cfquery name="InsertErsteller" datasource="SunDB">
(24)     insert into ERSTELLER (ERSTELLER)
(25)       select '#form.Ersteller_neu#'
```

```
(26)         where not exists
(27)             (select *
(28)                 from ERSTELLER
(29)                 where ERSTELLER.ERSTELLER='#form.Ersteller_neu#')
(30) </cfquery>

(31) <cfquery name="GetErID" datasource="SunDB">
(32)     select @@identity as EID
(33) </cfquery>
(34) <cfoutput query="GetErID">
(35)     <cfset insErst=EID>
(36) </cfoutput>
(37)</cfif>
```

In diesem Abschnitt werden mit Hilfe der CFIF-, CFELSEIF- und CFELSE-Tags drei verschiedene Zustände im Eingabeformular überprüft und entsprechend ausgewertet. In der CFIF-Anweisung wird geprüft, ob in dem Auswahllistenfeld *Ersteller* (links im Formular) ein bereits in der Datenbank vorhandener Ersteller ausgewählt wurde. Wenn ja, wird die in `form.Ersteller` gespeicherte ID in der Variablen `insErst` gespeichert. Andernfalls wird geprüft, ob auch in dem Texteingabefeld `Ersteller_neu` (rechts im Formular) kein Eintrag vorgenommen wurde. In diesem Fall wird die Variable `insErst` auf `'null'` gesetzt. Im dritten Fall (keine Auswahl aus der Auswahlliste aber Eingabe eines Erstellers im Textfeld) wird der eingetragene Ersteller neu angelegt (Zeilen (23) bis (29)) und die Variable `insErst` auf die ID des neuen Erstellers gesetzt (Zeilen (31) bis (36)). Die in Zeile (32) abgefragte Sybase-Variable `@@identity` enthält immer den zuletzt erzeugten Wert einer `identity`-Spalte.

```
(38)<!-- Datensatz für BLOB erzeugen -->
(39)<cfquery name="GenBlob" datasource="SunDB">
(40)     insert into BLOB (DATEITYPID,BESCHREIBUNG,GROSSE,ERSTELLERID)
(41)         select DATEITYPID,'#form.Beschreibung#','#file.FileSize#','#insErst#
(42)             from DATEITYP
(43)             where CONTENTTYPE='#file.ContentType#'
(44)                 and CONTENTSUBTYPE='#file.ContentSubType#'
(45)                 and DATEIENDUNG='#file.ServerFileExt#'
(46)</cfquery>

(47)<cfquery name="GetBID" datasource="SunDB">
(48)     select @@identity as BID
```

```
(49)</cfquery>
(50)<cfoutput query="GetBID">
(51)   <cfset BlobID = #BID#>
(52)</cfoutput>
```

In den Zeilen (39) bis (46) wird nun ein Datensatz in der Tabelle BLOB angelegt. Dabei werden bis auf die Spalte mit den Binärdaten alle Attribute gesetzt. Mit den Anweisungen der Zeilen (47) bis (52) wird dann wiederum die ID des neu angelegten Datensatzes ermittelt und der Variablen BLOBID zugewiesen.

```
(53)<!--- Datei vom Server in die Datenbank übertragen --->
(54)<CFX_PUTIMAGE DATASOURCE="SunDB" USER="stamm" PASSWORD="frusips"
(55)   SQL="update BLOB
(56)       set BINARDATEN=?
(57)       where BLOBID= #BlobID#"
(58)   INPUT="#Dateiname#">
```

Im darauffolgenden Abschnitt ((53) bis (58)) werden nun nachträglich die Binärdaten in den entsprechenden Datensatz eingefügt.

```
(59)<!--- neue Stichworte in Tabellen einfügen --->
(60)<cfloop index="idxStichwort" list="#Stichwort_neu#" delimiters=";">
(61)   <cfquery name="InsertStichwort" datasource="SunDB">
(62)     insert into STICHWORT (STICHWORT_DEUTSCH)
(63)       select '#idxStichwort#'
(64)       where not exists
(65)         (select *
(66)           from STICHWORT
(67)           where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#')
(68)   </cfquery>

(69)   <!--- Verknüpfung zwischen Blob und neuen Stichworten herstellen --->
(70)   <cfquery name="Insert_Stichw_Blob" datasource="SunDB">
(71)     insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(72)       select STICHWORT.STICHWORTID,#BlobID#
(73)       from STICHWORT
(74)       where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#'
(75)   </cfquery>
(76)</cfloop>

(77)<cfif ParameterExists(form.Stichwort)>
```

```
(78) <!-- Verknüpfung zwischen Blob und bestehenden Stichworten herstellen -->
(79) <cfquery name="Insert_Stichw_Blob2" datasource="SunDB">
(80)     insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(81)         select STICHWORT.STICHWORTID,#BlobID#
(82)         from STICHWORT
(83)         where STICHWORT.STICHWORTID in (#form.Stichwort#)
(84) </cfquery>
(85)</cfif>
```

Der Abschnitt von Zeile (59) bis (85) dient dem Einfügen und zuordnen von Stichwörtern. Im Gegensatz zum Ersteller können einem BLOB mehrere Stichwörter zugeordnet werden. Deshalb muß hier anders vorgegangen werden.

In den Zeilen (60) bis (76) werden die in dem Texteingabefeld `Stichwort_neu` eingegebenen Stichwörter nacheinander in der Datenbank angelegt und die Beziehung zum *BLOB* in der Tabelle *STICHWORT\_BLOB* hergestellt. Dazu wird in der Zeile (60) eine Schleife über alle Elemente der Liste `Stichwort_neu` initialisiert. Als Trennzeichen dient in dieser Liste das Semikolon. Bei jedem Schleifendurchlauf enthält die Variable `idxStichwort` jeweils ein Element der Liste. Dieses wird dann durch die SQL-Anweisung in den Zeilen (61) bis (68) in die Datenbank eingefügt. Durch die *where*-Klausel in den Zeilen (64) bis (67) wird verhindert, daß ein bereits existierendes Stichwort noch einmal in die Datenbank eingefügt wird (falls der Benutzer es z.B. in das Textfeld eingegeben hat, obwohl es in der Auswahlbox bereits vorhanden war).

In den Zeilen (79) bis (84) werden dann noch die Beziehungen zwischen dem neuen BLOB und den bestehenden Stichwörtern, die in der Auswahlbox gewählt wurden, hergestellt. Dabei enthält die Variable `form.Stichwort` eine kommasetrennte Liste mit den Nummern der Stichwörter.

```
(86)<!-- aus Liste gewähltes Digitalisiergerät -->
(87)<cfif ParameterExists(form.DigiGeraet)>
(88) <cfset insDigiG=form.DigiGeraet>
(89)<cfelseif form.DigiGeraet_neu is "">
(90) <!-- kein Digitalisiergerät -->
(91) <cfset insDigiG="null">
(92)<cfelse>
(93) <!-- neues DigiGeraet in die DB einfügen -->
(94) <cfquery name="InsertDigiGeraet" datasource="SunDB">
```

```
(95)      insert into DIGITALISIERGERAT (DIGITALISIERGERAT)
(96)          select '#form.DigiGeraet_neu#'
(97)          where not exists
(98)              (select *
(99)                  from DIGITALISIERGERAT
(100)                 where DIGITALISIERGERAT='#form.DigiGeraet_neu#')
(101) </cfquery>
(102) <cfquery name="GetDGID" datasource="SunDB">
(103)     select @@identity as DGID
(104) </cfquery>
(105) <cfoutput query="GetDGID">
(106)     <cfset insDigiG=DGID>
(107) </cfoutput>
(108)</cfif>
```

Das Einfügen eines neuen Digitalisiergerätes bzw. die Verknüpfung mit einem bestehenden funktioniert genauso wie die entsprechenden Vorgänge mit einem Ersteller. Die Variable `insDigiG` enthält die ID des Digitalisiergerätes.

```
(109)<!-- zusätzliche Bilddaten einfügen --->
(110)<cfif ParameterExists(form.Original)>
(111) <cfset ins_Original=1>
(112)<cfelse>
(113) <cfset ins_Original=0>
(114)</cfif>

(115)<CFX_IMGINFO NAME="ImageInfo" FILENAME="#Dateiname#">
(116)<CFIF #ImageInfo.RecordCount# is 0>
(117) <cfif form.Bildhoehe is '' >
(118)     <cfset ins_Bildhoehe='null'>
(119) <cfelse>
(120)     <cfset ins_Bildhoehe='#form.Bildhoehe#'>
(121) </cfif>
(122) <cfif form.Bildbreite is '' >
(123)     <cfset ins_Bildbreite='null'>
(124) <cfelse>
(125)     <cfset ins_Bildbreite='#form.Bildbreite#'>
(126) </cfif>
(127)<CFELSE>
(128) <cfset ins_Bildhoehe='#ImageInfo.Height#'>
(129) <cfset ins_Bildbreite='#ImageInfo.Width#'>
(130)</CFIF>

(131)<cfif form.Farbtiefe is '' >
(132) <cfset ins_Farbtiefe='null'>
```

```

(133)<cfelse>
(134)  <cfset ins_Farbtiefe='#form.Farbtiefe#'>
(135)</cfif>

(136)<cfif form.Farbsystem is '' >
(137)  <cfset ins_Farbsystem='null'>
(138)<cfelse>
(139)  <cfset ins_Farbsystem=chr(39) & form.Farbsystem & chr(39)>
(140)</cfif>

(141)<cfif form.Jahr is '' or form.Monat is '' or form.Tag is ''>
(142)  <cfset ins_Aufnahmedatum='null'>
(143)<cfelse>
(144)  <cfset ins_Aufnahmedatum=chr(39) & form.Jahr & '-' & form.Monat & '-' &
      form.Tag & chr(39)>
(145)</cfif>

(146)<cfquery name="InsBild" datasource="SunDB">
(147)  insert into BILD
(148)    values (#BlobID#,#insDigiG#,#ins_Bildhoehe#,#ins_Bildbreite#,
(149)            #ins_Farbtiefe#,#PreserveSingleQuotes(ins_Farbsystem)#,
(150)            #PreserveSingleQuotes(ins_Aufnahmedatum)#,#ins_Original#)
(151)</cfquery>

```

In den Zeilen (109) bis (145) werden die Daten für die Spalten der Tabelle *Bild* aufbereitet. So wird z.B. der Zustand der Checkbox *Original* (*form.Original*) je nachdem, ob sie aktiviert ist oder nicht, als boolescher Wert '1' bzw. '0' in der Variablen *ins\_Original* gespeichert ((110) bis (114)). Außerdem werden Bildhöhe und -breite mit dem *CFX\_IMGINFO*-Tag ermittelt bzw. aus den entsprechenden Texteingabefeldern übernommen.

Anschließend werden die Eingabefelder für die Farbtiefe, das Farbsystem und das Aufnahmedatum ausgewertet und die Werte in den Variablen *ins\_Farbtiefe*, *ins\_Farbsystem* und *ins\_Aufnahmedatum* gespeichert.

Die Zeilen (146) bis (151) enthalten dann die SQL-Abfrage, mit der die entsprechenden Daten in die Tabelle *BILD* eingefügt werden.

```

(152)<!-- neue Bildkategorie in Tabellen einfügen --->
(153)<cfloop index="idxKategorie" list="#Kategorie_neu#" delimiters=";">
(154)  <cfquery name="InsertKategorie" datasource="SunDB">
(155)    insert into BILDKATEGORIE (BILDKATEGORIE)
(156)      select '#idxKategorie#'

```

```
(157)         where not exists
(158)             (select *
(159)                 from BILDKATEGORIE
(160)                 where BILDKATEGORIE='#idxKategorie#')
(161) </cfquery>

(162) <!-- Verknüpfung zwischen BLOB und neuen Kategorien herstellen -->
(163) <cfquery name="Insert_BildKat_Blob" datasource="SunDB">
(164)     insert into BILDKAT_BILD (BILDKATEGORIEID,BLOBID)
(165)         select BILDKATEGORIEID,#BlobID#
(166)         from BILDKATEGORIE
(167)         where BILDKATEGORIE='#idxKategorie#'
(168) </cfquery>
(169)</cfloop>

(170)<cfif ParameterExists(form.Kategorie)>
(171) <!-- Verknüpfung zwischen BLOB und bestehenden Kategorien herstellen -->
(172) <cfquery name="Insert_BildKat_Blob2" datasource="SunDB">
(173)     insert into BILDKAT_BILD (BILDKATEGORIEID,BLOBID)
(174)         select BILDKATEGORIE.BILDKATEGORIEID,#BlobID#
(175)         from BILDKATEGORIE
(176)         where BILDKATEGORIE.BILDKATEGORIEID in (#form.Kategorie#)
(177) </cfquery>
(178)</cfif>
```

Die Vorgehensweise in den Zeilen (152) bis (178) zum Einfügen und Herstellen von Beziehungen zu den Bildkategorien ist prinzipiell genauso wie bei den Stichwörtern.

```
(179)<!-- Datei vom Server löschen -->
(180)<cffile action="Delete" File="#Dateiname#">

(181)<!-- Ausgabe -->
(182)<cflocation url="Bild_frm.cfm">
```

In Zeile (180) wird die zwischenzeitlich auf dem Server gespeicherte Datei mit den Binärdaten wieder gelöscht und in Zeile (182) wird der Webbrowser des Client dazu veranlaßt, das Anzeigeformular für die Bilder zu laden.

Mit dem hier beschriebenen Formular können nicht nur Bilder in die Datenbank eingefügt werden, sondern auch Ersteller, Digitalisiergeräte, Stichwörter und Bildkategorien. Für diese Entitäten gibt es keine eigenen Eingabeformulare, da die Daten nur in der Datenbank gespeichert werden sollen, wenn sie auch von

mindestens einem BLOB bzw. Bild benötigt werden. Ansonsten könnte man sehr schnell große Mengen an Daten erhalten, die nur Speicherplatz benötigen und die Performance des Systems mindern.


Das entsprechende Eingabeformular für BLOBs ist sehr ähnlich zu dem oben beschriebenen Formular für Bilder. Natürlich enthält es keine Eingabemöglichkeit für Digitalisiergeräte und Bildkategorien, da diese ja nur mit Bildern in einer Beziehung stehen. Das Formular für BLOBs ist dem Anhang C zu entnehmen.

#### **4.4 Datenänderungsformulare**

Ein Datenänderungsformular wird dazu benötigt, die Spalten eines bestehenden Datensatzes zu ändern. Vom Aufbau her unterscheidet es sich eigentlich nur unwesentlich von einem Eingabeformular. Der Unterschied besteht darin, daß die bereits vorhandenen Daten in den Textfeldern ausgegeben werden und dort direkt geändert werden können. In Abbildung 4-5 wird das Änderungsformular für Bilder angezeigt.

In der ersten Tabelle wird der Beschreibungstext des BLOBs, die Dateiendung und der Dateityp sowie der Ersteller und das Digitalisiergerät angezeigt. Die beiden letztgenannten können weiter unten geändert werden, indem aus der Liste der bestehenden Daten ein Ersteller bzw. ein Digitalisiergerät ausgewählt wird oder in den danebenstehenden Feldern ein neuer Ersteller bzw. ein neues Gerät eingegeben wird. Wird an diesen Feldern kein Eintrag vorgenommen, so bleibt der bestehende Eintrag erhalten.

Entitäten, die in einer m:n-Beziehung zu den Bildern stehen (Stichwort und Bildkategorie), werden in extra Tabellen aufgeführt. Durch aktivieren der Checkbox in der Spalte „Zuweisung aufheben“ kann die entsprechende Beziehung aufgehoben werden. Zusätzlich können andere bestehende und neue Stichwörter bzw. Bildkategorien in die Datenbank eingefügt und zugeordnet werden.


 Diplomarbeit: Binäre Datenobjekte
 
 Markus Stamm  
1998

---

**Daten ändern: Bild**

ID	Beschreibung	Endung	Content Type	Ersteller	Digitalisiergerät
5	Bodylotion	gif	image/gif	Christian Mark	Sony Marvica

Bildbreite (in Pixel): 
 Bildhöhe (in Pixel):

Farbtiefe (in Bit):

Aufnahme datum (tt.mm.jjjj):   
 Original (oder bearbeitet):

Farbsystem:

Ersteller:

Neuer Ersteller:

Digitalisiergerät:

Neues Digitalisiergerät:

Stichwort	Zuweisung entfernen
Körperpflege	<input type="checkbox"/>
Bodylotion	<input type="checkbox"/>

Stichwörter neu zuordnen:

Neue Stichwörter (neue Stichwörter mit Semikolon getrennt eingeben):

Bildkategorie	Zuweisung entfernen
Produktfoto	<input type="checkbox"/>

Bildkategorien neu zuordnen:

Neue Bildkategorien (neue Kategorien mit Semikolon getrennt eingeben):

---

[\[Suche\]](#) [\[Start-Seite\]](#)

[\[BLOB\]](#) [\[Bild\]](#) [\[Text\]](#) [\[Ersteller\]](#) [\[Digitalisiergerät\]](#)

[\[Stichwort\]](#) [\[Bildkategorie\]](#) [\[Dateizug\]](#) [\[Artikel\]](#)

Abbildung 4-5: Änderungsformular für Bilder (ohne Netscape-Rahmen)

Der komplette Cold Fusion-Code für die Auswertung des Formulars kann dem Anhang C entnommen werden. An dieser Stelle wird nur ein kurzer Auszug gezeigt

und erläutert. Mit der folgenden SQL-Anweisung wird die Zuweisung der markierten Stichwörter gelöscht.

#### Bild\_update.cfm

```
(1)      [...]  
(2) <cfquery name="DelStichw_Blob" datasource="SunDB">  
(3)      delete STICHWORT_BLOB  
(4)          where BLOBID=#url.ID#  
(5)          and STICHWORTID in (#form.delStichwort#)  
(6) </cfquery>
```

Die in Zeile (4) verwendete Variable `form.delStichwort` enthält eine kommasetrennte Liste mit den STICHWORTID-Werten der im Formular markierten Stichwörter.

Mit der folgenden Abfrage werden dann alle nicht mehr benötigten Stichwörter aus der Datenbank gelöscht.

```
(6) <cfquery name="DelStichwort" datasource="SunDB">  
(7)      delete STICHWORT  
(8)          where STICHWORTID not in  
(9)              (select STICHWORTID  
(10)                  from STICHWORT_BLOB)  
(11) </cfquery>  
(12)      [...]
```

## 4.5 Daten löschen

Um einen Datensatz aus einer Tabelle zu löschen muß lediglich der Link „Löschen“ in den Anzeigeformularen gewählt werden. Dadurch wird die jeweilige Cold Fusion-Anwendungsseite aufgerufen, die das Löschen eines Datensatzes durchführt. An diese Seite wird die jeweilige ID des zu löschenden Datensatzes als URL-Parameter übergeben.

Es gibt drei vom Aufbau her unterschiedliche Arten der Anwendungen zum Löschen von Daten:

- Löschen der Daten, die in einer 1:n-Beziehung zu den BLOBs stehen (*Ersteller* und *Digitalisiergerät*).
- Löschen von Daten, die in einer m:n-Beziehung zu den BLOBs stehen (*Stichwörter* und *Bildkategorien*)
- Löschen von BLOBs oder Bildern

Im Folgenden wird von jeder Variante jeweils eine Anwendung beispielhaft vorgestellt.

### Ersteller\_delete.cfm

```
(1)<cfquery name="UpdateBlob" datasource="SunDB">
(2)  update BLOB
(3)      set ERSTELLERID=null
(4)      where ERSTELLERID=#url.id#
(5)</cfquery>

(6)<cfquery name="DelErsteller" datasource="SunDB">
(7)  delete ERSTELLER
(8)      where ERSTELLERID=#url.id#
(9)</cfquery>

(10)<cflocation url="Ersteller_frm.cfm">
```

Mit dieser Anwendungsseite wird der im URL-Parameter `url.id` übergebene Ersteller gelöscht (Zeilen (6) bis (9)). Zuvor wird in der Tabelle *BLOB* die Spalte, über die die 1:n-Beziehung zu *Ersteller* hergestellt wurde, auf `null` gesetzt. Das

entsprechende BLOB hat somit keinen Ersteller mehr. In Zeile (10) wird die Ausgabe dann auf eine andere Seite umgeleitet.

#### Stichwort\_delete.cfm

```
(1)<cfquery name="DelStichwort_Blob" datasource="SunDB">
(2)  delete STICHWORT_BLOB
(3)      where STICHWORTID=#url.id#
(4)</cfquery>

(5)<cfquery name="DelStichwort" datasource="SunDB">
(6)  delete STICHWORT
(7)      where STICHWORTID=#url.id#
(8)</cfquery>

(9)<cflocation url="Stichwort_frm.cfm">
```

Da eine m:n-Beziehung durch eine eigene Tabelle repräsentiert wird, muß zum Löschen eines Datensatzes aus einer solchen Beziehung nicht nur der Datensatz selber gelöscht werden, sondern auch alle Datensätze aus der Beziehungs-Tabelle, an denen der zu löschende Datensatz beteiligt ist.

Im Beispiel der Stichwörter werden deshalb zuerst aus der Beziehungs-Tabelle *STICHWORT\_BLOB* alle Datensätze, die zu dem zu löschenden Stichwort gehören (Zeilen (1) bis (4)) und anschließend der Datensatz selber gelöscht ((5) bis (8)).

Das Löschen eines BLOBs oder eines Bildes ist weitaus umfangreicher als die beiden zuvor gezeigten Löschvorgänge, da diese beiden Entitäten in sehr vielen Beziehungen eingebunden sind. Bei dem Löschen eines Bildes muß deshalb darauf geachtet werden, daß alle anderen Daten, die mit dem zu löschenden Bild in Zusammenhang stehen und in keiner anderen Beziehung mehr benötigt werden, mit gelöscht werden.

Mit dem folgenden Cold Fusion-Code werden alle Datensätze aus allen Tabellen gelöscht, die mit dem im URL-Parameter übergebenen Bild im Zusammenhang stehen und nicht mehr in der Datenbank benötigt werden.

**Bild\_delete.cfm**

```
(1)<cfquery name="DelArtProdFoto" datasource="SunDB">
(2)   delete ART_PRODFOTO
(3)       where BLOBID=#url.id#
(4)</cfquery>
(5)<cfquery name="DelProdFoto" datasource="SunDB">
(6)   delete PRODUKTFOTO
(7)       where BLOBID=#url.id#
(8)</cfquery>
(9)<cfquery name="DelBildkatBild" datasource="SunDB">
(10)  delete BILDKAT_BILD
(11)      where BLOBID=#url.id#
(12)</cfquery>
(13)<cfquery name="DelBild" datasource="SunDB">
(14)  delete BILD
(15)      where BLOBID=#url.id#
(16)</cfquery>
(17)<cfquery name="DelStichwortBlob" datasource="SunDB">
(18)  delete STICHWORT_BLOB
(19)      where BLOBID=#url.id#
(20)</cfquery>
(21)<cfquery name="DelBlob" datasource="SunDB">
(22)  delete BLOB
(23)      where BLOB.BLOBID=#url.id#
(24)</cfquery>
(25)<cfquery name="DelDateityp" datasource="SunDB">
(26)  delete DATEITYP
(27)      where DATEITYPID not in
(28)          (select DATEITYPID
(29)              from BLOB)
(30)</cfquery>
(31)<cfquery name="DelErsteller" datasource="SunDB">
(32)  delete ERSTELLER
(33)      where ERSTELLERID not in
(34)          (select ERSTELLERID
(35)              from BLOB)
(36)</cfquery>
(37)<cfquery name="DelDigiGeraet" datasource="SunDB">
(38)  delete DIGITALISIERGERAT
(39)      where DIGITALISIERGERATID not in
(40)          (select DIGITALISIERGERATID
(41)              from BILD)
(42)</cfquery>
(43)<cfquery name="DelBildkategorie" datasource="SunDB">
(44)  delete BILDKATEGORIE
(45)      where BILDKATEGORIEID not in
(46)          (select BILDKATEGORIEID
(47)              from BILDKAT_BILD)
```

```
(48)</cfquery>
(49)<cfquery name="DelStichwort" datasource="SunDB">
(50)   delete STICHWORT
(51)       where STICHWORTID not in
(52)           (select STICHWORTID
(53)               from STICHWORT_BLOB)
(54)</cfquery>

(55)<cflocation url="Bild_frm.cfm">
```

## 4.6 Suchformular

Warum es für ein BLOB-Archivierungssystem wichtig ist, eine gute Suchfunktion zu bieten und nach welchen Entitäten eine Suche möglich sein sollte, wurde bereits in Kapitel 2.2 erläutert. Ein entsprechendes Suchformular wird in Abbildung 4-6 dargestellt.

The screenshot shows a web browser window titled "Spinnrad-Intranet: Suche nach Binary Large Objects - Netscape". The address bar shows "http://stamm/oldocs/dpl/Suche\_frm.cfm". The main content area is titled "Suche nach Binary Large Objects". It features a search form with the following elements:

- Stichwort:** A text input field containing "Eis;Haushaltsgeräte".
- Instructions:** "Für die Suche nach mehreren Stichwörtern, diese mit Semikolon getrennt eingeben. Als Wildcards können folgende Symbole verwendet werden:
  - Prozentzeichen (%): kein oder beliebig viele Zeichen
  - Unterstrich (\_): genau ein beliebiges Zeichen
- Dateityp:** A dropdown menu with options: "alle Dateitypen", "csv - application/x-unknown-content-type-Excel CSV", "gif - image/gif", "jpg - image/jpeg", "pdf - application/pdf".
- Ersteller:** A dropdown menu with options: "alle Ersteller", "Agentur ABC", "Design & Co GmbH", "Doe, John", "Intersok".
- Blob vom Typ:** Radio buttons for "Bild" (selected), "Text", and "andere BLOBs".
- Bildkategorie:** A dropdown menu with options: "alle Kategorien", "Kollage", "Landschaftsaufnahme", "Produktfoto", "Rohstoffherkunft".
- Buttons:** "Suche starten" and "Zurücksetzen".

Abbildung 4-6: Formular für die Suche nach BLOBs

In diesem Formular wird die Suche nach Stichwörtern, Dateitypen, Erstellern und der Art des BLOBs angeboten. Handelt es sich bei dem gesuchten BLOB um ein Bild, können zusätzlich eine oder mehrere Bildkategorien gewählt werden.

Die verwendeten Listboxen enthalten jeweils alle in der Datenbank vorliegenden Datensätze der jeweiligen Entität. Alle Listboxen lassen eine Mehrfachauswahl zu. Außerdem können in dem Feld für die Stichwörter mehrere Begriffe durch ein Semikolon getrennt eingegeben werden. Die Verwendung von sogenannten „Wildcards“ ist ebenfalls möglich. Das Prozentzeichen steht dabei als Platzhalter für kein bzw. beliebig viele Zeichen und der Unterstrich steht für genau ein beliebiges Zeichen.

Durch das Betätigen des Buttons „Suche starten“ wird der unten stehende CFML-Code ausgeführt, der für eine bessere Übersicht wieder unterteilt ist. Das Ergebnis der Suchanfrage wird in Abbildung 4-7 gezeigt. Im ersten Teil des CFML-Dokuments werden zunächst einige Variablen gesetzt, die zu einer SQL-Suchanfrage zusammengestellt werden, da der Benutzer des Suchformulars verschiedene Kriterien oder auch jeweils die Auswahl „alle ...“ benutzen kann. Die Variablennamen enthalten jeweils die entsprechende Klausel der SQL-Anweisung (z.B. wird `St_where` in die `WHERE`-Klausel eingebunden).

#### Suche.cfm

```
(1)<cfif form.Stichwort is ''>
(2)   <cfset St_where="1=1 ">
(3)   <cfset St_from="">
(4)<cfelse>
(5)   <cfset St_where="BLOB.BLOBID=STICHWORT_BLOB.BLOBID and
      STICHWORT_BLOB.STICHWORTID=STICHWORT.STICHWORTID and (">
(6)   <cfloop index="idxStichwort" list="#form.Stichwort#" delimiters=";">
(7)     <cfset St_where=St_where & '(STICHWORT.STICHWORT_DEUTSCH like
      "#idxStichwort#" ) or '>
(8)   </cfloop>
(9)   <cfset St_where=Left(St_where,(Len(St_where)-4)) & ">
(10)  <cfset St_from=" ,STICHWORT,STICHWORT_BLOB">
(11)</cfif>
```

In den Zeilen (1) bis (11) wird zunächst die Eingabe in das Stichwort-Feld ausgewertet. Wird in dieses Feld nichts eingetragen, so werden die entsprechenden Variablen ebenfalls leer gelassen bzw. mit einem „Dummy-Wert“ gesetzt, um keine falsche Syntax für die SQL-Anweisung zu erzeugen. Wurden vom Benutzer Suchwörter eingetragen, so werden diese in die SQL-Abfrage mit aufgenommen. Zunächst wird in Zeile (5) die Beziehung zwischen Stichwörtern und BLOBs

hergestellt. Die Schleife in den Zeilen (6) bis (8) hängt dann zusätzlich für jedes eingegebene Stichwort eine Abfragebedingung mit einer ODER-Verknüpfung an. In Zeile (9) wird das letzte OR in dem String `St_where` wieder entfernt. Schließlich werden in Zeile (10) noch die zugrundeliegenden Tabellen festgelegt.

```
(12)<cfset DT_where='and BLOB.DATEITYPID*=DATEITYP.DATEITYPID'>
(13)<cfif ParameterExists(form.Dateityp)>
(14)  <cfif form.Dateityp is not 0>
(15)      <cfset DT_where='and BLOB.DATEITYPID=DATEITYP.DATEITYPID and
DATEITYP.DATEITYPID in (' & #form.Dateityp# & ') '>
(16)  </cfif>
(17)</cfif>
```

Die Listbox `form.Dateityp` liefert eine kommasetrennte Liste von ID-Werten, die alle markierten Dateitypen enthält. Wird nur der Eintrag „alle Dateitypen“ aktiviert, enthält die Liste ausschließlich das Element „0“. In diesem Fall - oder wenn alle Einträge in der Liste deaktiviert sind - wird lediglich die Beziehung zwischen den Tabellen BLOB und DATEITYP in die WHERE-Klausel aufgenommen (Zeile (12)). Andernfalls wird in Zeile (15) zusätzlich die Bedingung eingefügt, daß alle DATEITYPID-Werte der Ergebnismenge in der eingegebenen Liste vorkommen müssen.

```
(18)<cfset Er_where='and BLOB.ERSTELLERID*=ERSTELLER.ERSTELLERID'>
(19)<cfif ParameterExists(form.Ersteller)>
(20)  <cfif form.Ersteller is not 0>
(21)      <cfset Er_where='and BLOB.ERSTELLERID=ERSTELLER.ERSTELLERID and
ERSTELLER.ERSTELLERID in (' & #form.Ersteller# & ') '>
(22)  </cfif>
(23)</cfif>
```

Die Vorgehensweise aus den Zeilen (12) bis (17) wiederholen sich in den Zeilen (18) bis (23) für die Listbox `form.Ersteller` und die Entität *Ersteller*.

```
(24) <cfset BK_from=" ">
(25) <cfset BK_where=" ">
(26)
(27) <cfif form.BlobTyp is "Bild">
(28)     <cfset BK_where=" and BLOB.BLOBID*=BILDKAT_BILD.BLOBID and
BILDKAT_BILD.BILDKATEGORIEID*=BILDKATEGORIE.BILDKATEGORIEID">
(29)     <cfif ParameterExists(form.Bildkategorie)>
```

```

(30)         <cfif form.Bildkategorie is not 0>
(31)             <cfset BK_where='and BLOB.BLOBID=BILDKAT_BILD.BLOBID and
                BILDKAT_BILD.BILDKATEGORIEID in (' & #form.Bildkategorie# & ' )'>
(32)         </cfif>
(33)     </cfif>

(34)     <cfset Typ_select=',BILD.*,DIGITALISIERGERAT'>
(35)     <cfset Typ_from=',BILD,DIGITALISIERGERAT'>
(36)     <cfset Typ_where='and BLOB.BLOBID=BILD.BLOBID and
                BILD.DIGITALISIERGERATID*=DIGITALISIERGERAT.DIGITALISIERGERATID'>
(37)     <cfset BK_from=",BILDKAT_BILD,BILDKATEGORIE">

(38) <cfelseif form.BlobTyp is "Text">
(39)     <cfset Typ_select=',TEXTDOKUMENT.*'>
(40)     <cfset Typ_from=',TEXTDOKUMENT'>
(41)     <cfset Typ_where='and BLOB.BLOBID=TEXTDOKUMENT.BLOBID'>

(42) <cfelse>
(43)     <cfset Typ_select=''>
(44)     <cfset Typ_from=''>
(45)     <cfset Typ_where='and BLOB.BLOBID not in (select BLOBID from BILD) and
                BLOB.BLOBID not in (select BLOBID from TEXTDOKUMENT)'>
(46) </cfif>

```

In den Zeilen (24) bis (46) wird das Selektionsgruppenfeld `form.BlobTyp` ausgewertet. Wurde in dem Feld der Punkt „Bild“ aktiviert, dann wird in den Zeilen (28) bis (33) zusätzlich noch die Listbox `form.Bildkategorie` verwendet, um analog zu den Vorgehensweisen bei *Ersteller* und *Dateityp*, den Teil der WHERE-Klausel zu setzen, der nach der entsprechenden Bildkategorie sucht.

Wurde vom Anwender der Punkt „Text“ bzw. „andere BLOBs“ gewählt, so werden in den Zeilen (39) bis (41) bzw. (43) bis (45) die Werte für die Suche nach Text-BLOBs bzw. anderen BLOBs gesetzt.

```

(47) <cfquery name="getBlobs" datasource="SunDB">
(48)     select BESCHREIBUNG, GROSSE, DATEIENDUNG, CONTENTTYPE, CONTENTSUBTYPE,
                BLOB.BLOBID, ERSTELLER #Typ_select#
(49)     from BLOB,DATEITYP,ERSTELLER #St_from# #BK_from# #Typ_from#
(50)     where #St_where# #DT_where# #Er_where# #BK_where# #Typ_where#
(51)     order by BESCHREIBUNG,ERSTELLER
(52) </cfquery>

```

Nachdem nun alle Werte aus den Eingabefeldern des Formulars in Teile der SQL-Anweisung umgesetzt wurden, wird in Zeile (47) bis (52) die SQL-Anweisung für die Suchanfrage zusammengesetzt und ausgeführt.

```
(53)<cf_head title="Suchergebnis">
(54)Ihre Suchanfrage brachte folgendes Ergebnis:<br>
(55)<p>
(56)<table border="1">
(57)<tr>
(58)<td valign="top"><b>ID</b></td>
(59)<td valign="top"><b>Beschreibung</b></td>
(60)<td valign="top"><b>Größe</b></td>
(61)<td valign="top"><b>Endung</b></td>
(62)<td valign="top"><b>Dateityp</b></td>
(63)<td valign="top"><b>Ersteller</b></td>
(64)<td valign="top"><b>Stichwörter</b></td>
(65)<cfif form.BlobTyp is "Bild">
(66)  <td valign="top"><b>Bildkategorien</b></td>
(67)  <td valign="top" align="center"><b>Bildgröße</b><br>B x H<br>(in Pixel)</td>
(68)  <td valign="top" align="center"><b>Farbtiefe</b><br>(in Bit)</td>
(69)  <td valign="top" align="center"><b>Farbsystem</b></td>
(70)  <td valign="top" align="center"><b>Aufnahmedatum</b></td>
(71)  <td valign="top" align="center"><b>Original</b></td>
(72)  <td valign="top"><b>Digitalisiergerät</b></td>
(73)<cfelseif form.BlobTyp is "Text">
(74)  <td valign="top"><b>Titel</b></td>
(75)  <td valign="top"><b>Erstelldatum</b></td>
(76)<cfelse>
(77)</cfif>
(78)<td valign="top"><b>Aktionen</b></td>
(79)</tr>
```

Ab Zeile (53) beginnt die Ausgabe des Suchergebnisses. Zunächst wird dafür wieder der Seitenkopf generiert (Zeile (53)) und anschließend die Tabelle mit der ersten Zeile für die Überschriften. Die Anzahl der Spalten hängt wiederum davon ab, welchen Punkt des Selektionsgruppenfeldes `form.BlobTyp` der Benutzer ausgewählt hat. In den Zeilen (66) bis (72) werden z.B. die Spaltenüberschriften erzeugt, wenn der Anwender nach Bildern sucht.

```
(80)<cfloop query="getBlobs">
(81)  <cfoutput>
```

```

(82)     <tr>
(83)     <td valign="top">#BLOBID#</td>
(84)     <td valign="top">#BESCHREIBUNG#</td>
(85)     <td valign="top">#GROSSE#</td>
(86)     <td valign="top">#DATEIENDUNG#</td>
(87)     <td valign="top">#CONTENTTYPE##/CONTENTSUBTYPE#</td>
(88)     <td valign="top">#ERSTELLER#</td>
(89) </cfoutput>
(90) <td valign="top">
(91)     <cfquery name="GetStichwort" datasource="SunDB">
(92)         select STICHWORT_DEUTSCH
(93)             from STICHWORT, STICHWORT_BLOB
(94)             where STICHWORT_BLOB.BLOBID=#getBlobs.BLOBID#
(95)                 and STICHWORT.STICHWORTID=STICHWORT_BLOB.STICHWORTID
(96)             order by STICHWORT_DEUTSCH
(97)     </cfquery>
(98)     <cfoutput query="GetStichwort">
(99)         <cfif ListContainsNoCase(form.Stichwort,STICHWORT_DEUTSCH,";")>
(100)             <b>#STICHWORT_DEUTSCH#<br></b>
(101)         <cfelse>
(102)             #STICHWORT_DEUTSCH#<br>
(103)         </cfif>
(104)     </cfoutput>
(105)     &nbsp;
(106) </td>

```

In Zeile (80) beginnt eine LOOP-Schleife, die für jeden Datensatz der Suchanfrage genau einmal durchlaufen wird. Danach werden zunächst alle Spalten ausgegeben, die die Suchanfrage zurückgegeben hat. In Zeile (91) bis (97) werden mit einer weiteren Abfrage alle Stichwörter aus der Datenbank gelesen, die zu dem jeweils gerade angezeigten BLOB gehören. Die Stichwörter, die auch in der Suchanfrage eingegeben wurden, werden in Fettdruck ausgegeben (Zeile (100)).

```

(107) <cfif form.BlobTyp is "Bild">
(108)     <td valign="top">
(109)         <cfquery name="GetBildkat" datasource="SunDB">
(110)             select BILDKATEGORIE.*
(111)                 from BILDKATEGORIE, BILDKAT_BILD
(112)                 where BILDKAT_BILD.BLOBID=#getBlobs.BLOBID#
(113)                     and
(114)                         BILDKATEGORIE.BILDKATEGORIEID=BILDKAT_BILD.BILDKATEGORIEID
(115)                 order by BILDKATEGORIE
(116)         </cfquery>
(116)     <cfoutput query="GetBildkat">

```

```

(117)         <cfif ListContains(form.Bildkategorie,BILDKATEGORIEID,";")>
(118)             <b>#BILDKATEGORIE#<br></b>
(119)         <cfelse>
(120)             #BILDKATEGORIE#<br>
(121)         </cfif>
(122)     </cfoutput>
(123)     &nbsp;
(124) </td>
(125) <cfoutput>
(126) <td valign="top" align="center">#BILDBREITE# x #BILDHOHE#</td>
(127) <td valign="top" align="center">#FARBTIEFE#&nbsp;</td>
(128) <td valign="top" align="center">#FARBSYSTEM#&nbsp;</td>
(129) <td valign="top" align="center">#AUFNAHME DATUM#&nbsp;</td>
(130) <td valign="top" align="center">
(131)     <cfif #ORIGINAL#>
(132)         ja
(133)     <cfelse>
(134)         nein
(135)     </cfif>
(136) </td>
(137) <td valign="top">#DIGITALISIERGERAT#&nbsp;</td>
(138) </cfoutput>
(139) <cfelseif form.BlobTyp is "Text">
(140)     <cfoutput>
(141)         <td valign="top">#TITEL#&nbsp;</td>
(142)         <td valign="top">#ERSTELLDATUM#&nbsp;</td>
(143)     </cfoutput>
(144) <cfelse>
(145) </cfif>

```

Falls der Benutzer nach Bildern gesucht hat, werden analog zu den Stichwörtern im vorherigen Abschnitt ab Zeile (107) die Bildkategorien zu jedem Bild und die restlichen Bildattribute ((Zeilen (126) bis (137))) ausgegeben. In den Zeilen (141) und (142) wird die Ausgabe der Attribute von Textdokumenten erzeugt, falls nach Texten gesucht wurde.

```

(146) <td align="center" valign="top">
(147)     <cfoutput>
(148)         <a href="Blob_view.cfm?ID=#BLOBID#">anzeigen/speichern</a><br>
(149)     <cfif form.BlobTyp is "Bild">
(150)         <a href="Bild_delete.cfm?ID=#BLOBID#">löschen</a><br>
(151)         <a href="Bild_update_frm.cfm?ID=#BLOBID#">ändern</a><br>
(152)     <cfelseif form.BlobTyp is "Text">
(153)         <a href="Text_delete.cfm?ID=#BLOBID#">löschen</a><br>

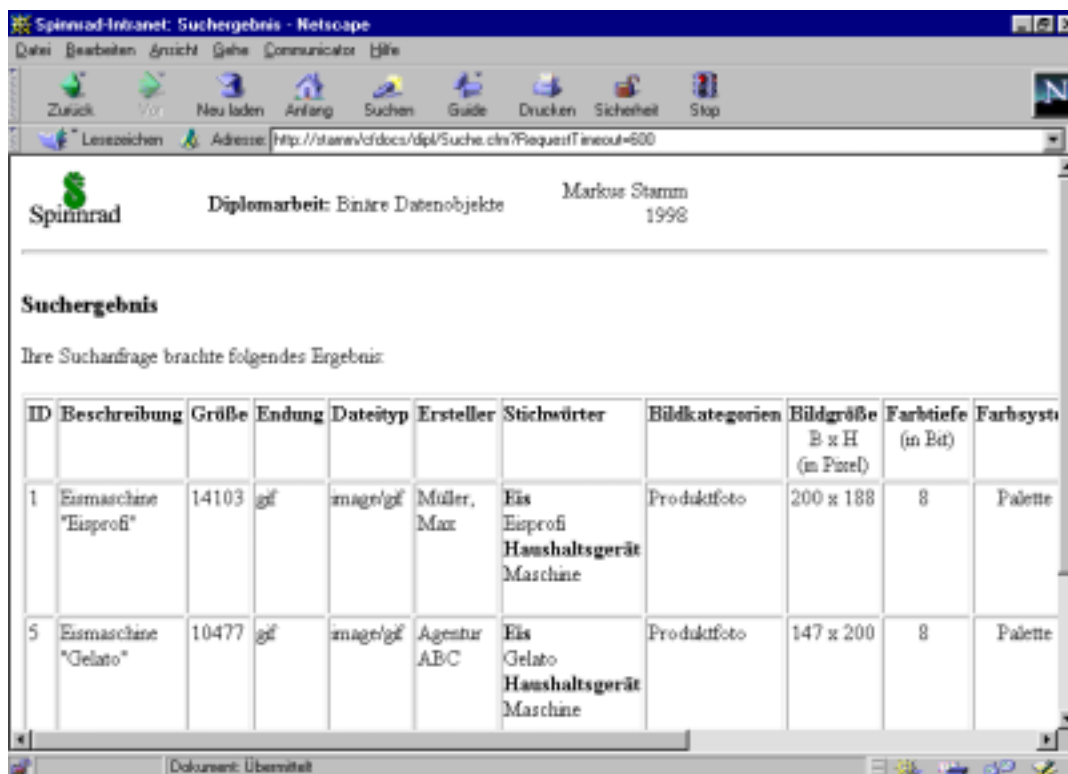
```

```

(154)             <a href="Text_update_frm.cfm?ID=#BLOBID#">ändern</a><br>
(155)             <cfelse>
(156)             <a href="Blob_delete.cfm?ID=#BLOBID#">löschen</a><br>
(157)             <a href="Blob_update_frm.cfm?ID=#BLOBID#">ändern</a><br>
(158)             </cfif>
(159)             </cfoutput>
(160)            </td>
(161)            </tr>
(162)            </cfloop>
(163)            </table>
(164)
(165)            <cf_foot>

```

Die Ausgabe in der Spalte „Aktionen“ wird ab Zeile (146) erzeugt. Auch hier wird wieder je nach Auswahl des BLOB-Typs unterschieden, um die Links auf die entsprechenden Anwendungsseiten für die jeweilige Art der angezeigten BLOBs zu generieren. In Zeile (161) wird die ausgegebene Tabellenzeile schließlich beendet und in Zeile (162) wird das CFLOOP-Tag geschlossen. Nach der kompletten Abarbeitung der Schleife wird in Zeile (165) noch der Seitenfuß erzeugt.



Spinnrad Diplomarbeit: Binaire Datenobjekte Markus Stamm 1998

**Suchergebnis**

Ihre Suchanfrage brachte folgendes Ergebnis:

ID	Beschreibung	Größe	Endung	Dateityp	Ersteller	Stichwörter	Bildkategorien	Bildgröße B x H (in Pixel)	Farbtiefe (in Bit)	Farbsyst.
1	Eismaschine "Eisprofi"	14103	gif	image/gif	Müller, Max	Eis Eisprofi Haushaltsgerät Maschine	Produktfoto	200 x 188	8	Palette
5	Eismaschine "Gelato"	10477	gif	image/gif	Agentur ABC	Eis Gelato Haushaltsgerät Maschine	Produktfoto	147 x 200	8	Palette

Abbildung 4-7: Beispiel für ein Suchergebnis

In Abbildung 4-7 wird als Beispiel das Ergebnis einer Suchanfrage dargestellt.



## 5 Schlußbemerkungen

Das in dieser Arbeit vorgestellte System zur Archivierung von Binary Large Objects und die Anbindung an ein Intranet vermittelt einen ersten Eindruck von den Möglichkeiten, die durch das Zusammenführen von multimedialen Daten, Datenbanken und Internettechnologie erschlossen werden. Natürlich erhebt dieses System nicht den Anspruch auf Vollständigkeit - es zeigt lediglich die Chancen auf.

Um aus dem vorgestellten Prototypen ein praxisgerechtes Archivierungssystem zu entwickeln, bedarf es allerdings noch einiger Verbesserungen. Zum einen müssen die gezeigten Formulare auf ihre Ergonomie hin überprüft und verbessert werden. Bisher wurde lediglich darauf Wert gelegt, die geforderte Funktionalität umzusetzen. Ergonomische Gesichtspunkte wurden dabei vernachlässigt. Aufgrund der Eigenschaften von HTML als Beschreibungssprache, die es letztendlich der Client-Anwendung überläßt, wie die jeweiligen Elemente dargestellt werden, ist eine entsprechende Gestaltung der Formulare schwieriger als bei Programmiersprachen, wie z.B. Visual Basic, Delphi oder Visual C++, mit denen direkt grafische Benutzeroberflächen erstellt werden können. Hinzu kommt die oft sehr unterschiedliche Art der Anzeige von HTML-Elementen in verschiedenen Webbrowsern und unter verschiedenen Betriebssystemen. Auch im Bereich der Benutzerführung könnten noch Verbesserungen erzielt werden, indem umfangreiche Formulare in mehrere Teilformulare aufgeteilt werden.

Um die einzelnen Schritte der Vorgehensweise besser erläutern zu können, wurden bisher sämtliche Datenbankabfragen in einzelnen SQL-Statements hintereinander ausgeführt. Zur Steigerung der Performance des Systems, sollten für bestimmte Arbeitsschritte oder immer wieder aufeinander folgende Abfragen, Stored Procedures verwendet werden. Der Geschwindigkeitzuwachs wird dadurch erzielt, daß die SQL-Abfragen in den Stored Procedures bereits vorkompiliert sind und nur noch durch die übergebenen Parameter gesteuert werden. Eine Prüfung und Übersetzung des SQL-Codes ist deshalb nicht mehr notwendig.

Trotz des erhöhten Aufwands für die Gestaltung ergonomischer Formulare erweist sich die Verwendung von Webbrowsern als Clients als gute und vor allem günstige Lösung, da es inzwischen nahezu für jedes Betriebssystem auf jeder Hardwareplattform kostenlose Webbrowser gibt. Teure Client-Applikationen sind

---

somit nicht mehr nötig. Außerdem wird nicht nur den eigenen Mitarbeitern über das Intranet der Zugang zur firmeneigenen Datenbank ermöglicht. Auch dem Kunden wird der Weg über das Internet zu Produkt- und Firmeninformationen in multimedialer Form auf einfache Art und Weise geebnet.

## Anhang

### A Datentypen im Power Designer und Sybase ASE

Die folgenden Tabellen enthalten nur die für diese Arbeit relevanten Datentypen. Weitere Datentypen werden in [Powe97a] und [Syba97c] beschrieben.

#### Zahlen- und Bit-Datentypen

Power Designer Data Architect	Abk.	Sybase ASE	Wertebereich	Speicherkapazität in Byte
Byte 1	BT1	tinyint	0 bis 255	1
Integer	I	int/integer	$-2^{31}$ bis $2^{31}-1$	4
Number m,n	Nm,n	numeric(m,n)	$-10^m$ bis $10^m-1$	2 bis 17
Serial n	Sn	numeric(n,0)*	$-10^n$ bis $10^n-1$	
Boolean	BL	bit	0 oder 1	1 (ein Byte enthält bis zu 8 bit-Spalten)

Der Datentyp Serial n wird in numeric(n,0) mit der Integritätsbedingung identity übersetzt, d.h. die so deklarierte Spalte wird automatisch mit dem nächst höheren Wert belegt.

#### String- und Datums-Datentypen

Power Designer Data Architect	Abk.	Sybase ASE	Wertebereich	Speicherkapazität in Byte
Characters n	An	char(n)	255 Zeichen oder weniger	n
Variable Characters n	Van	varchar(n)	255 Zeichen oder weniger	tatsächliche Eintragslänge
Date	D	datetime	1. Januar 1753 bis 31. Dezember 9999	8

**Binär-Datentypen**

<b>Power Designer Data Architect</b>	<b>Abk.</b>	<b>Sybase ASE</b>	<b>Wertebereich</b>	<b>Speicherkapazität in Byte</b>
Binary n	BINn	binary(n)	bis zu 255 Byte	n
Long Binary n	LBINn	varbinary(n)	bis zu 255 Byte	tatsächliche Eintragslänge
Picture	Pic	image	bis zu $2^{31}-1$ Byte	0 bis zur Initialisierung, dann ein mehrfaches von 2 Kbyte

## B SQL-Statements zum Erzeugen der Datenbank

```

(1)  /* ===== */
(2)  /* Database name:  BINARDATENBANK */
(3)  /* DBMS name:     Sybase SQL Server 11 */
(4)  /* Created on:    20.08.98 09:58 */
(5)  /* ===== */

(6)  execute sp_addtype T_ARTNR, 'char(5)', 'null'
(7)  go

(8)  execute sp_addtype T_PIXELANZAHL, 'int', 'null'
(9)  go

(10) create rule R_PIXELANZAHL
(11) as
(12) @PIXELANZAHL >= 0
(13) go

(14) execute sp_bindrule R_PIXELANZAHL, T_PIXELANZAHL
(15) go

(16) execute sp_addtype T_ID, 'numeric(10)', 'not null'
(17) go

(18) execute sp_addtype T_TEXT_LANG, 'varchar(200)', 'null'
(19) go

(20) execute sp_addtype T_TEXT_KURZ, 'varchar(50)', 'null'
(21) go

(22) /* ===== */
(23) /* Table: ARTIKELDUMMY */
(24) /* ===== */
(25) create table ARTIKELDUMMY
(26) (
(27) ARTIKELNR          T_ARTNR          not null,
(28) ARTIKELNAME       T_TEXT_KURZ      null,
(29) constraint PK_ARTIKELDUMMY primary key (ARTIKELNR)
(30) )
(31) go

(32) /* ===== */
(33) /* Table: DIGITALISIERGERAT */
(34) /* ===== */
(35) create table DIGITALISIERGERAT
(36) (
(37) DIGITALISIERGERATID T_ID          identity,
(38) DIGITALISIERGERAT  T_TEXT_KURZ    not null,
(39) constraint PK_DIGITALISIERGERAT primary key (DIGITALISIERGERATID)
(40) )
(41) go

(42) /* ===== */
(43) /* Table: ERSTELLER */
(44) /* ===== */
(45) create table ERSTELLER
(46) (
(47) ERSTELLERID        T_ID          identity,
(48) ERSTELLER          T_TEXT_KURZ    not null,
(49) constraint PK_ERSTELLER primary key (ERSTELLERID)
(50) )
(51) go

(52) /* ===== */
(53) /* Table: BILDKATEGORIE */
(54) /* ===== */
(55) create table BILDKATEGORIE
(56) (
(57) BILDKATEGORIEID    T_ID          identity,
(58) BILDKATEGORIE      T_TEXT_KURZ    not null,
(59) constraint PK_BILDKATEGORIE primary key (BILDKATEGORIEID)
(60) )
(61) go

(62) /* ===== */
(63) /* Table: DATEITYP */
(64) /* ===== */
(65) create table DATEITYP
(66) (
(67) DATEITYPID          T_ID          identity,

```

```
(68) DATEIENDUNG          char(4)          not null,
(69) CONTENTTYPE        T_TEXT_KURZ    not null,
(70) CONTENTSUBTYPE     T_TEXT_KURZ    not null,
(71) DT_BESCHREIBUNG   T_TEXT_LANG    null      ,
(72) constraint PK_DATEITYP primary key (DATEITYPID)
(73) )
(74) go

(75) /* ===== */
(76) /* Table: STICHWORT */
(77) /* ===== */
(78) create table STICHWORT
(79) (
(80) STICHWORTID          T_ID            identity,
(81) STICHWORT_DEUTSCH   T_TEXT_KURZ    not null,
(82) constraint PK_STICHWORT primary key (STICHWORTID)
(83) )
(84) go

(85) /* ===== */
(86) /* Table: BLOB */
(87) /* ===== */
(88) create table BLOB
(89) (
(90) BLOBID              T_ID            identity,
(91) DATEITYPID          T_ID            not null,
(92) BINARDATEN         image          null      ,
(93) BESCHREIBUNG      T_TEXT_LANG    null      ,
(94) GROSSE             int            null      ,
(95) constraint CKC_GROSSE_BLOB check (GROSSE >= 0),
(96) ERSTELLERID       T_ID            null      ,
(97) constraint PK_BLOB primary key (BLOBID)
(98) )
(99) go

(100) /* ===== */
(101) /* Index: BLOB_ERSTELLER_FK */
(102) /* ===== */
(103) create index BLOB_ERSTELLER_FK on BLOB (ERSTELLERID)
(104) go

(105) /* ===== */
(106) /* Index: BLOB_DATEITYP_FK */
(107) /* ===== */
(108) create index BLOB_DATEITYP_FK on BLOB (DATEITYPID)
(109) go

(110) /* ===== */
(111) /* Table: BILD */
(112) /* ===== */
(113) create table BILD
(114) (
(115) BLOBID              T_ID            not null,
(116) DIGITALISIERGERATID T_ID            null      ,
(117) BILDHOE             T_PIXELANZAHL null      ,
(118) BILDBREITE         T_PIXELANZAHL null      ,
(119) FARBTIEFE          tinyint         null      ,
(120) FARBSYSTEM         T_TEXT_KURZ    null      ,
(121) AUFNAHME DATUM     datetime        null      ,
(122) ORIGINAL           bit            default 1 not null,
(123) constraint PK_BILD primary key (BLOBID)
(124) )
(125) go

(126) /* ===== */
(127) /* Index: DIGIGERAT_BBILD_FK */
(128) /* ===== */
(129) create index DIGIGERAT_BBILD_FK on BILD (DIGITALISIERGERATID)
(130) go

(131) /* ===== */
(132) /* Table: PRODUKTFOTO */
(133) /* ===== */
(134) create table PRODUKTFOTO
(135) (
(136) BLOBID              T_ID            not null,
(137) BLICKRICHTUNG      T_TEXT_KURZ    null      ,
(138) FREIGESTELLT       bit            default 0 not null,
(139) PRODUKTAUSRICHTUNG T_TEXT_KURZ    null      ,
(140) constraint PK_PRODUKTFOTO primary key (BLOBID)
(141) )
(142) go
```

```
(143) /* ===== */
(144) /* Table: TEXTDOKUMENT */
(145) /* ===== */
(146) create table TEXTDOKUMENT
(147) (
(148) BLOBID                T_ID                not null,
(149) TITEL                 T_TEXT_LANG          null      ,
(150) ERSTELLDATUM          datetime           null      ,
(151) constraint PK_TEXTDOKUMENT primary key (BLOBID)
(152) )
(153) go

(154) /* ===== */
(155) /* Table: ART_PRODFOTO */
(156) /* ===== */
(157) create table ART_PRODFOTO
(158) (
(159) BLOBID                T_ID                not null,
(160) ARTIKELNR            T_ARTNR            not null,
(161) constraint PK_ART_PRODFOTO primary key (BLOBID, ARTIKELNR)
(162) )
(163) go

(164) /* ===== */
(165) /* Index: PF_STELLT_A_DAR_FK */
(166) /* ===== */
(167) create index PF_STELLT_A_DAR_FK on ART_PRODFOTO (BLOBID)
(168) go

(169) /* ===== */
(170) /* Index: A_WIRD_AUF_PF_DARGESTELLT_FK */
(171) /* ===== */
(172) create index A_WIRD_AUF_PF_DARGESTELLT_FK on ART_PRODFOTO (ARTIKELNR)
(173) go

(174) /* ===== */
(175) /* Table: BILDKAT_BILD */
(176) /* ===== */
(177) create table BILDKAT_BILD
(178) (
(179) BILDKATEGORIEID      T_ID                not null,
(180) BLOBID                T_ID                not null,
(181) constraint PK_BILDKAT_BILD primary key (BILDKATEGORIEID, BLOBID)
(182) )
(183) go

(184) /* ===== */
(185) /* Index: BK_ENTHALT_BB_FK */
(186) /* ===== */
(187) create index BK_ENTHALT_BB_FK on BILDKAT_BILD (BILDKATEGORIEID)
(188) go

(189) /* ===== */
(190) /* Index: BB_GEHORT_ZU_BK_FK */
(191) /* ===== */
(192) create index BB_GEHORT_ZU_BK_FK on BILDKAT_BILD (BLOBID)
(193) go

(194) /* ===== */
(195) /* Table: STICHWORT_BLOB */
(196) /* ===== */
(197) create table STICHWORT_BLOB
(198) (
(199) STICHWORTID          T_ID                not null,
(200) BLOBID                T_ID                not null,
(201) constraint PK_STICHWORT_BLOB primary key (STICHWORTID, BLOBID)
(202) )
(203) go

(204) /* ===== */
(205) /* Index: SW_BESCHREIBT_BLOB_FK */
(206) /* ===== */
(207) create index SW_BESCHREIBT_BLOB_FK on STICHWORT_BLOB (STICHWORTID)
(208) go

(209) /* ===== */
(210) /* Index: BLOB_WIRD_BESCHR_D_SW_FK */
(211) /* ===== */
(212) create index BLOB_WIRD_BESCHR_D_SW_FK on STICHWORT_BLOB (BLOBID)
(213) go

(214) alter table BLOB
(215) add constraint FK_BLOB_BLOB_DATE_DATEITYP foreign key (DATEITYPID)
```

```
(216) references DATEITYP (DATEITYPID)
(217) go

(218) alter table BILD
(219) add constraint FK_BILD_DIGIGERAT_DIGITALI foreign key (DIGITALISIERGERATID)
(220) references DIGITALISIERGERAT (DIGITALISIERGERATID)
(221) go

(222) alter table BILD
(223) add constraint FK_BILD_BLOB_BILD_BLOB foreign key (BLOBID)
(224) references BLOB (BLOBID)
(225) go

(226) alter table PRODUKTFOTO
(227) add constraint FK_PRODUKTF_BILD_PROD_BILD foreign key (BLOBID)
(228) references BILD (BLOBID)
(229) go

(230) alter table TEXTDOKUMENT
(231) add constraint FK_TEXTDOKU_TD_BLOB_BLOB foreign key (BLOBID)
(232) references BLOB (BLOBID)
(233) go

(234) alter table ART_PRODFOTO
(235) add constraint FK_ART_PROD_PF_STELLT_PRODUKTF foreign key (BLOBID)
(236) references PRODUKTFOTO (BLOBID)
(237) go

(238) alter table ART_PRODFOTO
(239) add constraint FK_ART_PROD_A_WIRD_AU_ARTIKELD foreign key (ARTIKELNR)
(240) references ARTIKELDUMMY (ARTIKELNR)
(241) go

(242) alter table BILDKAT_BILD
(243) add constraint FK_BILDKAT__BK_ENTHAL_BILDKATE foreign key (BILDKATEGORIEID)
(244) references BILDKATEGORIE (BILDKATEGORIEID)
(245) go

(246) alter table BILDKAT_BILD
(247) add constraint FK_BILDKAT__BB_GEHORT_BILD foreign key (BLOBID)
(248) references BILD (BLOBID)
(249) go

(250) alter table STICHWORT_BLOB
(251) add constraint FK_STICHWOR_SW_BESCHR_STICHWOR foreign key (STICHWORTID)
(252) references STICHWORT (STICHWORTID)
(253) go

(254) alter table STICHWORT_BLOB
(255) add constraint FK_STICHWOR_BLOB_WIRD_BLOB foreign key (BLOBID)
(256) references BLOB (BLOBID)
(257) go
```

## C Cold Fusion-Datenbearbeitungsformulare

### 1 Artikel\_delete.cfm

```
(1) <cfquery name="DelArt_ProdFoto" datasource="SunDB">
(2)   delete ART_PRODFOTO
(3)     where ARTIKELNR='#url.id#'
(4) </cfquery>
(5)
(6) <cfquery name="DelArtikel" datasource="SunDB">
(7)   delete ARTIKELDUMMY
(8)     where ARTIKELNR='#url.id#'
(9) </cfquery>
(10)
(11) <cflocation url="Artikel_frm.cfm">
```

### 2 Artikel\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Artikel" datasource="SunDB">
(4)   select ARTIKELDUMMY.*,BLOB.BESCHREIBUNG
(5)   from ARTIKELDUMMY,ART_PRODFOTO,BLOB
(6)   where ARTIKELDUMMY.ARTIKELNR*=ART_PRODFOTO.ARTIKELNR
(7)         and ART_PRODFOTO.BLOBID*=BLOB.BLOBID
(8)   order by ARTIKELNAME
(9) </cfquery>
(10)
(11) <cf_head title="Daten anzeigen: Artikel">
(12)
(13) <table border="1">
(14) <tr>
(15) <td valign="top"><b>ArtNr</b></td>
(16) <td valign="top"><b>Artikel</b></td>
(17) <td valign="top"><b>Benutzt für Bild</b></td>
(18) <td valign="top"><b>Aktionen</b></td>
(19) </tr>
(20) <cfoutput query="Artikel" group="ARTIKELNR">
(21) <tr>
(22) <td valign="top">#ARTIKELNR#</td>
(23) <td valign="top">#ARTIKELNAME#&nbsp;</td>
(24) <td valign="top">
(25)   <cfoutput>#BESCHREIBUNG#<br></cfoutput>
(26) </td>
(27) <td align="center" valign="top">
(28)   <a href="Artikel_delete.cfm?ID=#ARTIKELNR#">löschen</a><br>
(29)   <a href="Artikel_update_frm.cfm?ID=#ARTIKELNR#">ändern</a><br>
(30) </td>
(31) </tr>
(32) </cfoutput>
(33) </table>
(34) <p>
(35) <cf_foot>
```

### 3 Artikel\_update.cfm

```
(1) <cfquery name="UpdateArtikel" datasource="SunDB">
(2)   update ARTIKELDUMMY
(3)     set ARTIKELNAME='#form.Artikel#'
(4)     where ARTIKELNR='#form.ArtikelNr#'
(5) </cfquery>
(6)
(7) <cflocation url="Artikel_frm.cfm?RequestTimeout=120">
```

### 4 Artikel\_update\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Artikel" datasource="SunDB">
(4)   select *
(5)   from ARTIKELDUMMY
(6)   where ARTIKELNR='#url.id#'
```

```

(7) </cfquery>
(8)
(9) <cf_head title="Daten ändern: Artikel">
(10)
(11) <cfoutput query="Artikel">
(12) <form action="Artikel_update.cfm" method="POST">
(13) <input type="Hidden" name="ArtikelNr" value="#ARTIKELNR#">
(14) <table border="1">
(15) <tr>
(16) <td valign="top"><b>ArtNr</b></td>
(17) <td valign="top"><b>Artikel</b></td>
(18) </tr>
(19) <tr>
(20) <td valign="top">#ARTIKELNR#</td>
(21) <td valign="top">
(22) <input type="Text" name="Artikel" value="#ARTIKELNAME#" maxlength="50"
    size="50">
(23) </td>
(24) </tr>
(25) </table>
(26) <p>
(27) <input type="Submit" value="Ändern">
(28) <input type="reset" value="Zurücksetzen">
(29) </form>
(30) </cfoutput>
(31) <p>
(32) <cf_foot>

```

## 5 ArtProfoto\_update.cfm

```

(1) <cfloop index="ArtNr" list="#form.delArtikel#" delimiters=",">
(2) <cfquery name="DelArtProfoto" datasource="SunDB">
(3) delete ART_PRODFOTO
(4) where BLOBID=#form.BlobID#
(5) and ARTIKELNR='#ArtNr#'
(6) </cfquery>
(7) </cfloop>
(8)
(9) <cfloop index="ArtNr" list="#form.ArtikelNr#" delimiters=",">
(10) <cfquery name="InsArtProfoto" datasource="SunDB">
(11) insert into ART_PRODFOTO
(12) values (#form.BlobID#, '#ArtNr#')
(13) </cfquery>
(14) </cfloop>
(15)
(16) <!-- Ausgabe --->
(17) <cflocation url="Produktfoto_frm.cfm?ID=#form.BlobID#">

```

## 6 ArtProfoto\_update\_frm.cfm

```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="GetBlob" datasource="SunDB">
(4) select BLOB.*
(5) from BLOB
(6) where BLOBID=#url.ID#
(7) </cfquery>
(8)
(9) <cfquery name="GetAlleArtikel" datasource="SunDB">
(10) select ARTIKELDUMMY.*
(11) from ARTIKELDUMMY
(12) where ARTIKELNR not in
(13) (select ARTIKELNR
(14) from ART_PRODFOTO
(15) where BLOBID=#url.ID#)
(16) order by ARTIKELNR
(17) </cfquery>
(18)
(19) <cfquery name="GetZugeordArtikel" datasource="SunDB">
(20) select ARTIKELDUMMY.*
(21) from ARTIKELDUMMY,ART_PRODFOTO
(22) where BLOBID=#url.ID#
(23) and ARTIKELDUMMY.ARTIKELNR=ART_PRODFOTO.ARTIKELNR
(24) order by ARTIKELDUMMY.ARTIKELNR
(25) </cfquery>
(26)
(27) <cf_head title="Daten ändern: Artikel">
(28)
(29) <cfoutput query="GetBlob">

```

```

(30)      <b>#BESCHREIBUNG# (#BLOBID#)</b><br>
(31) </cfoutput>
(32)
(33) <form action="ArtProdfoto_update.cfm" method="POST">
(34) <cfoutput>
(35) <input type="Hidden" name="BlobID" value="#url.ID#">
(36) </cfoutput>
(37) <p>
(38) <b>Zugeordnete Artikel:</b><br>
(39) <table border="1">
(40) <tr>
(41)     <td>Art.-Nr.</td>
(42)     <td>Artikel</td>
(43)     <td>Zuordnung entfernen?</td>
(44) </tr>
(45) <cfoutput query="GetZugeordArtikel">
(46) <tr>
(47)     <td>#ARTIKELNR#</td>
(48)     <td>#ARTIKELNAME#</td>
(49)     <td><input type="Checkbox" name="delArtikel" value="#ARTIKELNR#"></td>
(50) </tr>
(51) </cfoutput>
(52) </table>
(53) <p>
(54) <b>Artikel neu zuordnen:</b><br>
(55) <select name="ArtikelNr" multiple size="10">
(56)     <cfoutput query="GetAlleArtikel">
(57)         <option value="#ARTIKELNR#">#ARTIKELNR# - #ARTIKELNAME#</option>
(58)     </cfoutput>
(59) </select>
(60) <p>
(61) <input type="Submit" value="Übernehmen">
(62) <input type="reset" value="Zurücksetzen">
(63) </form>
(64)
(65) <cf_foot>

```

## 7 Bild\_delete.cfm

Siehe Kapitel 0.

## 8 Bild\_frm.cfm

Siehe Kapitel 4.1.

## 9 Bild\_insert.cfm

Siehe Kapitel 4.3.

## 10 Bild\_insert\_frm.cfm

```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="GetErsteller" datasource="SunDB">
(4)     select *
(5)     from ERSTELLER
(6)     order by ERSTELLER
(7) </cfquery>
(8)
(9) <cfquery name="GetDigiGeraet" datasource="SunDB">
(10)    select *
(11)    from DIGITALISIERGERAT
(12)    order by DIGITALISIERGERAT
(13) </cfquery>
(14)
(15) <cfquery name="GetStichwort" datasource="SunDB">
(16)    select *
(17)    from STICHWORT
(18)    order by STICHWORT_DEUTSCH
(19) </cfquery>
(20)

```

```
(21) <cfquery name="GetKategorie" datasource="SunDB">
(22)     select *
(23)     from BILDKATEGORIE
(24)     order by BILDKATEGORIE
(25) </cfquery>
(26)
(27) <cf_head title="Daten einfügen: Bild">
(28)
(29) <FORM ACTION="Bild_insert.cfm?RequestTimeout=600" ENCTYPE="multipart/form-data"
METHOD="post">
(30) <p>Bitte wählen Sie eine Bilddatei aus, die Sie in der Datenbank speichern
möchten:
(31) <p><INPUT NAME="BINARDATEN" size="50" TYPE="file">
(32) <p>Geben Sie der Datei einen Titel oder kurze Beschreibung:
(33) <p><INPUT NAME="Beschreibung" size="50" TYPE="text">
(34) <p>
(35) <table width="600">
(36) <tr>
(37)     <td>Bildbreite* (in Pixel): <INPUT NAME="Bildbreite" size="10"
TYPE="text"></td>
(38)     <td>Bildhöhe* (in Pixel): <INPUT NAME="Bildhoehe" size="10"
TYPE="text"></td>
(39) </tr>
(40) <tr>
(41)     <td colspan="2">
(42)         *<font size="-1">Falls es sich bei dem Bild um ein Compuserve Bitmap
(gif) oder JPEG-Bitmap (jpg) handelt, brauchen Sie die Felder 'Bildhöhe' und
'Bildbreite' nicht ausfüllen.</font>
(43)     </td>
(44) </tr>
(45) <tr>
(46)     <td>Farbtiefe (in Bit): <INPUT NAME="Farbtiefe" size="10" TYPE="text"></td>
(47)     <td>Farbsystem:
(48)         <select name="Farbsystem">
(49)             <cfloop index="idxFarbsystem" list="#lstFarbsystem#"
delimiters=", ">
(50)                 <cfoutput>
(51)                     <option value="#idxFarbsystem#">#idxFarbsystem#
(52)                 </cfoutput>
(53)             </cfloop>
(54)         </select>
(55)     </td>
(56) </tr>
(57) <tr>
(58)     <td>Aufnahmedatum (tt.mm.jjjj): <INPUT NAME="Tag" size="2" maxlength="2"
TYPE="text">.<INPUT NAME="Monat" size="2" maxlength="2" TYPE="text">.<INPUT
NAME="Jahr" size="4" maxlength="4" TYPE="text" value="19"></td>
(59)     <td>Original (oder bearbeitet): <input type="checkbox" name="Original"
checked></td>
(60) </tr>
(61) </table>
(62) <p>
(63) <table width="600">
(64) <tr>
(65)     <td valign="top" width="50%">
(66)         Ersteller:<br>
(67)         <select name="Ersteller" size="5">
(68)             <cfoutput query="GetErsteller">
(69)                 <option value="#ERSTELLERID#">#ERSTELLER#
(70)             </cfoutput>
(71)         </select>
(72)     </td>
(73)     <td valign="top" width="50%">
(74)         Neuer Ersteller:<br>
(75)         <input type="Text" name="Ersteller_neu" maxlength="50" size="30">
(76)     </td>
(77) </tr>
(78) </table>
(79) <p>
(80) <table width="600">
(81) <tr>
(82)     <td valign="top" width="50%">
(83)         Digitalisiergerät,<br>mit dem das Bild erstellt wurde:<br>
(84)         <select name="DigiGeraet" size="5">
(85)             <cfoutput query="GetDigiGeraet">
(86)                 <option value="#DIGITALISIERGERATID#">#DIGITALISIERGERAT#
(87)             </cfoutput>
(88)         </select>
(89)     </td>
(90)     <td valign="top" width="50%">
(91)         Neues Digitalisiergerät:<br>
(92)         <input type="Text" name="DigiGeraet_neu" maxlength="50" size="30">
(93)     </td>
```

```

(94) </tr>
(95) </table>
(96) <p>
(97) <table width="600">
(98) <tr>
(99)     <td valign="top" width="50%">
(100)         Stichworte:<br>
(101)         <select name="Stichwort" multiple size="5">
(102)             <cfoutput query="GetStichwort">
(103)                 <option value="#STICHWORTID#">#STICHWORT_DEUTSCH#
(104)             </cfoutput>
(105)         </select>
(106)     </td>
(107)     <td valign="top" width="50%">
(108)         Neue Stichwörter<br>
(109)         <font size="-1">(mehrere Stichwörter mit Semikolon getrennt
eingeben)</font>:<br>
(110)         <input type="Text" name="Stichwort_neu" maxlength="250" size="30">
(111)     </td>
(112) </tr>
(113) </table>
(114) <p>
(115) <table width="600">
(116) <tr>
(117)     <td valign="top" width="50%">
(118)         Bildkategorie:<br>
(119)         <select name="Kategorie" multiple size="5">
(120)             <cfoutput query="GetKategorie">
(121)                 <option value="#BILDKATEGORIEID#">#BILDKATEGORIE#
(122)             </cfoutput>
(123)         </select>
(124)     </td>
(125)     <td valign="top" width="50%">
(126)         Neue Kategorie<br>
(127)         <font size="-1">(mehrere Kategorien mit Semikolon getrennt
eingeben)</font>:<br>
(128)         <input type="Text" name="Kategorie_neu" maxlength="250" size="30">
(129)     </td>
(130) </tr>
(131) </table>
(132) <p>
(133) <table width="600">
(134) <tr>
(135)     <td><INPUT TYPE="submit" VALUE="Datei hochladen"></td>
(136)     <td><input type="reset" value="Zurücksetzen"></td>
(137) </tr>
(138) </table>
(139)
(140) </FORM>
(141) <p>Der Hochladevorgang kann je nach Größe der Binärdatei bis zu 5 Minuten
dauern.
(142)
(143) <cf_foot>

```

## 11 Bild\_update.cfm

```

(1) <!-- aus Liste gewählter Ersteller -->
(2) <cfif ParameterExists(form.Ersteller)>
(3)     <cfset setBlob=",ERSTELLERID=" & form.Ersteller>
(4) <cfelseif form.Ersteller_neu is "">
(5)     <!-- kein Ersteller -->
(6)     <cfset setBlob="">
(7) <cfelse>
(8)     <!-- neuen Ersteller in die DB einfügen -->
(9)     <cfquery name="InsertErsteller" datasource="SunDB">
(10)         insert into ERSTELLER (ERSTELLER)
(11)             select '#form.Ersteller_neu#'
(12)             where not exists
(13)                 (select *
(14)                  from ERSTELLER
(15)                  where ERSTELLER='#form.Ersteller_neu#')
(16)     </cfquery>
(17)     <cfquery name="GetErID" datasource="SunDB">
(18)         select @@identity as ErID
(19)     </cfquery>
(20)     <cfoutput query="GetErID">
(21)         <cfset setBlob=",ERSTELLERID=" & #ErID#>
(22)     </cfoutput>
(23) </cfif>
(24)
(25) <cfquery name="UpdateBlob" datasource="SunDB">

```

```
(26)     update BLOB
(27)         set BESCHREIBUNG='#form.Beschreibung_neu#'
(28)         #setBlob#
(29)         where BLOBID=#url.ID#
(30) </cfquery>
(31)
(32) <cfquery name="DelErsteller" datasource="SunDB">
(33)     delete ERSTELLER
(34)         where ERSTELLERID not in
(35)             (select ERSTELLERID
(36)                 from BLOB)
(37) </cfquery>
(38)
(39) <cfif ParameterExists(form.delStichwort)>
(40)     <cfquery name="DelStichw_Blob" datasource="SunDB">
(41)         delete STICHWORT_BLOB
(42)             where BLOBID=#url.ID#
(43)                 and STICHWORTID in (#form.delStichwort#)
(44)     </cfquery>
(45)
(46)     <cfquery name="DelStichwort" datasource="SunDB">
(47)         delete STICHWORT
(48)             where STICHWORTID not in
(49)                 (select STICHWORTID
(50)                     from STICHWORT_BLOB)
(51)     </cfquery>
(52) </cfif>
(53)
(54) <!-- neue Stichworte in Tabellen einfügen --->
(55) <cfloop index="idxStichwort" list="#Stichwort_neu#" delimiters=";">
(56)     <cfquery name="InsertStichwort" datasource="SunDB">
(57)         insert into STICHWORT (STICHWORT_DEUTSCH)
(58)             select '#idxStichwort#'
(59)             where not exists
(60)                 (select *
(61)                     from STICHWORT
(62)                     where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#')
(63)     </cfquery>
(64)
(65)     <!-- Verknüpfung zwischen Blob und neuen Stichworten herstellen --->
(66)     <cfquery name="Insert_Stichw_Blob" datasource="SunDB">
(67)         insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(68)             select STICHWORT.STICHWORTID,#url.id#
(69)             from STICHWORT
(70)             where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#'
(71)     </cfquery>
(72) </cfloop>
(73)
(74) <cfif ParameterExists(form.Stichwort)>
(75)     <!-- Verknüpfung zwischen Blob und bestehenden Stichworten herstellen --->
(76)     <cfquery name="Insert_Stichw_Blob2" datasource="SunDB">
(77)         insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(78)             select STICHWORT.STICHWORTID,#url.id#
(79)             from STICHWORT
(80)             where STICHWORT.STICHWORTID in (#Stichwort#)
(81)     </cfquery>
(82) </cfif>
(83)
(84) <!-- aus Liste gewählter Digigeräte --->
(85) <cfif ParameterExists(form.Digigeraet)>
(86)     <cfset setBild=",DIGITALISIERGERATID=" & form.Digigeraet>
(87) <cfelseif form.Digigeraet_neu is "">
(88)     <!-- kein Digigeraet --->
(89)     <cfset setBild="">
(90) <cfelse>
(91)     <!-- neues Digigeraet in die DB einfügen --->
(92)     <cfquery name="InsertDigigeraet" datasource="SunDB">
(93)         insert into DIGITALISIERGERAT (DIGITALISIERGERAT)
(94)             select '#form.Digigeraet_neu#'
(95)             where not exists
(96)                 (select *
(97)                     from DIGITALISIERGERAT
(98)                     where DIGITALISIERGERAT='#form.Digigeraet_neu#')
(99)     </cfquery>
(100)    <cfquery name="GetDGID" datasource="SunDB">
(101)        select @@identity as DGID
(102)    </cfquery>
(103)    <cfoutput query="GetDGID">
(104)        <cfset setBild=",DIGITALISIERGERATID=" & #DGID#>
(105)    </cfoutput>
(106) </cfif>
(107)
(108) <cfif ParameterExists(form.Original)>
```

```
(109) <cfset set_Original=1>
(110)<cfelse>
(111) <cfset set_Original=0>
(112)</cfif>
(113)
(114)<cfif form.Bildhoehe is '' >
(115) <cfset set_Bildhoehe='null'>
(116)<cfelse>
(117) <cfset set_Bildhoehe='#form.Bildhoehe#'>
(118)</cfif>
(119)<cfif form.Bildbreite is '' >
(120) <cfset set_Bildbreite='null'>
(121)<cfelse>
(122) <cfset set_Bildbreite='#form.Bildbreite#'>
(123)</cfif>
(124)
(125)<cfif form.Farbtiefe is '' >
(126) <cfset set_Farbtiefe='null'>
(127)<cfelse>
(128) <cfset set_Farbtiefe='#form.Farbtiefe#'>
(129)</cfif>
(130)
(131)<cfif form.Farbsystem is '' >
(132) <cfset set_Farbsystem='null'>
(133)<cfelse>
(134) <cfset set_Farbsystem=chr(39) & form.Farbsystem & chr(39)>
(135)</cfif>
(136)
(137)<cfif form.Jahr is '' or form.Monat is '' or form.Tag is ''>
(138) <cfset set_Aufnahmedatum='null'>
(139)<cfelse>
(140) <cfset set_Aufnahmedatum=chr(39) & form.Jahr & '-' & form.Monat & '-' &
form.Tag & chr(39)>
(141)</cfif>
(142)
(143)<cfquery name="UpdateBild" datasource="SunDB">
(144) update BILD
(145) set BILDHOHE=#set_Bildhoehe#,
(146) BILDBREITE=#set_Bildbreite#,
(147) FARBTIEFE=#set_Farbtiefe#,
(148) FARBSYSTEM=#PreserveSingleQuotes(set_Farbsystem)#,
(149) AUFNAHMEDATUM=#PreserveSingleQuotes(set_Aufnahmedatum)#,
(150) ORIGINAL=#set_Original#
(151) #setBild#
(152) where BLOBID=#url.ID#
(153)</cfquery>
(154)
(155)<cfquery name="DelDigigeraet" datasource="SunDB">
(156) delete DIGITALISIERGERAT
(157) where DIGITALISIERGERATID not in
(158) (select DIGITALISIERGERATID
(159) from BILD)
(160)</cfquery>
(161)
(162)<cfif ParameterExists(form.delBildkat)>
(163) <cfquery name="DelBildkat_Bild" datasource="SunDB">
(164) delete BILDKAT_BILD
(165) where BLOBID=#url.ID#
(166) and BILDKATEGORIE in (#form.delBildkat#)
(167) </cfquery>
(168)
(169) <cfquery name="DelBildkategorie" datasource="SunDB">
(170) delete BILDKATEGORIE
(171) where BILDKATEGORIEID not in
(172) (select BILDKATEGORIEID
(173) from BILDKAT_BILD)
(174) </cfquery>
(175)</cfif>
(176)
(177)<!-- neue Bildkategorien in Tabellen einfügen --->
(178)<cfloop index="idxBildkategorie" list="#Bildkategorie_neu#" delimiters=";">
(179) <cfquery name="InsertBildkategorie" datasource="SunDB">
(180) insert into BILDKATEGORIE (BILDKATEGORIE)
(181) select '#idxBildkategorie#'
(182) where not exists
(183) (select *
(184) from BILDKATEGORIE
(185) where BILDKATEGORIE.BILDKATEGORIE='#idxBildkategorie#')
(186) </cfquery>
(187)
(188) <!-- Verknüpfung zwischen Bild und neuen Bildkategorien herstellen --->
(189) <cfquery name="Insert_Bildkat_Bild" datasource="SunDB">
(190) insert into BILDKAT_BILD (BILDKATEGORIEID,BLOBID)
```

```

(191)         select BILDKATEGORIE.BILDKATEGORIEID,#url.id#
(192)             from BILDKATEGORIE
(193)             where BILDKATEGORIE.BILDKATEGORIE=#idxBildkategorie#
(194)     </cfquery>
(195)</cfloop>
(196)
(197)<cfif ParameterExists(form.Bildkategorie)>
(198)     <!--- Verknüpfung zwischen Bild und bestehenden Bildkategorien herstellen --
        ->
(199)     <cfquery name="Insert_Bildkat_Bild2" datasource="SunDB">
(200)         insert into BILDKAT_BILD (BILDKATEGORIEID,BLOBID)
(201)         select BILDKATEGORIE.BILDKATEGORIEID,#url.id#
(202)         from BILDKATEGORIE
(203)         where BILDKATEGORIE.BILDKATEGORIEID in (#Bildkategorie#)
(204)     </cfquery>
(205)</cfif>
(206)
(207)<cfquery name="DelArt_Prodfoto" datasource="SunDB">
(208)     delete ART_PRODFOTO
(209)     where ART_PRODFOTO.BLOBID not in (
(210)         select BLOBID
(211)         from BILDKAT_BILD,BILDKATEGORIE
(212)         where BILDKAT_BILD.BILDKATEGORIEID=BILDKATEGORIE.BILDKATEGORIEID
(213)         and BILDKATEGORIE.BILDKATEGORIE='Produktfoto')
(214)</cfquery>
(215)
(216)<cfquery name="DelProdfoto" datasource="SunDB">
(217)     delete PRODUKTFOTO
(218)     where PRODUKTFOTO.BLOBID not in (
(219)         select BLOBID
(220)         from BILDKAT_BILD,BILDKATEGORIE
(221)         where BILDKAT_BILD.BILDKATEGORIEID=BILDKATEGORIE.BILDKATEGORIEID
(222)         and BILDKATEGORIE.BILDKATEGORIE='Produktfoto')
(223)</cfquery>
(224)
(225)<cflocation url="Bild_frm.cfm?RequestTimeout=300">

```

## 12 Bild\_update\_frm.cfm

```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="GetData" datasource="SunDB">
(4)     select
(5)         BLOB.BLOBID,BESCHREIBUNG,DATEIENDUNG,CONTENTTYPE,CONTENTSUBTYPE,ERSTELLER
(6)         from BLOB,DATEITYP,ERSTELLER
(7)         where BLOB.BLOBID=#url.ID#
(8)         and BLOB.DATEITYPID=DATEITYP.DATEITYPID
(9)         and BLOB.ERSTELLERID*=ERSTELLER.ERSTELLERID
(10) </cfquery>
(11) <cfquery name="GetZugeordStichworte" datasource="SunDB">
(12)     select *
(13)         from STICHWORT,STICHWORT_BLOB
(14)         where #url.id#=STICHWORT_BLOB.BLOBID
(15)         and STICHWORT_BLOB.STICHWORTID=STICHWORT.STICHWORTID
(16) </cfquery>
(17)
(18) <cfquery name="GetAlleStichworte" datasource="SunDB">
(19)     select *
(20)         from STICHWORT
(21)         where STICHWORT.STICHWORTID not in
(22)             (select STICHWORTID
(23)              from STICHWORT_BLOB
(24)              where STICHWORT_BLOB.BLOBID=#url.id#)
(25)     order by STICHWORT_DEUTSCH
(26) </cfquery>
(27)
(28) <cfquery name="GetAlleErsteller" datasource="SunDB">
(29)     select *
(30)         from ERSTELLER
(31)         order by ERSTELLER
(32) </cfquery>
(33)
(34) <cf_head title="Daten ändern: Binary Large Object">
(35)
(36) <cfoutput query="GetData">
(37) <form action="Blob_update.cfm?ID=#url.ID#&RequestTimeout=600" method="POST">
(38) <table border=1>
(39) <tr>
(40)     <td><b>ID</b></td>
(41)     <td><b>Beschreibung</b></td>

```



```
(7)     delete BILDKATEGORIE
(8)         where BILDKATEGORIEID=#url.id#
(9) </cfquery>
(10)
(11) <cflocation url="Bildkategorie_frm.cfm">
```

## 14 Bildkategorie\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Bildkategorie" datasource="SunDB">
(4)     select BILDKATEGORIE.*,BLOB.BESCHREIBUNG
(5)     from BILDKATEGORIE,BILDKAT_BILD,BLOB
(6)     where BILDKATEGORIE.BILDKATEGORIEID*=BILDKAT_BILD.BILDKATEGORIEID
(7)         and BILDKAT_BILD.BLOBID*=BLOB.BLOBID
(8)     order by BILDKATEGORIE
(9) </cfquery>
(10)
(11) <cf_head title="Daten anzeigen: Bildkategorien">
(12)
(13) <table border="1">
(14) <tr>
(15) <td valign="top"><b>ID</b></td>
(16) <td valign="top"><b>Bildkategorie</b></td>
(17) <td valign="top"><b>Benutzt für Bild</b></td>
(18) <td valign="top"><b>Aktionen</b></td>
(19) </tr>
(20) <cfoutput query="Bildkategorie" group="BILDKATEGORIEID">
(21) <tr>
(22) <td valign="top">#BILDKATEGORIEID#</td>
(23) <td valign="top">#BILDKATEGORIE#</td>
(24) <td valign="top">
(25)     <cfoutput>#BESCHREIBUNG#<br></cfoutput>
(26) </td>
(27) <td align="center" valign="top">
(28)     <a href="Bildkategorie_delete.cfm?ID=#BILDKATEGORIEID#">löschen</a><br>
(29)     <a href="Bildkategorie_update_frm.cfm?ID=#BILDKATEGORIEID#">ändern</a><br>
(30) </td>
(31) </tr>
(32) </cfoutput>
(33) </table>
(34) <p>
(35) <cf_foot>
```

## 15 Bildkategorie\_update.cfm

```
(1) <cfquery name="UpdateBildkategorie" datasource="SunDB">
(2)     update BILDKATEGORIE
(3)         set BILDKATEGORIE='#form.Bildkategorie#'
(4)         where BILDKATEGORIEID=#form.BildkategorieID#
(5) </cfquery>
(6)
(7) <cflocation url="Bildkategorie_frm.cfm">
```

## 16 Bildkategorie\_update\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Bildkategorie" datasource="SunDB">
(4)     select *
(5)     from BILDKATEGORIE
(6)     where BILDKATEGORIEID=#url.id#
(7) </cfquery>
(8)
(9) <cf_head title="Daten ändern: Bildkategorie">
(10)
(11) <cfoutput query="Bildkategorie">
(12) <form action="Bildkategorie_update.cfm" method="POST">
(13) <input type="Hidden" name="BildkategorieID" value="#BILDKATEGORIEID#">
(14) <table border="1">
(15) <tr>
(16) <td valign="top"><b>ID</b></td>
(17) <td valign="top"><b>Bildkategorie</b></td>
(18) </tr>
(19) <tr>
(20) <td valign="top">#BILDKATEGORIEID#</td>
(21) <td valign="top">
```

```

(22)     <input type="Text" name="Bildkategorie" value="#BILDKATEGORIE#"
        maxlength="50" size="50">
(23) </td>
(24) </tr>
(25) </table>
(26) <p>
(27) <input type="Submit" value="Ändern">
(28) <input type="reset" value="Zurücksetzen">
(29) </form>
(30) </cfoutput>
(31) <p>
(32) <cf_foot>

```

## 17 Blob\_delete.cfm

```

(1) <cfquery name="DelArtProdFoto" datasource="SunDB">
(2)     delete ART_PRODFOTO
(3)     where BLOBID=#url.id#
(4) </cfquery>
(5) <cfquery name="DelProdFoto" datasource="SunDB">
(6)     delete PRODUKTFOTO
(7)     where BLOBID=#url.id#
(8) </cfquery>
(9) <cfquery name="DelBildkatBild" datasource="SunDB">
(10)    delete BILDKAT_BILD
(11)    where BLOBID=#url.id#
(12) </cfquery>
(13) <cfquery name="DelBild" datasource="SunDB">
(14)    delete BILD
(15)    where BLOBID=#url.id#
(16) </cfquery>
(17) <cfquery name="DelText" datasource="SunDB">
(18)    delete TEXTDOKUMENT
(19)    where BLOBID=#url.id#
(20) </cfquery>
(21) <cfquery name="DelStichwortBlob" datasource="SunDB">
(22)    delete STICHWORT_BLOB
(23)    where BLOBID=#url.id#
(24) </cfquery>
(25) <cfquery name="DelBlob" datasource="SunDB">
(26)    delete BLOB
(27)    where BLOB.BLOBID=#url.id#
(28) </cfquery>
(29) <cfquery name="DelDateityp" datasource="SunDB">
(30)    delete DATEITYP
(31)    where DATEITYPID not in
(32)        (select DATEITYPID
(33)         from BLOB)
(34) </cfquery>
(35) <cfquery name="DelErsteller" datasource="SunDB">
(36)    delete ERSTELLER
(37)    where ERSTELLERID not in
(38)        (select ERSTELLERID
(39)         from BLOB)
(40) </cfquery>
(41) <cfquery name="DelDigiGeraet" datasource="SunDB">
(42)    delete DIGITALISIERGERAT
(43)    where DIGITALISIERGERATID not in
(44)        (select DIGITALISIERGERATID
(45)         from BILD)
(46) </cfquery>
(47) <cfquery name="DelBildkategorie" datasource="SunDB">
(48)    delete BILDKATEGORIE
(49)    where BILDKATEGORIEID not in
(50)        (select BILDKATEGORIEID
(51)         from BILDKAT_BILD)
(52) </cfquery>
(53) <cfquery name="DelStichwort" datasource="SunDB">
(54)    delete STICHWORT
(55)    where STICHWORTID not in
(56)        (select STICHWORTID
(57)         from STICHWORT_BLOB)
(58) </cfquery>
(59)
(60) <cflocation url="Blob_frm.cfm">

```

## 18 Blob\_frm.cfm

```

(1) <cfinclude template="const.cfm">

```

```

(2)
(3) <cfquery name="Blobs" datasource="SunDB">
(4)   select BLOB.*, ERSTELLER, DATEITYP.*, STICHWORT_DEUTSCH
(5)   from BLOB,DATEITYP,ERSTELLER,STICHWORT,STICHWORT_BLOB
(6)   where BLOB.DATEITYPID=DATEITYP.DATEITYPID
(7)         and BLOB.BLOBID*=STICHWORT_BLOB.BLOBID
(8)         and STICHWORT_BLOB.STICHWORTID*=STICHWORT.STICHWORTID
(9)         and BLOB.ERSTELLERID*=ERSTELLER.ERSTELLERID
(10)        and BLOB.BLOBID not in
(11)          (select BLOBID from BILD)
(12)        and BLOB.BLOBID not in
(13)          (select BLOBID from TEXTDOKUMENT)
(14)   order by BLOB.BESCHREIBUNG
(15) </cfquery>
(16)
(17) <cf_head title="Daten anzeigen: Binary Large Objects">
(18)
(19) BLOBs, die in der DB gespeichert sind und keine Bilder bzw. Textdokumente sind:
(20) <p>
(21) <table border="1">
(22) <tr>
(23) <td valign="top"><b>ID</b></td>
(24) <td valign="top"><b>Beschreibung</b></td>
(25) <td valign="top"><b>Größe</b></td>
(26) <td valign="top">&nbsp;</td>
(27) <td valign="top"><b>Ersteller</b></td>
(28) <td valign="top"><b>Stichwort</b></td>
(29) <td valign="top"><b>Aktionen</b></td>
(30) </tr>
(31) <cfoutput query="Blobs" group="BLOBID">
(32) <tr>
(33) <td valign="top">#BLOBID#</td>
(34) <td valign="top">#BESCHREIBUNG#</td>
(35) <td valign="top">#GROSSE#</td>
(36) <td valign="top">
(37)   Endung: #DATEIENDUNG#<br>
(38)   Typ: #CONTENTTYPE#/#CONTENTSUBTYPE#
(39) </td>
(40) <td valign="top">#ERSTELLER#</td>
(41) <td valign="top">
(42) <cfoutput>#STICHWORT_DEUTSCH#<br></cfoutput>
(43) </td>
(44) <td align="center" valign="top">
(45)   <a href="Blob_view.cfm?ID=#BLOBID#">anzeigen/speichern</a><br>
(46)   <a href="Blob_delete.cfm?ID=#BLOBID#">löschen</a><br>
(47)   <a href="Blob_update_frm.cfm?ID=#BLOBID#">ändern</a><br>
(48) </td>
(49) </tr>
(50) </cfoutput>
(51) </table>
(52)
(53) <br>
(54) <p>
(55) <a href="Blob_insert_frm.cfm">"Binary Large Object" hinzufügen.</a>
(56)
(57) <cf_foot>

```

## 19 Blob\_insert.cfm

```

(1) <cfinclude template="const.cfm">
(2)
(3) <!--- Datei vom Client auf Server übertragen --->
(4) <cffile action="upload" filefield="BINARDATEN" destination="#wwwroot##wwwsubdir#"
(5)   nameconflict="Makeunique" attributes="temporary">
(6)
(7) <cfset Dateiname = #file.ServerDirectory# & "\" & #file.ServerFile#>
(8) <cfquery name="InsertDateityp" datasource="SunDB">
(9)   insert into DATEITYP (DATEIENDUNG,CONTENTTYPE,CONTENTSUBTYPE)
(10)  select
(11)    '#file.ServerFileExt#','#file.ContentType#','#file.ContentSubType#'
(12)    where not exists
(13)      (select *
(14)       from DATEITYP
(15)       where CONTENTTYPE='#file.ContentType#'
(16)         and CONTENTSUBTYPE='#file.ContentSubType#'
(17)         and DATEIENDUNG='#file.ServerFileExt#')
(18) </cfquery>
(19) <!--- aus Liste gewählter Ersteller --->
(20) <cfif ParameterExists(form.Ersteller)>

```

```
(21) <cfset insBlob=form.Ersteller>
(22) <cfelseif form.Ersteller_neu is "">
(23) <!--- kein Ersteller --->
(24) <cfset insBlob="null">
(25) <cfelse>
(26) <!--- neuen Ersteller in die DB einfügen --->
(27) <cfquery name="InsertErsteller" datasource="SunDB">
(28)     insert into ERSTELLER (ERSTELLER)
(29)         select '#form.Ersteller_neu#'
(30)         where not exists
(31)             (select *
(32)              from ERSTELLER
(33)              where ERSTELLER.ERSTELLER='#form.Ersteller_neu#')
(34) </cfquery>
(35) <cfquery name="GetErID" datasource="SunDB">
(36)     select @@identity as EID
(37) </cfquery>
(38) <cfoutput query="GetErID">
(39)     <cfset insBlob=EID>
(40) </cfoutput>
(41) </cfif>
(42)
(43) <!--- Datei vom Server in die Datenbank übertragen --->
(44) <cfquery name="GenBlob" datasource="SunDB">
(45)     insert into BLOB (DATEITYPID,BESCHREIBUNG,GROSSE,ERSTELLERID)
(46)         select DATEITYPID,'#form.Beschreibung#','#file.FileSize#','#insBlob#
(47)         from DATEITYP
(48)         where CONTENTTYPE='#file.ContentType#'
(49)             and CONTENTSUBTYPE='#file.ContentSubType#'
(50)             and DATEIENDUNG='#file.ServerFileExt#'
(51) </cfquery>
(52)
(53) <cfquery name="GetBID" datasource="SunDB">
(54)     select @@identity as BID
(55) </cfquery>
(56) <cfoutput query="GetBID">
(57)     <cfset BlobID = #BID#>
(58) </cfoutput>
(59)
(60) <CFX_PUTIMAGE DATASOURCE="SunDB" USER="stamm" PASSWORD="frusips"
(61)     SQL="     update BLOB
(62)           set BINARDATEN=?
(63)           where BLOBID= #BlobID#"
(64)     INPUT="#Dateiname#">
(65)
(66) <!--- Datei vom Server löschen --->
(67) <cffile action="Delete" File="#Dateiname#">
(68)
(69) <!--- neue Stichworte in Tabellen einfügen --->
(70) <cfloop index="idxStichwort" list="#Stichwort_neu#" delimiters=";">
(71)     <cfquery name="InsertStichwort" datasource="SunDB">
(72)         insert into STICHWORT (STICHWORT_DEUTSCH)
(73)             select '#idxStichwort#'
(74)             where not exists
(75)                 (select *
(76)                  from STICHWORT
(77)                  where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#')
(78)     </cfquery>
(79)
(80) <!--- Verknüpfung zwischen Blob und neuen Stichworten herstellen --->
(81)     <cfquery name="Insert_Stichw_Blob" datasource="SunDB">
(82)         insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(83)             select STICHWORT.STICHWORTID,#BlobID#
(84)             from STICHWORT
(85)             where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#'
(86)     </cfquery>
(87) </cfloop>
(88)
(89) <cfif ParameterExists(form.Stichwort)>
(90) <!--- Verknüpfung zwischen Blob und bestehenden Stichworten herstellen --->
(91)     <cfquery name="Insert_Stichw_Blob2" datasource="SunDB">
(92)         insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(93)             select STICHWORT.STICHWORTID,#BlobID#
(94)             from STICHWORT
(95)             where STICHWORT.STICHWORTID in (#form.Stichwort#)
(96)     </cfquery>
(97) </cfif>
(98)
(99) <!--- Ausgabe --->
(100) <cflocation url="Blob_frm.cfm?RequestTimeout=300">
```

## 20 Blob\_insert\_frm.cfm

```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="GetErsteller" datasource="SunDB">
(4)     select *
(5)     from ERSTELLER
(6)     order by ERSTELLER.ERSTELLER
(7) </cfquery>
(8)
(9) <cfquery name="GetStichwort" datasource="SunDB">
(10)    select *
(11)    from STICHWORT
(12)    order by STICHWORT.STICHWORT_DEUTSCH
(13) </cfquery>
(14)
(15) <cf_head title="Daten einfügen: Binary Large Object">
(16)
(17) <FORM ACTION="Blob_insert.cfm?RequestTimeout=300" ENCTYPE="multipart/form-data"
    METHOD="post">
(18) <p>Bitte wählen Sie eine Datei aus, die Sie in der Datenbank speichern möchten:
(19) <p><INPUT NAME="BINARDATEN" size="50" TYPE="file">
(20) <p> Geben Sie der Datei einen Titel oder kurze Beschreibung:
(21) <p><INPUT NAME="Beschreibung" size="50" TYPE="text">
(22)
(23) <table>
(24) <tr>
(25)     <td valign="top" width="50%">
(26)         Ersteller:<br>
(27)         <select name="Ersteller" size="5">
(28)             <cfoutput query="GetErsteller">
(29)                 <option value="#ERSTELLERID#">#ERSTELLER#
(30)             </cfoutput>
(31)         </select>
(32)     </td>
(33)     <td valign="top" width="50%">
(34)         Neuer Ersteller:<br>
(35)         <input type="Text" name="Ersteller_neu" maxlength="50" size="30">
(36)     </td>
(37) </tr>
(38) </table>
(39) <table>
(40) <tr>
(41)     <td valign="top" width="50%">
(42)         Stichworte:<br>
(43)         <select name="Stichwort" multiple size="5">
(44)             <cfoutput query="GetStichwort">
(45)                 <option value="#STICHWORTID#">#STICHWORT_DEUTSCH#
(46)             </cfoutput>
(47)         </select>
(48)     </td>
(49)     <td valign="top" width="50%">
(50)         Neue Stichwörter <font size="-1">(mehrere Stichwörter mit Semikolon
getrennt eingeben)</font>:<br>
(51)         <input type="Text" name="Stichwort_neu" maxlength="250" size="30">
(52)     </td>
(53) </tr>
(54) </table>
(55) <p>
(56) <table>
(57) <tr>
(58)     <td><INPUT TYPE="submit" VALUE="Datei hochladen"></td>
(59)     <td><input type="reset" value="Zurücksetzen"></td>
(60) </tr>
(61) </table>
(62)
(63) </FORM>
(64) <p>Der Hochladevorgang kann je nach Größe der Binärdatei einige Minuten dauern.
(65)
(66) <cf_foot>

```

## 21 Blob\_update.cfm

```

(1) <!-- aus Liste gewählter Ersteller -->
(2) <cfif ParameterExists(form.Ersteller)>
(3)     <cfset setBlob=",ERSTELLERID=" & form.Ersteller>
(4) <cfelseif form.Ersteller_neu is "">
(5)     <!-- kein Ersteller -->
(6)     <cfset setBlob="">
(7) <cfelse>

```

```

(8)      <!--- neuen Ersteller in die DB einfügen --->
(9)      <cfquery name="InsertErsteller" datasource="SunDB">
(10)         insert into ERSTELLER (ERSTELLER)
(11)            select '#form.Ersteller_neu#'
(12)            where not exists
(13)                (select *
(14)                   from ERSTELLER
(15)                   where ERSTELLER='#form.Ersteller_neu#')
(16)      </cfquery>
(17)      <cfquery name="GetErID" datasource="SunDB">
(18)         select @@identity as ErID
(19)      </cfquery>
(20)      <cfoutput query="GetErID">
(21)         <cfset setBlob=",ERSTELLERID=" & #ErID#>
(22)      </cfoutput>
(23) </cfif>
(24)
(25) <cfquery name="UpdateBlob" datasource="SunDB">
(26)     update BLOB
(27)        set BESCHREIBUNG='#form.Beschreibung_neu#'
(28)        #setBlob#
(29)        where BLOBID=#url.ID#
(30) </cfquery>
(31)
(32) <cfquery name="DelErsteller" datasource="SunDB">
(33)     delete ERSTELLER
(34)        where ERSTELLERID not in
(35)            (select ERSTELLERID
(36)               from BLOB)
(37) </cfquery>
(38)
(39) <cfif ParameterExists(form.del)>
(40)     <cfquery name="DelStichw_Blob" datasource="SunDB">
(41)        delete STICHWORT_BLOB
(42)           where BLOBID=#url.ID#
(43)           and STICHWORTID in (#form.del#)
(44)     </cfquery>
(45)
(46)     <cfquery name="DelStichwort" datasource="SunDB">
(47)        delete STICHWORT
(48)           where STICHWORTID not in
(49)               (select STICHWORTID
(50)                  from STICHWORT_BLOB)
(51)     </cfquery>
(52) </cfif>
(53)
(54) <!--- neue Stichworte in Tabellen einfügen --->
(55) <cfloop index="idxStichwort" list="#Stichwort_neu#" delimiters=";">
(56)     <cfquery name="InsertStichwort" datasource="SunDB">
(57)        insert into STICHWORT (STICHWORT_DEUTSCH)
(58)           select '#idxStichwort#'
(59)           where not exists
(60)               (select *
(61)                  from STICHWORT
(62)                  where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#')
(63)     </cfquery>
(64)
(65)     <!--- Verknüpfung zwischen Blob und neuen Stichworten herstellen --->
(66)     <cfquery name="Insert_Stichw_Blob" datasource="SunDB">
(67)        insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(68)           select STICHWORT.STICHWORTID,#url.id#
(69)           from STICHWORT
(70)           where STICHWORT.STICHWORT_DEUTSCH='#idxStichwort#'
(71)     </cfquery>
(72) </cfloop>
(73)
(74) <cfif ParameterExists(form.Stichwort)>
(75)     <!--- Verknüpfung zwischen Blob und bestehenden Stichworten herstellen --->
(76)     <cfquery name="Insert_Stichw_Blob2" datasource="SunDB">
(77)        insert into STICHWORT_BLOB (STICHWORTID,BLOBID)
(78)           select STICHWORT.STICHWORTID,#url.id#
(79)           from STICHWORT
(80)           where STICHWORT.STICHWORTID in (#Stichwort#)
(81)     </cfquery>
(82) </cfif>
(83)
(84) <cflocation url="Blob_frm.cfm?RequestTimeout=300">

```

## 22 Blob\_update\_frm.cfm

```

(1) <cfinclude template="const.cfm">

```

```

(2)
(3) <cfquery name="GetData" datasource="SunDB">
(4)     select
      BLOB.BLOBID,BESCHREIBUNG,DATEIENDUNG,CONTENTTYPE,CONTENTSUBTYPE,ERSTELLER
(5)     from BLOB,DATEITYP,ERSTELLER
(6)     where BLOB.BLOBID=#url.ID#
(7)           and BLOB.DATEITYPID=DATEITYP.DATEITYPID
(8)           and BLOB.ERSTELLERID*=ERSTELLER.ERSTELLERID
(9) </cfquery>
(10)
(11) <cfquery name="GetZugeordStichworte" datasource="SunDB">
(12)     select *
(13)     from STICHWORT,STICHWORT_BLOB
(14)     where #url.id#=STICHWORT_BLOB.BLOBID
(15)           and STICHWORT_BLOB.STICHWORTID=STICHWORT.STICHWORTID
(16) </cfquery>
(17)
(18) <cfquery name="GetAlleStichworte" datasource="SunDB">
(19)     select *
(20)     from STICHWORT
(21)     where STICHWORT.STICHWORTID not in
(22)           (select STICHWORTID
(23)            from STICHWORT_BLOB
(24)            where STICHWORT_BLOB.BLOBID=#url.id#)
(25)     order by STICHWORT_DEUTSCH
(26) </cfquery>
(27)
(28) <cfquery name="GetAlleErsteller" datasource="SunDB">
(29)     select *
(30)     from ERSTELLER
(31)     order by ERSTELLER
(32) </cfquery>
(33)
(34) <cf_head title="Daten ändern: Binary Large Object">
(35)
(36) <cfoutput query="GetData">
(37) <form action="Blob_update.cfm?ID=#url.ID#&RequestTimeout=600" method="POST">
(38) <table border=1>
(39) <tr>
(40)     <td><b>ID</b></td>
(41)     <td><b>Beschreibung</b></td>
(42)     <td><b>Endung</b></td>
(43)     <td><b>Content Type</b></td>
(44)     <td><b>Ersteller</b></td>
(45) </tr>
(46) <tr>
(47)     <td>#BLOBID#</td>
(48)     <td><input type="Text" name="Beschreibung_neu" value="#BESCHREIBUNG#"
      maxlength="200" size="50"></td>
(49)     <td align="center">#DATEIENDUNG#</td>
(50)     <td align="center">#CONTENTTYPE#/#CONTENTSUBTYPE#</td>
(51)     <td align="center">#ERSTELLER#</td>
(52) </tr>
(53) </table>
(54) </cfoutput>
(55) <p>
(56) <table>
(57) <tr>
(58)     <td valign="top" width="50%">
(59)         Ersteller:<br>
(60)         <select name="Ersteller" size="5">
(61)             <cfoutput query="GetAlleErsteller">
(62)                 <option value="#ERSTELLERID#">#ERSTELLER#
(63)             </cfoutput>
(64)         </select>
(65)     </td>
(66)     <td valign="top" width="50%">
(67)         Neuer Ersteller:<br>
(68)         <input type="Text" name="Ersteller_neu" maxlength="50" size="30">
(69)     </td>
(70) </tr>
(71) </table>
(72) <p>
(73) <table border="1">
(74) <tr>
(75)     <td><b>Stichwort</b></td>
(76)     <td>Zuweisung entfernen</td>
(77) </tr>
(78) <cfoutput query="GetZugeordStichworte">
(79)     <tr>
(80)         <td>#STICHWORT_DEUTSCH#</td>
(81)         <td align="center">

```



```
(28) <td valign="top">#CONTENTSUBTYPE#</td>
(29) <td valign="top">
(30)     <cfoutput>#BESCHREIBUNG#<br></cfoutput>
(31) </td>
(32) <td align="center" valign="top">
(33)     <a href="Dateityp_update_frm.cfm?ID=#DATEITYPID#">ändern</a><br>
(34) </td>
(35) </tr>
(36) </cfoutput>
(37) </table>
(38) <p>
(39) <cf_foot>
```

## 26 Dateityp\_update.cfm

```
(1) <cfquery name="UpdateDateityp" datasource="SunDB">
(2)     update DATEITYP
(3)         set DT_BESCHREIBUNG='#form.DT_Beschreibung#',
(4)             DATEIENDUNG='#form.Dateiendung#',
(5)             CONTENTTYPE='#form.Contenttype#',
(6)             CONTENTSUBTYPE='#form.Contentsubtype#'
(7)         where DATEITYPID=#form.Dateitypid#
(8) </cfquery>
(9)
(10) <cflocation url="Dateityp_frm.cfm">
```

## 27 Dateityp\_update\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Dateityp" datasource="SunDB">
(4)     select *
(5)     from DATEITYP
(6)     where DATEITYPID=#url.id#
(7) </cfquery>
(8)
(9) <cf_head title="Daten ändern: Dateityp">
(10)
(11) <cfoutput query="Dateityp">
(12) <form action="Dateityp_update.cfm" method="POST">
(13) <input type="Hidden" name="Dateitypid" value="#DATEITYPID#">
(14) <table border="1">
(15) <tr>
(16) <td valign="top"><b>ID</b></td>
(17) <td valign="top"><b>Beschreibung</b></td>
(18) </tr>
(19) <tr>
(20) <td valign="top">#DATEITYPID#</td>
(21) <td valign="top">
(22)     <input type="Text" name="DT_Beschreibung" value="#DT_BESCHREIBUNG#"
(23)     maxlength="200" size="50">
(24) </td>
(25) </tr>
(26) <p>
(27) <table border="1">
(28) <tr>
(29) <td valign="top"><b>Dateiendung</b></td>
(30) <td valign="top"><b>Contenttype</b></td>
(31) <td valign="top"><b>Contentsubtype</b></td>
(32) </tr>
(33) <tr>
(34) <td valign="top">
(35)     <input type="Text" name="Dateiendung" value="#DATEIENDUNG#" maxlength="4"
(36)     size="4">
(37) </td>
(38) <td valign="top">
(39)     <input type="Text" name="Contenttype" value="#CONTENTTYPE#" maxlength="50"
(40)     size="20">
(41) </td>
(42) <td valign="top">
(43)     <input type="Text" name="Contentsubtype" value="#CONTENTSUBTYPE#"
(44)     maxlength="50" size="20">
(45) </td>
(46) </tr>
(47) </table>
(48) <p>
(49) <input type="Submit" value="Ändern">
(50) <input type="reset" value="Zurücksetzen">
```

```
(48) </form>
(49) </cfoutput>
(50) <p>
(51) <cf_foot>
```

## 28 Digigeraet\_delete.cfm

```
(1) <cfquery name="UpdateBild" datasource="SunDB">
(2)     update BILD
(3)         set DIGITALISIERGERATID=null
(4)         where DIGITALISIERGERATID=#url.id#
(5) </cfquery>
(6)
(7) <cfquery name="DelDigitalisiergeraet" datasource="SunDB">
(8)     delete DIGITALISIERGERAT
(9)         where DIGITALISIERGERATID=#url.id#
(10) </cfquery>
(11)
(12) <cflocation url="Digitalisiergeraet_frm.cfm">
```

## 29 Digigeraet\_update.cfm

```
(1) <cfquery name="UpdateDigigeraet" datasource="SunDB">
(2)     update DIGITALISIERGERAT
(3)         set DIGITALISIERGERAT='#form.Digigeraet#'
(4)         where DIGITALISIERGERATID=#form.DigigeraetID#
(5) </cfquery>
(6)
(7) <cflocation url="Digitalisiergeraet_frm.cfm">
```

## 30 Digigeraet\_update\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Digigeraet" datasource="SunDB">
(4)     select *
(5)     from DIGITALISIERGERAT
(6)     where DIGITALISIERGERATID=#url.id#
(7) </cfquery>
(8)
(9) <cf_head title="Daten ändern: Digitalisiergerät">
(10)
(11) <cfoutput query="Digigeraet">
(12) <form action="Digigeraet_update.cfm" method="POST">
(13) <input type="Hidden" name="DigigeraetID" value="#DIGITALISIERGERATID#">
(14) <table border="1">
(15) <tr>
(16) <td valign="top"><b>ID</b></td>
(17) <td valign="top"><b>Digitalisiergeraet</b></td>
(18) </tr>
(19) <tr>
(20) <td valign="top">#DIGITALISIERGERATID#</td>
(21) <td valign="top">
(22)     <input type="Text" name="Digigeraet" value="#DIGITALISIERGERAT#"
(23)     maxlength="50" size="50">
(24) </td>
(25) </tr>
(26) </table>
(27) <input type="Submit" value="Ändern">
(28) <input type="reset" value="Zurücksetzen">
(29) </form>
(30) </cfoutput>
(31) <p>
(32) <cf_foot>
```

## 31 Digitalisiergeraet\_frm.cfm

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Digitalisiergeraet" datasource="SunDB">
(4)     select DIGITALISIERGERAT.*,BLOB.BESCHREIBUNG
(5)     from DIGITALISIERGERAT,BILD,BLOB
(6)     where DIGITALISIERGERAT.DIGITALISIERGERATID=BILD.DIGITALISIERGERATID
```

```

(7)         and BILD.BLOBID=BLOB.BLOBID
(8)         order by DIGITALISIERGERAT
(9) </cfquery>
(10)
(11) <cf_head title="Daten anzeigen: Digitalisiergerät">
(12)
(13) <table border="1">
(14) <tr>
(15) <td valign="top"><b>ID</b></td>
(16) <td valign="top"><b>Digitalisiergeraet</b></td>
(17) <td valign="top"><b>Benutzt für Bild</b></td>
(18) <td valign="top"><b>Aktionen</b></td>
(19) </tr>
(20) <cfoutput query="Digitalisiergeraet" group="DIGITALISIERGERATID">
(21) <tr>
(22) <td valign="top">#DIGITALISIERGERATID#</td>
(23) <td valign="top">#DIGITALISIERGERAT#</td>
(24) <td valign="top">
(25) <cfoutput>#BESCHREIBUNG#<br></cfoutput>
(26) </td>
(27) <td align="center" valign="top">
(28) <a href="Digigeraet_delete.cfm?ID=#DIGITALISIERGERATID#">löschen</a><br>
(29) <a href="Digigeraet_update_frm.cfm?ID=#DIGITALISIERGERATID#">ändern</a><br>
(30) </td>
(31) </tr>
(32) </cfoutput>
(33) </table>
(34) <p>
(35) </cf_foot>

```

### 32 Ersteller\_delete.cfm

Siehe Kapitel 0.

### 33 Ersteller\_update.cfm

```

(1) <cfquery name="UpdateErsteller" datasource="SunDB">
(2)     update ERSTELLER
(3)     set ERSTELLER='#form.Ersteller#'
(4)     where ERSTELLERID=#form.ErstellerID#
(5) </cfquery>
(6)
(7) <cflocation url="Ersteller_frm.cfm">

```

### 34 Ersteller\_update\_frm.cfm

```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Ersteller" datasource="SunDB">
(4)     select *
(5)     from ERSTELLER
(6)     where ERSTELLERID=#url.id#
(7) </cfquery>
(8)
(9) <cf_head title="Daten ändern: Ersteller">
(10)
(11) <cfoutput query="Ersteller">
(12) <form action="Ersteller_update.cfm" method="POST">
(13) <input type="Hidden" name="ErstellerID" value="#ERSTELLERID#">
(14) <table border="1">
(15) <tr>
(16) <td valign="top"><b>ID</b></td>
(17) <td valign="top"><b>Ersteller</b></td>
(18) </tr>
(19) <tr>
(20) <td valign="top">#ERSTELLERID#</td>
(21) <td valign="top">
(22) <input type="Text" name="Ersteller" value="#ERSTELLER#" maxlength="50"
(23) size="50">
(24) </td>
(25) </tr>
(26) </table>
(27) <p>
(28) <input type="Submit" value="Ändern">
(29) <input type="reset" value="Zurücksetzen">

```

```
(29) </form>
(30) </cfoutput>
(31) <p>
(32) <cf_foot>
```

### 35 *Foot.cfm*

Siehe Kapitel 3.5.1.

### 36 *Head.cfm*

Siehe Kapitel 3.5.1.

### 37 *Prodfoto\_update.cfm*

```
(1) <cfquery name="GetProdfoto" datasource="SunDB">
(2)     select *
(3)         from PRODUKTFOTO
(4)         where BLOBID=#url.ID#
(5) </cfquery>
(6)
(7) <cfif ParameterExists(form.Freigestellt)>
(8)     <cfset Freigestellt=1>
(9) <cfelse>
(10)    <cfset Freigestellt=0>
(11) </cfif>
(12)
(13) <cfset Produktausrichtung='null'>
(14)
(15) <cfif ParameterExists(form.Produktausrichtung)>
(16)    <cfif form.Produktausrichtung is not 'keine Angabe'>
(17)        <cfset Produktausrichtung=chr(39) & form.Produktausrichtung & chr(39)>
(18)    </cfif>
(19) </cfif>
(20)
(21) <cfif GetProdfoto.RecordCount is 0 >
(22)    <cfquery name="insProdfoto" datasource="SunDB">
(23)        insert into PRODUKTFOTO
(24)            values (#url.ID#,'#form.Blickrichtung#',
(25)                #Freigestellt#,#PreserveSingleQuotes(Produktausrichtung)#)
(26)    </cfquery>
(27) <cfelse>
(28)    <cfquery name="updProdfoto" datasource="SunDB">
(29)        update PRODUKTFOTO
(30)            set BLICKRICHTUNG='#form.Blickrichtung#',
(31)                FREIGESTELLT=#Freigestellt#,
(32)                PRODUKTAUSRICHTUNG=#PreserveSingleQuotes(Produktausrichtung)#
(33)        where BLOBID=#url.ID#
(34)    </cfquery>
(35) </cfif>
(36)
(37) <cflocation url="Produktfoto_frm.cfm?ID=#url.ID#">
```

### 38 *Prodfoto\_update\_frm.cfm*

```
(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="GetData" datasource="SunDB">
(4)     select BLOB.*,DATEITYP.*,PRODUKTFOTO.BLICKRICHTUNG,
(5)         PRODUKTFOTO.FREIGESTELLT,PRODUKTFOTO.PRODUKTAUSRICHTUNG,
(6)         ARTIKELDUMMY.*
(7)     from BLOB,DATEITYP,PRODUKTFOTO,ART_PRODFOTO, ARTIKELDUMMY
(8)     where BLOB.BLOBID=#url.id#
(9)         and BLOB.DATEITYPID=DATEITYP.DATEITYPID
(10)        and BLOB.BLOBID*=PRODUKTFOTO.BLOBID
(11)        and PRODUKTFOTO.BLOBID*=ART_PRODFOTO.BLOBID
(12)        and ART_PRODFOTO.ARTIKELNR*=ARTIKELDUMMY.ARTIKELNR
(13)     order by BLOB.BESCHREIBUNG
(14) </cfquery>
(15)
(16) <cf_head title="Daten ändern: Produktfoto">
(17)
```



```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Bilder" datasource="SunDB">
(4)     select BLOB.*,DATEITYP.*,PRODUKTFOTO.BLICKRICHTUNG,
(5)           PRODUKTFOTO.FREIGESTELLT,PRODUKTFOTO.PRODUKTAUSRICHTUNG,
(6)           ARTIKELDUMMY.*
(7)     from BLOB,DATEITYP,PRODUKTFOTO,ART_PRODFOTO,ARTIKELDUMMY
(8)     where BLOB.BLOBID=#url.id#
(9)           and BLOB.DATEITYPID=DATEITYP.DATEITYPID
(10)          and BLOB.BLOBID*=PRODUKTFOTO.BLOBID
(11)          and BLOB.BLOBID*=ART_PRODFOTO.BLOBID
(12)          and ART_PRODFOTO.ARTIKELNR*=ARTIKELDUMMY.ARTIKELNR
(13)     order by BLOB.BESCHREIBUNG
(14) </cfquery>
(15)
(16) <cf_head title="Daten anzeigen: Produktfoto">
(17)
(18) <table border="1">
(19) <tr>
(20) <td valign="top"><b>ID</b></td>
(21) <td valign="top"><b>Beschreibung</b></td>
(22) <td valign="top" align="center"><b>Dateigröße</b><br>(in Byte)</td>
(23) <td valign="top" align="center"><b>Endung<br>Dateityp</b></td>
(24) <td valign="top" align="center"><b>Blickrichtung</b></td>
(25) <td valign="top" align="center"><b>Freigestellt</b></td>
(26) <td valign="top" align="center"><b>Produktausrichtung</b></td>
(27) <td valign="top"><b>zugeordnete Artikel</b></td>
(28) <td valign="top"><b>Aktionen</b></td>
(29) </tr>
(30) <cfoutput query="Bilder" group="BLOBID">
(31) <tr>
(32) <td valign="top">#BLOBID#</td>
(33) <td valign="top">#BESCHREIBUNG#&nbsp;&nbsp;&nbsp;</td>
(34) <td valign="top" align="center">#GROSSE#&nbsp;&nbsp;&nbsp;</td>
(35) <td valign="top"
(36)     align="center">#DATEIENDUNG#<br>#CONTENTTYPE#/#CONTENTSUBTYPE#</td>
(37) <td valign="top" align="center">
(38)     <cfif #FREIGESTELLT#>
(39)         ja
(40)     <cfelse>
(41)         nein
(42)     </cfif></td>
(43) <td valign="top" align="center">#PRODUKTAUSRICHTUNG#&nbsp;&nbsp;&nbsp;</td>
(44) <td valign="top" align="left">
(45) <cfoutput>
(46)     <b>#ARTIKELNR#</b> #ARTIKELNAME#<br>
(47) </cfoutput>
(48) </td>
(49) <td align="center" valign="top">
(50)     <a href="Blob_view.cfm?ID=#BLOBID#">anzeigen/speichern</a><br>
(51)     <a href="Bild_delete.cfm?ID=#BLOBID#">löschen</a><br>
(52)     <a href="Prodfoto_update_frm.cfm?ID=#BLOBID#">ändern</a><br>
(53)     <a
(54) href="ArtProdfoto_update_frm.cfm?RequestTimeout=300&ID=#BLOBID#">Artikelzuordnung
(55) </a><br>
(56) </td>
(57) </tr>
(58) </cfoutput>
(59) </table>
(60) <p>
(61) </p>
(62) <cf_foot>

```

## 40 Start.cfm

```

(1) <cfinclude template="const.cfm">
(2) <cf_head title="Startseite">
(3)
(4) <b>Suchformular:</b>
(5) <ul>
(6)     <li><a href="suche_frm.cfm">BLOBs suchen</a>
(7) </li>
(8) </ul>
(9) <b>Datenbearbeitungsformulare:</b>
(10) <table>
(11) <tr>
(12) <td valign="top" width="50%">
(13)     <ul>
(14)         <li><a href="Blob_frm.cfm">BLOBs</a>
(15)         <li><a href="Bild_frm.cfm">Bilder</a>
(16)         <li><a href="Text_frm.cfm">Texte</a>

```

```

(16)         <li><a href="Dateityp_frm.cfm">Dateitypen</a>
(17)         <li><a href="Ersteller_frm.cfm">Ersteller</a>
(18)     </ul>
(19) </td>
(20) <td valign="top" width="50%">
(21)     <ul>
(22)         <li><a href="Stichwort_frm.cfm">Stichwörter</a>
(23)         <li><a href="Digitalisiergeraet_frm.cfm">Digitalisiergeräte</a>
(24)         <li><a href="Bildkategorie_frm.cfm">Bildkategorien</a>
(25)         <li><a href="Artikel_frm.cfm?RequestTimeout=120">Artikel</a>
(26)     </ul>
(27) </td>
(28) </tr>
(29) </table>
(30)
(31) <cf_foot>

```

#### 41 *Stichwort\_delete.cfm*

Siehe Kapitel 0.

#### 42 *Stichwort\_frm.cfm*

Siehe Kapitel 4.1.

#### 43 *Stichwort\_update.cfm*

```

(1) <cfquery name="UpdateStichwort" datasource="SunDB">
(2)     update STICHWORT
(3)         set STICHWORT_DEUTSCH='#form.Stichwort#'
(4)         where STICHWORTID=#form.StichwortID#
(5) </cfquery>
(6)
(7) <cflocation url="Stichwort_frm.cfm">

```

#### 44 *Stichwort\_update\_frm.cfm*

```

(1) <cfinclude template="const.cfm">
(2)
(3) <cfquery name="Stichwort" datasource="SunDB">
(4)     select *
(5)     from STICHWORT
(6)     where STICHWORTID=#url.id#
(7) </cfquery>
(8)
(9) <cf_head title="Daten ändern: Stichwort">
(10)
(11) <cfoutput query="Stichwort">
(12) <form action="Stichwort_update.cfm" method="POST">
(13) <input type="Hidden" name="StichwortID" value="#STICHWORTID#">
(14) <table border="1">
(15) <tr>
(16) <td valign="top"><b>ID</b></td>
(17) <td valign="top"><b>Stichwort</b></td>
(18) </tr>
(19) <tr>
(20) <td valign="top">#STICHWORTID#</td>
(21) <td valign="top">
(22)     <input type="Text" name="Stichwort" value="#STICHWORT_DEUTSCH#"
(23)     maxlength="50" size="50">
(24) </td>
(25) </tr>
(26) </table>
(27) <p>
(28) <input type="Submit" value="Ändern">
(29) <input type="reset" value="Zurücksetzen">
(30) </form>
(31) </cfoutput>
(32) <cf_foot>

```



```
(72)         </cfoutput>
(73)         </select>
(74)     </td>
(75)
(76) </tr>
(77) <tr>
(78)     <td></td>
(79)     <td>Text</td>
(80)     <td><input type="Radio" name="BlobTyp" value="Text"></td>
(81) </tr>
(82) <tr>
(83)     <td></td>
(84)     <td>andere BLOBs</td>
(85)     <td><input type="Radio" name="BlobTyp" value="andere"></td>
(86) </tr>
(87) </table>
(88) <p>
(89) <table>
(90) <tr>
(91)     <td><input type="Submit" name="submit" value="Suche starten"></td>
(92)     <td><input type="reset" value="Zurücksetzen"></td>
(93) </tr>
(94) </table>
(95)
(96) </form>
(97)
(98)
(99) <cf_foot>
```

## D Inhalt der beigefügten CD-ROM

Verzeichnis/Datei	Beschreibung
CD:\ar32d301.exe	Adobe Acrobat Reader 3.01 deutsch Installationsprogramm
CD:\ColdFusion\CFX_Tags	Diverse Cold Fusion Erweiterungs-Tags
CD:\ColdFusion\CFX_Tags\ getputimage	CFX-Tags Put- und Getimage inkl. C++ Quellcode
CD:\ColdFusion\CFX_Tags\imginfo	CFX-Tag ImgInfo
CD:\ColdFusion\Dokumentation	Verschiedene Handbücher für Cold Fusion Application Server und Cold Fusion Studio im Adobe Acrobat oder MS-Word-Format
CD:\ColdFusion\Quellcode	Alle Cold Fusion Anwendungsseiten, die in dieser Arbeit verwendet wurden
CD:\ColdFusion\Testversion	Testversionen verschiedener Programme von Allaire, u.a. Cold Fusion Application Server für Win 95/NT und CF Studio Start mit setup.exe
CD:\ColdFusion\Upgrades	Upgrades für diverse Allaire-Produkte
CD:\HTML-Kurzreferenz	Aktuelle Version der HTML-Kurzreferenz von Stefan Münz ([Münz96a]). Start durch Aufruf der Seite start.htm in einem Webbrowser.
CD:\PowerDesigner und CD:\PowerDesigner\Testversion	Verschiedene Testversionen von Powersoft- Produkten. Start durch Aufruf der Seite start.htm in einem Webbrowser.
CD:\PowerDesigner\Testversion\ Installs\PowrDesr\Da32tlk	Demoversion des Power Designer Data Architect. Start mit setup.exe
CD:\PowerDesigner\Modelle	Konzeptionelles (.cdm) und physikalisches (.pdm) Datenmodell im PowerDesigner- Format und die SQL-Anweisungen (.sql) zur Generierung der Tabellen.
CD:\RFC\	Einige RFCs, u.a. [RFC1344], [RFC1945] und [RFC2302]
CD:\SybaseASE\WinNT60TageEval	60-Tage-Demoversion des Sybase Adaptive Server Enterprise für Win NT. Start mit setup.exe

## Abbildungsverzeichnis

<i>Abbildung 2-1: Grafische Benutzeroberfläche des Power Designer Data Architect.....</i>	<i>12</i>
<i>Abbildung 2-2: Liste der Domains im Power Designer Data Architect.....</i>	<i>15</i>
<i>Abbildung 2-3: Die Entität BLOB .....</i>	<i>17</i>
<i>Abbildung 2-4: 19</i>	
<i>Abbildung 2-5: Symbol für die BLOB-Bild-Relation.....</i>	<i>21</i>
<i>Abbildung 2-6: Entitätstypen BLOB, Bild, Textdokument und weitere binäre Objekte.....</i>	<i>22</i>
<i>Abbildung 2-7: Produktfoto.....</i>	<i>22</i>
<i>Abbildung 2-8: Konzeptionelles Datenmodell zur Archivierung von BLOBs .....</i>	<i>24</i>
<i>Abbildung 2-9: Lavendelfeld in Frankreich .....</i>	<i>27</i>
<i>Abbildung 2-10: Ersteller.....</i>	<i>27</i>
<i>Abbildung 2-11: Konzeptionelles Datenmodell zur Archivierung von BLOBs inklusive der Entitäten für ein geeignetes Information Retrieval.....</i>	<i>29</i>
<i>Abbildung 2-12: Parameterformular für die Generierung eines physikalischen Datenmodells .....</i>	<i>30</i>
<i>Abbildung 2-13: Symbol einer Tabelle .....</i>	<i>31</i>
<i>Abbildung 2-14: Physikalisches Datenmodell zur Archivierung von BLOBs .....</i>	<i>33</i>
<i>Abbildung 3-1: Serverkonfiguration bei der Firma Spinnrad.....</i>	<i>35</i>
<i>Abbildung 3-2: Einstellungen für die BLOB-Datenbank im ODBC-Administrator .....</i>	<i>39</i>
<i>Abbildung 3-3: ... und im Cold Fusion Administrator .....</i>	<i>39</i>
<i>Abbildung 4-1: Übersicht-Menü des BLOB-Archivierungssystems .....</i>	<i>54</i>
<i>Abbildung 4-2: Datenanzeigeformular für Stichwörter .....</i>	<i>55</i>
<i>Abbildung 4-3: Datenanzeigeformular für Bilder.....</i>	<i>57</i>
<i>Abbildung 4-4: Formular zum Einfügen von Bildern (ohne Netscape-Rahmen) .....</i>	<i>64</i>
<i>Abbildung 4-5: Änderungsformular für Bilder (ohne Netscape-Rahmen) .....</i>	<i>73</i>
<i>Abbildung 4-6: Formular für die Suche nach BLOBs.....</i>	<i>78</i>
<i>Abbildung 4-7: Beispiel für ein Suchergebnis.....</i>	<i>85</i>

## Abkürzungsverzeichnis

ASE	Sybase Adaptive Server Enterprise
BLOB	Binary Large Object - Binäres Datenobjekt
BMP	Windows Bitmap Format - Dateiformat für Rasterbilder
CDR	Corel Draw Format - Dateiformat für Vektorgrafiken
CGI	Common Gateway Interface
DB	Datenbank
DBMS	Database Management System oder Datenbank Management System
EPS	Encapsulated Postscript Format - Dateiformat für Vektorgrafiken
ER-Modell	Entity-Relationship-Model - Entitäten-Relationen-Modell nach [Chen91a]
ERM	siehe ER-Modell
GIF	Graphics Interchange Format - Dateiformat für Rasterbilder
HTTP	Hypertext Transfer Protocol
ID	Identifizier - Identifizierungsnummer
int	Integer - Ganzzahldatentyp
IR	Information Retrieval - Wiederauffinden von Informationen
JPG/JPEG	Joint Photographic Experts Group Format - Dateiformat für Rasterbilder
NSAPI	Netscape Server Application Programming Interface
ODBC	Open Database Connectivity
PSD	Adobe Photoshop Format - Dateiformat für Rasterbilder
RDBMS	Relationales Datenbank Management System
RDBVS	Relationales Datenbank-Verwaltungssystem (siehe RDBMS)
RFC	Request For Comment
SQL	Standard Query Language
TIF/TIFF	Tag Image File Format - Dateiformat für Rasterbilder
URL	Uniform Resource Identifier
WWW	World Wide Web

## Literaturverzeichnis

- [Alla97a] Allaire Corp.: „*Cold Fusion Sprachreferenz*“, 1997
- [Alla98a] Allaire Corp.: „*Cold Fusion 3.1 Benutzerhandbuch*“, 1998
- [Ashl96a] Ed Ashley und Beth Epperson: „*Sybase SQL Server on the World Wide Web*“, International Thomson Computer Press
- [Bage96a] Jo Bager: „*Angebot und Abfrage, Datenbanken/Web-Gateways machen Webserver zu Informationsbrennpunkten*“, c't Computermagazin Ausgabe 06/96, Heise Verlag
- [Belo98a] Patrick Belowski: „*Praktische Umsetzung eines Internetschops mit Hilfe einer ausgewählten Electronic-Commerce-Softwarelösung. Diplomarbeit an der Fachhochschule Gelsenkirchen*“, 1998
- [Brat90a] Kenneth S. Brathwaite: „*Datenbankentwurf - eine Einführung*“, McGraw-Hill, 1990
- [Chen91a] Peter P.S. Chen und Heinz-Dieter Knöll: „*Der Entity-Relationship-Ansatz zum logischen Systementwurf, Datenbank- und Programmmentwurf*“, B.I. Wissenschaftsverlag, 1991
- [Comp87a] CompuServe Inc.: „*GIF™ - Graphics Interchange Format, a standard defining a mechanism for the storage and transmission of raster-based graphics information*“, Columbus, 1987
- [Date93a] C. J. Date mit Hugh Darwen: „*A guide to the SQL Standard*“, Addison-Wesley, 1993
- [Dege98a] Werner Degenhardt: „*Verbindung mit Hindernissen*“, LANline Ausgabe 01/98, Seite 62ff
- [Dege98b] Werner Degenhardt und Oliver Diekamp: „*Cold Fusion ist heiß*“, LANline Ausgabe 03/98, Seite 50ff
- [Dude88a] „*Duden Informatik*“, B.I. Wissenschaftsverlag, 1988
- [Fran98a] Christian Franke und Thomas Wehrich: „*Kalte Fusion*“, internet world Ausgabe 03/98, Seite 56ff
- [Gree98a] Philip Greenspan: „*Datenbankgestützte Web-Sites*“, Hanser, 1998
- [Klei97a] Prof. Dr. Peter Kleinschmidt und Christian Rank: „*Relationale Datenbanksysteme, eine praktische Einführung*“, Springer, 1997
- [Meye91a] Prof. Dr.-Ing. Klaus Meyer-Wegener: „*Multimedia-Datenbanken*“, B. G. Teubner, 1991
- [Münz96a] Stefan Münz und Wolfgang Nefzger: „*HTML-Referenz*“, Franzis, 1996
- [Powe97a] Powersoft: „*Power Designer Data Architect™ User's Guide*“, Sybase Inc., 1997
- [RFC1344] Request For Comment No. 1344: „*Implications of MIME for Internet Mail Gateways*“, 1992
- [RFC1945] Request For Comment No. 1945: „*Hypertext Transfer Protocol - HTTP/1.0*“, 1996

- [RFC2302] Request For Comment No. 2302: „*Tag Image File Format (TIFF) - image/tiff, MIME Sub-type Registration*“, 1998
- [Schw96a] Dr. Joachim Schwarte: „*Das große Buch zu HTML, Publizieren im Internet*“, DATA BECKER, 1996
- [Syba97a] Server Publications Group: „*Referenzhandbuch für den Sybase® Adaptive Server™ Enterprise, Band 1: Befehle und Funktionen*“, Sybase Inc., 1997
- [Syba97b] Server Publications Group: „*Referenzhandbuch für den Sybase® Adaptive Server™ Enterprise, Band 2: Prozeduren*“, Sybase Inc., 1997
- [Syba97c] Server Publications Group: „*Referenzhandbuch für den Sybase® Adaptive Server™ Enterprise, Band 3: Datentypen und Systemtabellen*“, Sybase Inc., 1997
- [Syba97d] Server Publications Group: „*Benutzerhandbuch für Sybase® Adaptive Server™ Enterprise Transact-SQL®*“, Sybase Inc., 1997
- [WWW01a] <http://www.webadvisor.com/odbc.html>: „*ODBC*“, 04.09.1998
- [WWW01b] <http://www.webadvisor.com/nsapi.html>: „*NSAPI*“, 04.09.1998
- [WWW02a] [http://home.netscape.com/newsref/std/server\\_api.html](http://home.netscape.com/newsref/std/server_api.html): „*The Netscape Server API*“, 04.09.1998
- [WWW02b] [http://home.netscape.com/newsref/std/nsapi\\_vs\\_cgi.html](http://home.netscape.com/newsref/std/nsapi_vs_cgi.html): „*The NSAPI versus the CGI interface*“, 01.07.1998
- [WWW03a] <http://www.tu-chemnitz.de/~fischer/cgi/overview.html>: „*Common Gateway Interface*“, 20.08.1998
- [WWW04a] <http://www.allaire.com/developer/taggallery/index.cfm>: „*<CF\_Taggallery>*“, 12.08.1998
- [WWW04b] <http://www.allaire.com/Handlers/index.cfm?ID=1462&Method=Full>: „*Cold Fusion Cannot Return BLOBs*“, 12.08.1998
- [WWW04c] <http://www.allaire.com/support/knowledgebase/searchform.cfm>, Artikel Nr. 169: „*Dynamically Populating Images with Cold Fusion*“, 12.08.1998
- [WWW05a] <http://focus.de/D/DC/DCM/dcm.htm?snr=11952>: „*Internet-Boom: Jeder sechste Deutsche ist online*“, 17.09.1998
- [WWW05b] <http://focus.de/D/DB/DBY02/dby02.htm>: „*Wer surft wann, wie lange und warum?*“, 17.09.1998

Bei den Literaturangaben aus dem Internet ([WWW...]) ist zu beachten, daß das dort veröffentlichte Informationsangebot geändert werden kann. Für diese Arbeit wurde der Inhalt zum jeweils angegebenen Datum verwendet.