Datenbankanbindung an das WWW

Handreichung des
Kompetenzzentrums für Multimedia und Telematik
am
Deutschen Institut für Fernstudienforschung

zum Seminar
Datenbankanbindung an das World Wide Web

Autor: M. Knobloch
KMMT am DIFF
Konrad Adenauer Str. 40
72072 Tübingen

1 Datenbankanbindung an das World Wide Web

Das Wort Datenbank taucht im Kontext des WWW häufiger auf als in der Datenverarbeitung der Vor-WWW Zeit. Meist umschreibt man damit einen Bestand von HTML-Seiten, in dem von einem Browser aus recherchiert werden kann.

Die große Menge der Nutzdaten lagert als Dateien, verwaltet lediglich durch das jeweilige Betriebsystem auf Arbeitsstationen oder Netzwerkservern. Privatpersonen, große Verbände und kleine Firmen nutzen fast ausschließlich diese Form der Datenspeicherung, um ihre Informationen zu verwalten.

Die Menge dieser Art Daten hat sich durch das WWW noch einmal drastisch erhöht. Die HTML Dateien entstehen nicht nur für neue Informationen, sondern kommen häufig als webfähiges Doppel zu den vorhandenen Informationen hinzu. Die zentralisierte Auslieferung der HTML Dokumente über einen Web-Server, legt allerdings die Überlegung einer ebenfalls zentralisierten Verwaltung der Daten nahe. Eine Form von Client-Server Architektur entsteht damit fast zwingend.

Diese unmittelbare Form der Speicherung hängt mit dem Aufbau der gebräuchlichen Rechnersysteme zusammen. Der Ort an dem alle Daten vorhanden sein müssen, um vom menschlichen (und z.t. maschinellen) Benutzer konsumiert werden zu können, der Hauptspeicher des Rechners ist flüchtig (transient), damit auch alle Daten und Einstellungen, die zur Laufzeit eines Anwendungsprogramms verfügbar sind. Sollen Informationen über einen Programmlauf hinaus dauerhaft (persistent) sein, müssen sie in irgendeiner über eine Arbeitssitzung hinaus gesichert werden. Dies geschieht traditionell, indem die Menge der zu rettenden Informationen in eine Sequenz von Bytes umgeformt und auf einen dauerhaften Datenträger kopiert wird.

In den 80er Jahren wurde der Begriff Datenbank exakter benutzt. Im klassischen Sprachgebrauch bezeichnet Datenbank ein System zur zentralen Speicherung und Verwaltung von Daten, auf die von einer Nutzergruppe lesend und schreibend zugegriffen werden kann. Wichtig bei der Entstehung der Datenbanken war die Betonung auf die Fähigkeit des Systems, Daten zentral zu verwalten und den Zugriff auf die Daten zu kontrollieren.

Das Thema persistente Datenhaltung läßt sich entsprechend ihrer Entstehung in drei Kategorien unterteilen

- Dateisysteme
- Relationale Datenbanken
- Objektorientierte Datenbanken

Vor den unmittelbaren Zugriff auf persistente Daten ist im WWW noch eine Zwischeninstanz getreten, die Webserver, die die Zugriffe aus dem Netz entgegennehmen und weiterverarbeiten oder weiterleiten müssen. Die "klassische" Client-Server Welt basierte auf speziellen Clients, die einem bestimmten zentralen Dienst zugeordnet waren, d.h. oftmals einer speziellen Datenbank. Die Hersteller von relationalen Datenbanken haben dazu Anwendungsschichtprotokolle auf diverse Netzwerkprotokolle aufgesetzt. Diese herstellerabhängigen Zusätze sind in der Lage, Sequentialisierung und Desequenzialisierung der Nutzdaten zu sichern und dem Endbenutzer zustandsorientiertes Arbeiten, unabhängig vom zugrundeliegenden Netzwerkprotokoll, zu bieten. Dies funktionierte, weil jeder Client über die entsprechenden Zusatzprogramme verfügte, d.h. diese auf der lokalen Maschine installiert waren.

Diese Logik ist im WWW nicht ohne weiters möglich, da es hier bei den Benutzern eines Dienstes meist um unbekannte und gelegentliche Nutzer, also nicht um registrierte Clienten handelt. Das Verbindungsprotokoll zwischen Client und Server ist zustandslos: Die Abarbeitung einer Anfrage an einen Web-Server beendet auch die Client-Server Sitzung, eine weitere Anfrage bedeutet den Beginn einer neuen Client-Server Sitzung. Die Entscheidung, ob eine Anfrage an einen Web-Server nur die Fortsetzung einer vorausgegangenen Anfrage oder vollständig neu ist, ist nicht ohne Hilfskonstrukte möglich.

Eine weiteres Problem der "unbekannten Clients" ist die graphische Repräsentation der Informationen, die ausgeliefert werden. Die bekannten Clients der herkömmlichen Client-Server Welt bedienten sich eines bekannten Betriebssytems und einer bekannten graphischen Oberfläche. Die Interaktionsmöglichkeiten der Nutzer mit dem zentralen Dienst wird durch angepaßte Visualisierungsmöglichkeiten der entsprechenden GUI erleichtert und unterstützt.

Das WWW bietet als Benutzerschnittstelle standardmäßig den Browser, d.h. mit HTML gestaltete Informationen an. Die Möglichkeiten der Bildschirmsteuerung mit HTML sind gegenüber traditionellen Programmierumgebungen stark eingeschränkt. Soll die Benutzeroberfläche flexibel sein und aktuellen Bedienungskomfort bieten, muß auf Applets, oder ActiveX Komponenten ausgewichen werden. Beide Lösungen bringen ihre jeweils eigenen Schwierigkeiten mit sich.

Die Faktoren, die bei der Überlegung einer Daten(bank)-Anbindung an das WWW eine Rolle spielen sind demnach

- Datenhaltung
- Datentransport
- graphische Repräsentation

In der Folgenden Darstellung wird anhand der Datenhaltung gegliedert. Diese Gliederung ist jedoch nicht zwingend und eher als roter Faden durch ein Dickicht von Themen zu verstehen. Die Fragen von Datenhaltung, Kommunikation und Repräsentation sind in der Praxis frei kombiniert.

2 Formen dauerhafter Datenspeicherung

2.1 Speicherung im Dateisystem

Die Ablage von Daten als separate Dateien im Dateisystem ist immer noch die häufigste Form der dauerhaften Speicherung von Nutzdaten. Bei PC-AnwenderInnen sind die Daten innerhalb der Datei strukturiert durch die verwendeten Anwendungsprogramme. Jedes Textverarbeitungssystem legt die Nutzdaten zusammen mit den Steuer- und Darstellungsinformationen (Formatierungen) in einer bestimmten Anordnung innerhalb der Datei ab. Diese Anordnung wird vom jeweiligen Anwendungsprogramm erzeugt, wenn die Daten zum ersten mal gespeichert werden. Entsprechend dieser Anordnung (Dateiformat) werden die Daten beim Öffnen der Datei erneut interpretiert. Die Hersteller von PC-Programmen definieren meist jeweils eigene Dateiformate.

Die Verknüpfung von Inhalten, die in verschiedenen Dateien liegen, ist nur sehr bedingt möglich(Register, Inhaltsverzeichnisse). Eine Überwachung der Gültigkeit von Verknüpfungen zwischen Dateien findet nicht statt. Es gibt ebfalls keine Überwachungsinstanz, die prüft, ob eine Datei gelöscht oder verschoben wurde, auf die noch ein Verweis besteht.

Die Ablage der Informationen im Dateisystem brachte folgende Probleme mit sich.

- Keine konsistenten Verknüpfungen
- Dateiaufbau muß in der Logik des Zugriffsprogramms abgebildet werden
- Dateien mußten sequentiell durchgelesen werden (lange Suchzeiten)
- Dateiformat Wildwuchs
- Gleichzeitiger Zugriff durch mehrere Benutzer schwierig
- Verknüpfung untereinander schwierig
- Redundante Daten

Dennoch wird die übergroße Mehrzahl aller digitalisierten Informationen in Dateien abgelegt. Sie entstehen völlig dezentral und enthalten die Informationen, die die AutorInnen mit Hilfe eines Zugriffsprogramms in ein spezielles Format gebracht haben. Man kann sagen, die Masse der Dateien enthält schwach strukturierte Informationen. Die Strukturierung ist zwingend nur auf der Seite der Technik, also des Dateiformats, jedoch beliebig in inhaltlicher Hinsicht. Als Dateiformat des WWW wird HTML in dieser Form benutzt.

Spezielle Mechanismen und Dateiformen versuchen die Probleme teilweise zu entschärfen. Eine aus der UNIX-Welt kommende Form spezialisierter Dateien und zugehöriger Zugriffsmechanismen sind die dbm-Systeme (dbm, gdbm, sdbm, ...) Sie sind in der Lage Schlüssel-Wert-Paare in Dateien abzulegen. Der Schlüsselwert wird über ein hash-Verfahren codiert und sichert damit einen raschen Zugriff über den Schlüsselbegriff. Die Systeme werden in Anwendungsprogrammen eingesetzt. Traditionelle Sprachen sind C und Perl, eine Schnittstelle für JAVA existiert inzwischen. Die Nutzung der Systeme ist sehr einfach. Eine Programmvariable, die eine hash-Tabelle enthält wird per Funkionsaufruf an eine xdbm Datei gebunden

```
dbmopen(%telinfo, "telinfo", 0666);
```

Im Programm wird nun mit der Variablen gearbeitet. Am Ende der Operationen auf der hash-Tabelle, wird die Datei wieder geschlossen.

```
dbmclose(%telinfo);
```

Diese Dateiform erlaubt den schnellen Zugriff auf Informationen über die Schlüsselbegriffe. Bei Bedarf von verschiedenen Schlüsseln, werden mehrere Dateien angelegt. Eine Sortierung der Daten ist nicht vorhanden, diese muß durch Logik des Anwendungsprogramms erzeugt werden.

Bei der Erschließung von Inhalten, die in statischen HTML Dateien liegen werden Volltextindexsysteme eingesetzt. Viele Volltextindexsystem des WWW bedienen sich dieser Dateiform zur persistenten Haltung der Schlüsselbegriffe.

2.2 Relationale Systeme

Die Datenhaltung und der Zugriff auf Daten wird in relationalen Datenbanksystemen einer zentralen Instanz unterworfen. Sie verwaltet und prüft die Zugriffsberechtigungen und führt angeforderte Aktionen an Daten und Strukturen durch.

Relationale Datenbanksysteme sollen es ermöglichen, von der Art der Datenhaltung zu abstrahieren. Die Nutzdaten sind deshalb, unabhängig von der jeweiligen Anwendung, so umzuformen, daß sie den Regeln relationaler Systeme entsprechen.

Dies ermöglicht es den AnwenderInnen mit standardisierten Zugriffsmethoden auf die Nutzdaten Einfluß zu nehmen. Die Idee, daß AnwenderInnen sich Informationen aus den Datenbeständen erzeugen können ohne Zugriffsprogramme zu schreiben, führte zur Entwicklung einer standardisierten Abfragesprache (SQL – structured query language).

Mit SQL ist es möglich Ergebnismengen zu definieren, die Erzeugung dieser Ergebnismengen jedoch dem relational database management system (RDBMS) zu überlassen.

Meist läuft das Datenbanksystem als Serverprozess auf einer Maschine im Netz. Die Anfragen werden von Clients im Netz an die Datenbank gestellt. Der DB Serverprozess verarbeitet die Anfragen und schickt die Ergebnismenge an den Client zurück.

2.2.1 Datenhaltung in Tabellen

Relationale Datenbanken legen Nutzdaten und Verwaltungsdaten in Tabellen (Relationen) ab. Eine Tabelle besteht aus einer festen Anzahl von Spalten (Attribute) und einer variablen Anzahl von Zeilen (Tupel).

Für eine Tabelle gelten folgende Regeln.

- Ein Inhalt soll nur einmal an einer Stelle gespeichert sein. In einer Tabelle darf es keine doppelten Zeilen geben.
- Die Reihenfolge, in der Zeilen gespeichert werden ist nicht definiert. Die Reihenfolge / Sortierung entsteht lediglich bei der Abfrage der Daten.
- Die Reihenfolge der Spalten ist nicht definiert, d.h. eine Spalte kann nur über ihren Namen, nicht über eine Position angesprochen werden.
- Werte in einer Spalte müssen atomar sein, d.h. es dürfen keine Wiederholgruppen vorhanden sein

Um diesen und weiteren Bedingungen zu genügen, müssen die Nutzdaten umgeformt und auf verschiedene Tabellen aufgeteilt werden. Der Vorgang der Umformung der Nutzdaten entsprechend den Regeln relationaler Systeme wird Normalisierung genannt.

Ist die Datenmodellierung vollzogen, werden die Tabellen definiert, d.h. Namen für die Tabelle und ihre Spalten festgelegt, jeder Spalte ein Datentyp zugeordnet, Schlüssel und Bedingungen definiert. Die Anzahl der Spalten in einer Tabelle und der Datentyp der Spalte sind also festzulegen, bevor Nutzdaten abgelegt werden. Wird eine Information eingetragen, muß ausreichend Platz vorgesehen sein. Als Beispiel soll eine Sammlung von Adressdaten dienen.

Zuerst erhalten die Personen zur Identifikation einen eindeutigen Schlüssel. Die Information zu Telefon wird abgetrennt. Enthält eine Adresstabelle nur ein Telefonfeld, wird es bereits bei Fax zu eng. Umgekehrt wird Platz verschenkt, wenn der maximale Bedarf definiert wird, dieser aber nur in einem Bruchteil aller Fälle notwendig ist.

Ident	Name	Str	PLZ	ORT
1	M.Oswald	Heinzelstr. 6	44556	Oberdorf
2	A.und H. Gerns	Bachgasse 12	44557	Unterdorf
3	Fa. Nägele	Industriestr. 224	44550	Hauptdorf
4	Silvia Nägele	Schönblick 4	44550	Hauptdorf
5	H.Müller	Windenhof 44	44560	Altbach
9	A.Grieser	Trompetenstr. 23	23562	Niederstätt

(Tabelle Person)

Der Hauptschlüssel (primary Key) der Haupttabelle (master table) taucht als Fremdschlüssel (foreign Key) in der abgetrennten Telekom Tabelle wieder auf. Da die Einträge nun separiert und anders angeordnet wurden, ist es problemlos möglich die Handynr von A.Grieser zu speichern, ohne die Tabellenstruktur zu ändern. Denkbar wäre auch unter Bezeichnung eine email Adresse aufzunehmen.

FKEY	BEZ	NR
1	Tel	07896 6678
2	Tel	07896 7788
3	Tel	07896 102030
3	Fax	07896 102111
4	Tel	07896 1020
5	Tel	07899 33214
9	Tel	03451 42890
9	Handy	0171 667097123

(Tabelle Telekom)

Um alle Namen von Faxbesitzern aus Hauptdorf zu erfragen, müßte nun nicht mehr ein Programm geschrieben werden, das zunächst eine Tabelle öffnet, diese durchliest, sich entsprechende Schlüssel merkt, mit den gemerkten Schlüsseln die nächste Tabelle öffnet und die betreffenden Einträge heraussucht. Eine SQL Anweisung, die die Ergebnismenge definiert sähe so aus:

```
Select p.name, t.nr
  From person p, telekom t
 Where p.ort = ´Hauptdorf´
   And p.ident = t.fkey
   And t.bez = ´FAX´;
```

Die logische Zusammengehörigkeit der Inhalte / Nutzdaten wird durch Definition von Schlüsseln, Regeln und Abfragen hergestellt. Eine Abfrage, die mehrere Tabellen über deren Schlüssel verknüpft, heißt JOIN. Wird eine Abfragedefinition in der Datenbank dauerhaft gespeichert, wird von einer VIEW gesprochen. Mit VIEWS ist es möglich Teile einer Tabelle für die Endbenutzer quasi unsichtbar zu machen, indem man den Benutzern nur Nutzungsrecht an der View, nicht aber an der Tabelle gewährt.

2.2.2 Aufgaben von RDBMS

Relationale System unterstützen bei Anlage, Löschen und Ändern von Tabellendefinitionen. Auch Beziehungen zwischen Tabellen können definiert werden. So kann z.B. festgelegt werden, daß bei Löschung einer Person in der abhängigen Tabelle alle zugehörigen Einträge gelöscht werden sollen, oder daß die Löschung der Person verweigert wird, wenn abhängige Sätze vorhanden sind. (Das ist einsichtig für Personen, die noch offene Rechnungen in einem System aufweisen!) Mit diesen Regeln wird verhindert, daß verlorene Datensätze in abhängigen Tabellen stehen bleiben, für die es keinen Einstiegspunkt mehr in einer Master Tabelle gibt. Die Gewährleistung eines gesicherten Zugriffs auf vorhandene Daten wird mit dem Begriff referentielle Integrität bezeichnet.

Aktuelle Systeme erlauben es Funktionen anzulegen, die ausgeführt werden, wenn ein bestimmtes Ereignis eintritt.

- Trigger
- Funktionen und Erweiterbarkeit
- Zeitabläufe / Jobs
- Benutzer und Rechte

2.2.3 Schnittstellen zu relationalen Systemen

2.2.3.1 CALL-Level APIs

Datenbankverwaltungssysteme bestehen in der Regel aus einer Sammlung von Programmen, die untereinander kommunizieren und Arbeitsergebnisse austauschen können. Die Offenlegung der Aufrufmöglichkeiten dieser internen Funktion und deren Parametrisierung bildet die API-Level Schnittstelle, die aus Benutzerprogrammen angesprochen werden kann. Diese Schnittstellenfunktionen sind herstellerabhängig, d.h. nicht auf andere Datenbanksysteme übertragbar. API-Level Funktionsaufrufe werden aus 3GL Programmen angesprochen, nach dem Compile mit einer Laufzeitbibliothek des Datenbanksystems gelinkt. Der Lernaufwand für API-Level Programmierung ist hoch. Manche Hersteller von DB Systemen bieten deshalb eine leichter zu erlernende Programmiersprache an, die durch einen Precompiler in z.B. C-Programmcode umgewandelt werden kann, der dann die entsprechenden API-Aufrufe enthält.

2.2.3.2 SQL und embedded SQL

Als normierte Abfragesprache im Bereich der relationalen Systeme hat sich die Structured Query Language (SQL) durchgesetzt. Sie ist herstellerunabhängig. Da sie ergebnis- und mengenorientiert aufgebaut ist, fehlen prozedurale Konstrukte, wie Schleifen und Verzweigungen. Embedded SQL beschreibt demgegenüber die Möglichkeit SQL - Abfragen in prozedurale Sprachen einzubetten. Dabei werden SQL Befehle in den Programm Quellcode eingetragen. Die Codezeilen enthalten spezielle Schlüsselwörter:

```
EXEC SQL CONNECT :vcUserId IDENTIFIED BY :vcPassword USING :vcConnect;
```

Die Variablen müssen zuvor in einem Programmabschnitt für den Precompiler definiert worden sein.

```
EXEC SQL BEGIN DECLARE SECTION;
char  vcUserId[30];
char  vcPassword[30];
char  vcConnect[30];
EXEC SQL END DECLARE SECTION;
```

Vor der Übersetzung des Programms durch den Compiler werden die EXEC SQL Anweisungen durch einen Precompiler umgesetzt in entsprechende Api-Level Calls.

2.2.3.3 ODBC

Open Database Connectivity (ODBC) ist eine Schnittstellendefinition für den Zugriff auf proprietäre Datenbank APIs. ODBC wurde zu Beginn der neunziger Jahre von Microsoft entwickelt. ODBC legt auf die Call Level APIS eine weitere Schicht, die es erlaubt, in standardisierte Form mit einer Datenquelle Verbindung aufzunehmen, Abfragen an die Datenquelle zu richten und die Abfrageergebnisse entgegen zu nehmen.

Ein Treibermanager verwaltet die Treiber der verschiedenen Datenquellen. Der Treibermanager dient sowohl der Definition und Verwaltung von Datenquellennamen und Eigenschaften bei administrativen Tätigkeiten, als auch als Vermittler zu Anwendungsprogrammen.

Ein Administrator legt zunächst einen Datenquellennamen fest und ordnet diesem einen Treiber zu, bzw. legt die einstellbaren Eigenschaften fest. Will ein Anwendungsprogramm Verbindung zu einer Datenbank aufnehmen, wendet es sich mit dem Datenquellennamen an den Treiber-Manager. Dieser lädt den zugehörigen Treiber und stellt die Verbindung her. Die Anwendung kann nun mithilfe von SQL Befehlen Daten in der Datenbank manipulieren.

ODBC hat sich zu einem de facto Standard für Datenbankzugriffe entwickelt, da viele Anwendungsprogramme nun ihren Datenbankzugriff unabhängig von der später eingesetzten Datenbank definieren konnten und auch fertige Anwendungsprogramme, wie Textverarbeitung und Tabellenkalkulation Daten aus Datenbanken benutzen konnten.

Mußten ODBC Funktionen ursprünglich in C, C++ geschrieben werden, begann Microsoft sehr rasch damit, die eigenen Programmiersprachen mit komponentenorientierten Kapseln zu umkleiden. In Visual C++ und VB konnte man die DAO (Data Access Objecte) nutzen. Weitere Schnittstellen wurden obendrauf gepackt. Die neueste heißt ADO(ActiveX Data objects), sie integriert ODBC als eine Quelle unter vielen. Anlaß für diese Entwicklung war u.a. die Mängel in der Netzwerkfähigkeit von ODBC. Jeder Client muß lokal über einen installierten ODBC-Treiber, d.h. auch über einen Treiber-Manager verfügen.

2.2.3.4 JDBC

Zeitmangel und Erfolgsdruck führten dazu, daß die Java Database Connectivity (JDBC) sich eng an ODBC anlehnt. Wie bei ODBC wird auch hier ein Treibermanager verwendet, der die entsprechenden Treiber verwaltet und für Anwendungsprogramme aufruft. JavaSoft unterteilt die Treiber in vier Typen:

1. **JDBC-ODBC Bridge** (Typ-1-Treiber) Übersetzen JDBC-Aufrufe in ODBC-Aufrufe. Beim Client müssen ODBC-Treiber und Datenbankbibliotheken installiert sein, das gilt auch für Netzwerkkomponenten.
2. **Java to Native Api** (Typ-2-Treiber) Übersetzen JDBC-Aufrufe in Api-Level calls der jeweiligen Datenbank. Beim Client müssen die Datenbankbibliotheken für Api-Calls installiert sein.
3. **Java to Proprietary Network Protocol** (Typ-3-Treiber). Die Treiber sind in Java geschrieben und können zum Client heruntergeladen werden. Der Client muß nun eine Verbindung mit einem Serverprozess aufnehmen (Application Server, RMI...), der die JDBC-Aufrufe in Api-Level Aufrufe umsetzt. Diese Treiber sind herstellerunabhängig und werden meist von Drittanbietern hergestellt.
4. **Java to Native Database Protocol** (Typ-4-Treiber/thin driver) Die Treiber kommunizieren direkt mit dem Datenbankserver, dessen Api sie direkt aufrufen können. Sie können zum Client heruntergeladen werden. Diese Treiber bieten schnelle Zugriffsgeschwindigkeiten, sind jedoch wieder abhängig vom Datenbankhersteller.

2.3 Objektorientierte Systeme

Objektorientierte Datenbanksysteme etablieren sich seit Beginn der 90er Jahre nur zögerlich. Sie speichern Datengebilde, die mit objektorientierten Programmen erzeugt werden scheinbar unverändert ab, d.h. die Objekte werden zur Speicherung nicht in tabellengängige Teile aufgespaltet. Ziel der Anstrengungen im Bereich OODB ist, es transiente und persistente Objekte möglichst gleichwertig zu behandeln, so daß keine spezielle Programmlogik gebraucht wird, um Objekte persistent zu machen.

Gegenüber RDB, die Datenverknüpfungen auf wertbasierten Schlüsseln realisieren, ist die Objektidentität ein Grundpfeiler objektorientierter Datenbanksysteme. Objektidentifikatoren (Surrogate) werden vom Datenbanksystem generiert und sind vollständig unabhängig von Objekteigenschaften. Die Fähigkeit eines Objekts, Listen oder Mengen von OIDs (Objekt Identifikatoren oder Objekt Ids) als Elemente zu halten, ermöglicht komplexe Objektverbindungen, die unter relationalen Systemen nicht oder nur mit erheblichem Aufwand zu realisieren sind. Hieraus re-

sultiert die Forderung an ein Objektverwaltungssystem, Referenzsicherheit zu bieten. Ähnlich der Sicherung der referentiellen Integrität in relationalen Systemen, sollte eine OODBMS die Gültigkeit von Objektverknüpfungen über Einfüge-, Lösch- und Änderungsaktionen hinweg gewährleisen. Diese Forderung wird von den marktgängigen OODBMS in unterschiedlicher Weise erfüllt.

Die Menge aller Objekte einer Klasse wird Extent genannt. Ein Extent entspricht funktional einer Tabelle. So wie der Name einer Tabelle oder einer View in einem relationalen System einen benannten Einstiegspunkt in die Datenbank darstellt, so wird häufig der Name eines Extents als Einstiegspunkt für die Abfrage in einem oo-System benutzt. Die Namen müssen eindeutig sein und können auch für einzelne persistente Objekte vergeben werden. Diese Einstiegspunkte werden häufig dbroot genannt.

Standardisierungsbemühungen im Bereich Objektorientierter Systeme werden von der Object Management Group (OMG)[1], bzw. speziell für Datenbanken von der Object Database Management Group (ODMG)[2] vorangetrieben. Für die Funktion der Schemadefinition wird Object Definition Language (ODL), für die Abfrage von Informationen aus Objektmengen Object Query Language (OQL) verwendet.

Die Verwendung von ODL geschieht entweder aus Werkzeugen, die der DB-Hersteller anbietet, oder aus einer speziellen Sprachanbindung. So soll es möglich sein, zur Definition und Bearbeitung von persistenten Daten lediglich eine einzige Sprachumgebung verwenden zu können.

Abfragen gegen eine OODB können mit Object Query Language durchgeführt werden. Ähnlich SQL kann die Abfragesprache frei oder eingebettet in Programmcode verwendet werden. Auch die Syntax entspricht in ihrem schematischen Aufbau der Struktur einer SQL-Select Anfrage:

`Select` <Resultatmenge> `from` <Objektmenge> `where` <Eigenschaften der gesuchten Objekte>

Das Resultat einer OQL-Abfrage kann ein Wert, ein Objekt oder eine Sammlung von Werten sein. OQL verletzt an manchen Stellen die strenge Objektorientierung, so ist es möglich auf Attribute, die private deklariert sind, lesend zuzugreifen.

OQL kann, im Unterschied zu SQL jedoch keine Datenmanipulation durchführen, sondern beschränkt sich auf die Abfrage. Die Möglichkeit Objektmethoden aus OQL-Abfragen auszuführen, ist jedoch erlaubt. Zur freien Datenmanipulation müssen Anwendungsprogramme geschrieben werden. Objektverknüpfungen können in OQL verwendet werden, d.h. navigierender Zugriff wird unterstützt.

Schnittstellen wie SQL, ODBC oder JDBC gibt es nicht, d.h. sie sind auch nicht nötig, da die Datenbank APIs, als Call-Level APIs in einer objektorientierten Programmiersprache definiert sind, den sogenannten Language-Bindings[3]. Gebräuchliche Zugriffssprachen sind in diesem Umfeld Smalltalk, C++ und JAVA. Aktuell wird bei der ODMG an der Definition eines XML-Bindings gearbeitet.

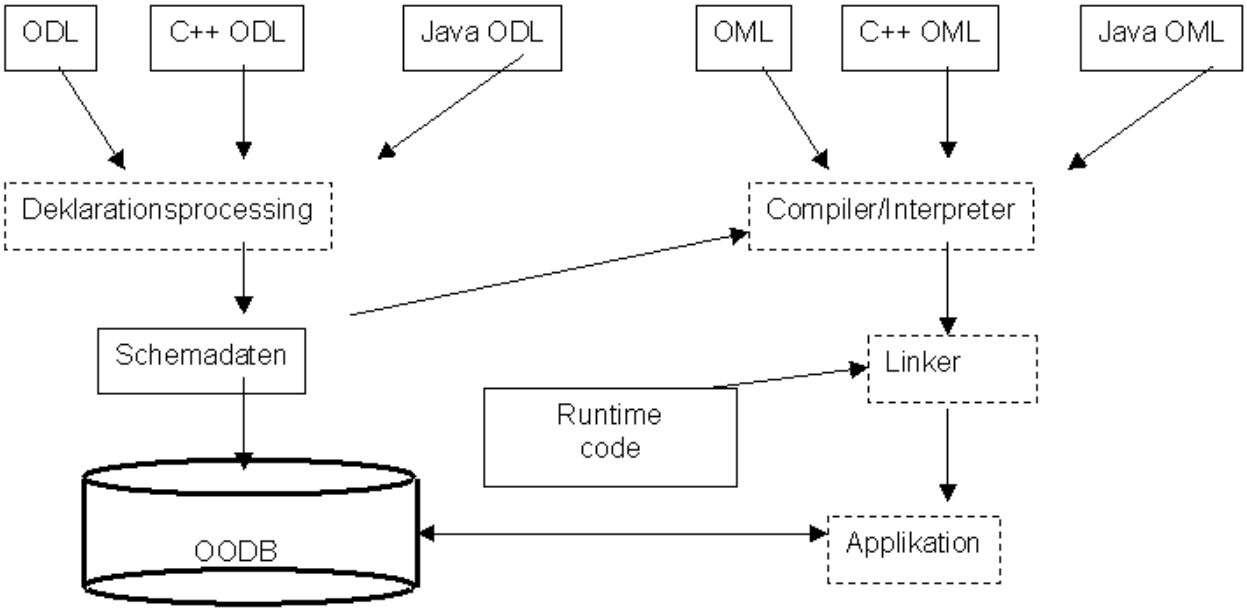
Die Definition eines Sprachbindings befreit die Programmierungdavon, über Fragen der Datentypenabbildung selbst nachdenken zu müssen. Entsprechend den Vorgaben der ODMG, enthält ein Programm keine Informationen darüber welche Klassen persistent, bzw. transient sind.

Im Fall des Java Bindings bedeutet dies, daß der Programmcode erstellt wird, ohne Speichermechanismen zu definieren. Nach der Übersetzung des Java Quellcodes in Bytecode, wird mit einem Postprozessor eine Markierung in die .class Dateien eingetragen, die die Persistenzfähigen Klassen kennzeichnet und entsprechend erweitert. Die Definition des JAVA Binding besteht aus folgenden Klassen und Schnittstellen

- Datenbankklassen (Transaction, Database, Query)
- Sammlungsklassen und Schnittstellen (Set, Bag, List, Array)

- Exception Klassen

Sofern nicht explizit mit Schlüsselwort transient gekennzeichnet, werden alle Benannten Objekte und alle von diesen abhängigen und durch Namen oder OID erreichbaren Objekte dauerhaft in der Datenbank gespreichert. Das Java-Binding realisiert Persistenz durch Erreichbarkeit.



Das Zusammenspiel von Schemadefinition und Erstellung von Anwendungsprogrammen ist in obiger Grafik dargestellt.

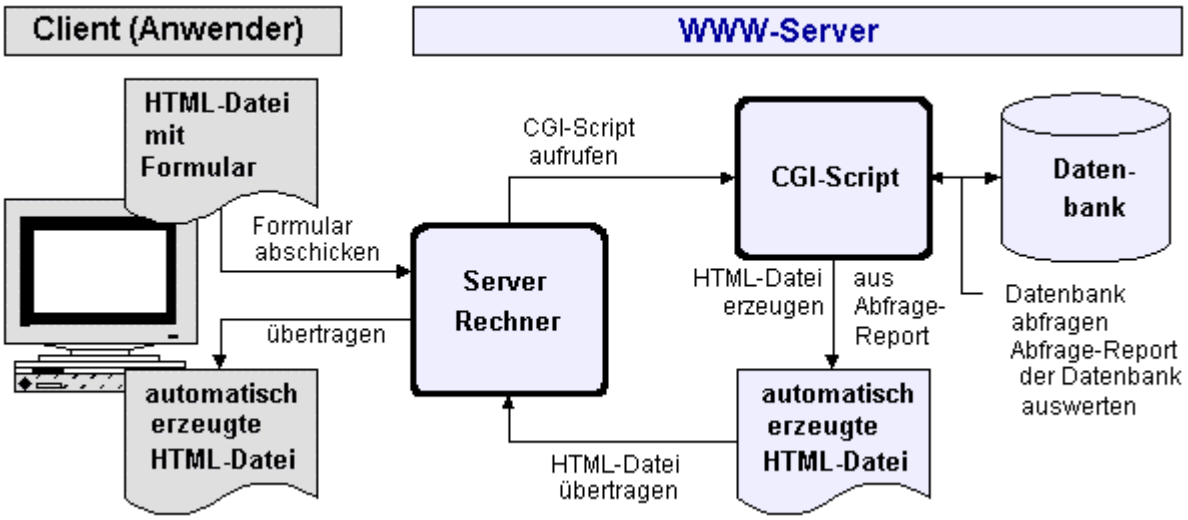
Die, wenn auch zögerliche Verbreitung von OODB Systemen hat in den letzen zwei Jahren sehr stark auf die Weiterentwicklung der klassischen relationalen Produkte zurückgewirkt. Alle großen Hersteller haben damit begonnen, objektorientierte Erweiterungen in ihre Produkte zu integrieren, d.h. in der Regel handelt es sich um eine Flexibilisierung des Typsystems, die Möglichkeit, Mengen in Tabellenfeldern abzulegen und die Fähigkeit eines Systems, eindeutige Objektidentifikationen (OID) zu generieren.

3 Web-Server Erweiterungen

Standardaufgabe von Web-Servern ist das Bearbeiten von HTTP Anfragen und das Rücksenden von HTML Dokumenten. Die Bearbeitung von Anforderungen, die zusätzliche Dienste in Anspruch nehmen werden nicht durch den Web-Server selbst erledigt, sondern müssen über Server-Erweiterungen und Schnittstellen zu verfügbaren Diensten realisiert werden.

3.1 Common Gateway Interface

Das Common Gateway Protokoll beschreibt eine Programmschnittstelle zwischen einem Webserver und einem Anwendungsprogramm. Das Gateway Protokoll legt die Form der Übergabe von Parametern aus dem Webserver fest. Die Schnittstelle ist auf keine bestimmte Programmiersprache festgelegt. Der Sprachgebrauch "CGI-Script" resultiert aus dem Umstand, daß bislang vor allem Perl scripts oder shell-scripts als CGI Programme zum Einsatz kamen. Es können jedoch auch kompilierte Sprachen wie C eingesetzt werden.



Ablauf CGI-Verarbeitung, Quelle S.Münz: Selfhtml http:// www.teamone.de/selfaktuell

Relevant ist lediglich, daß die Parameter, die an das Programm übergeben werden entsprechend den Konventionen von CGI verarbeitet werden.

CGI Programme können über verschiedene Möglichkeiten aktiviert werden:

- einen URL
- eine Referenz oder Grafikreferenz
  ```
  <a href="/cgi-bin/counter.pl">Zaehler inkrementieren</a>.
  <img src="/cgi-bin/counter.pl">.
  ```
- über ein SSI
  ```
  <!-- #exec cgi="/cgi-bin/counter.pl" -->.
  ```
- oder über HTML-Formulare
  ```
  <form action="/cgi-bin/guestbook.pl" method="get">).
  ```

Wie HTML Dokumente können CGI-Programme mit den HTTP Methoden GET, darüberhinaus mit POST aufgerufen werden. Die beiden Methoden unterscheiden sich in der Art der Übergabe der Parameter.

Bei einem Aufruf mit GET werden die Parameter an den übermittelten URL angehängt, im Web-Server an die Umgebungsvariable QUERY_STRING übergeben. Die Parameterliste wird mit einem "?" vom URL abgetrennt und besteht aus Key-Value Paaren, die in der Form Key=Value angegeben werden. Mehrere Parameter werden durch "&" voneinander abgetrennt. Alle Zeichen größer als ASCII 128 und Zeichen, die als Steuerzeichen verwendet werden (also &, +, = und %) werden hexadezimal in der Form %80 codiert. Die Parameterliste darf eine je nach Browser verschiedene Maximallänge nicht überschreiten[4].

Ein Aufruf via POST übermittelt die Parameter als Teil der Nachricht und ist deshalb bei Parameterketten von unabsehbarer Länge vorzuziehen. Die übertragenen Daten werden über STDIN an das aufgerufene CGI-Programm übermittelt. Die Länge der Daten wird in der Umgebungsvariablen CONTENT_LENGTH abgelegt.

```
#-------------------------------------------
#parameter zerlegen
#-------------------------------------------
if ($ENV{'REQUEST_METHOD'} eq "GET")
{
    $Daten =  $ENV{'QUERY_STRING'};
}
else
{
    read(STDIN, $Daten, $ENV{'CONTENT_LENGTH'});
}
```

Ein Beispiel für die automatische Verarbeitung beider Request-Arten findet sich in obigem Auszug aus dem Script: "Klassisches Perl CGI-Script" im Anhang.

Die bekanntesten Nachteile von CGI sind schlechte Performance, umständliche Session-Verwaltung und Probleme mit persistenenten Informationen.

3.2 HTML - Präprozessoren

Um HTML-Dateien mit dynamischen Inhalten versehen zu können, wurden eine Reihe von Erweiterungen geschaffen, die als Bereiche in einer HTML-Datei mit speziellen Auszeichnungen kenntlich gemacht werden. Vor der Auslieferung solcher HTML-Dokumente durch den Web-Server, müssen die Auszeichnungen, die nicht dem HTML Standard entsprechen, durch Inhalte ersetzt und gegebenenfalls mit HTML-Tags ausgezeichnet werden.

In Produktionsteams, in denen Programmierung und Seitendesign auf verschiedene Personen verteilt sind, hat dieser Ansatz viele Vorteile. Seitendesigner können mit ihrem KnowHow verwerten, was die Programmierung an Funktionen zur Verfügung stellt.

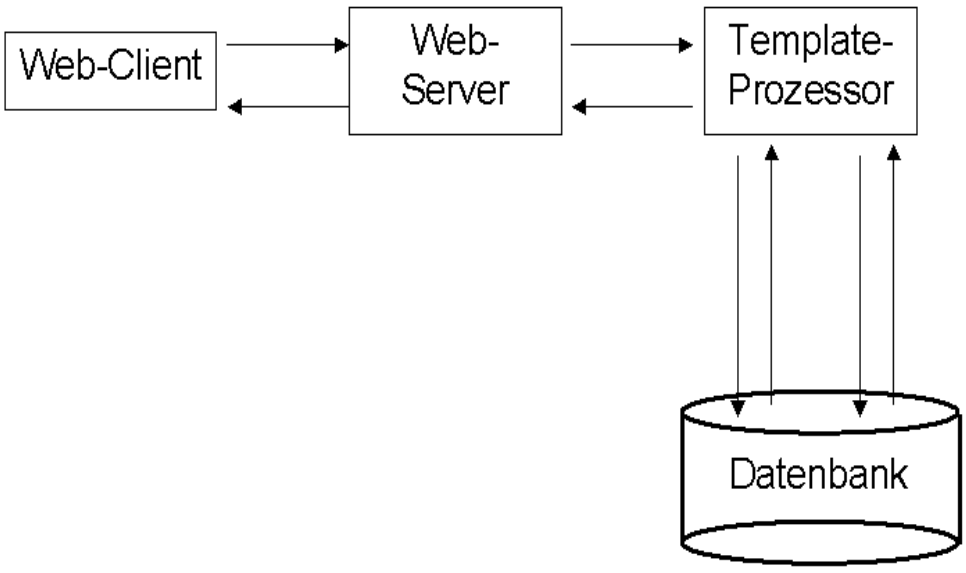
3.2.1 Server Side Includes - SSI

Ein früher Ansatz HTML dynamischer zu gestalten waren die Server Side Includes, die als HTML-Kommentare in das Dokument eingefügt werden. Web-Server, die SSI unterstützen, analysieren die Dokumente, die als SSI haltig gekennzeichnet wurden und werten die Anweisungen aus. Die Kommentare werden entfernt und durch die spezifizierten Inhalte ersetzt. In SSI Feldern können Umgebungsvariablen, weitere Dateien und der Aufruf von Programmen eingebettet sein.

3.2.2 HTML-Schablonen / Templates

Mit Templates werden HTML Dokumente bezeichnet, die Auszeichnungselemente enthalten, die von HTML Prozessoren nicht verstanden werden. Diese Erweiterungen müssen, ähnlich SSI, vor Auslieferung des Dokuments durch die spezifizierten Inhalte ersetzt werden. Dies wird durch zusätzliche Programme, oder durch in den Web-Server integrierte Module erreicht.

Der Aufruf eines Template Dokuments aus einem Browser wird vom Web-Server an das Template-Verarbeitungsprogramm weitergegeben. Dieses Modul wertet die erweiterten Tags aus und gibt den entstandenen Inhalt in ein HTML Dokument, welches wieder dem Web-Server zur Auslieferung bereitgestellt wird.



Alle nachfolgend genannten Lösungen, sind nur bedingt auf andere Systemplattformen oder andere Web-Server übertragbar. Die Abgrenzung zwischen Produkten, die lediglich Templates auswerten und Scripting Umgebungen, die die Nachteile von klassischem CGI vermeiden, ist schwer. Alle Produkte - vielleicht mit Ausnahme der Netscape Lösung - entwickeln sich in Scripting Umgebung.

3.2.3 Serverseitiges Java Script von Netscape

Eine proprietäre Erweiterung des Netscape Servers stellt **serverseitiges Java-Script** dar. Ausgangsmotiv von Netscapes Entwicklung war es, Verbindungen mit relationalen Datenbanken zu realisieren. Explizit unterstützt wird die Verbindung zu Informix, Oracle, Sybase und allen ODBC fähigen Datenbanken. Die serverseitig auszuführenden Anweisungen werden von den Tags <SERVER>.....</SERVER> umschlossen. Diese Anweisung darf mehrfach im HTML-Dokument vorkommen. Ein kurzes Code Beispiel[5] findet sich im Anhang.
Nach der Übernahme der Netscape Server Entwicklung durch Sun, ist zu erwarten, daß die serverseitige Unterstützung für Java-Script zugunsten von serverseitigem Java nicht weiter entwickelt wird.

3.2.4 ColdFusion

Ein von Webserverherstellern unabhängiges, kommerzielles Produkt ist **ColdFusion 4.0** von Allaire Software. ColdFusion benutzt eine eigene ColdFusion Markup Language (CFML), die vom ColdFusion Application Server verstanden wird. Der Server ist ein selbständiges Produkt, das unter Windows NT, neuerdings auch unter Solaris/Unix verfügbar ist, Portierungen für Linux sind für 1999 angekündigt. Die Verbindung zwischen Web-Server und ColdFusion Server wird über Server APIs (unterstützt ISAPI, NSAPI, Apache), oder über CGI realisiert.
CF ist als Webtool weit verbreitet, da CFML als HTML ähnliche Sprache leicht zu lernen ist. CFML Tags sind erweiterbar, und Allaire bietet in einer riesigen Tag Gallery hunderte von Custom Tags an, die zum Teil von den ColdFusion Kunden erstellt und kostenfrei zur Verfügung gestellt wurden.
Zu ColdFusion 4.0 gehören der Server, ColdFusion Studio (Visuelle Entwicklungsumgebung) und ColdFusion Extensions.

3.2.5 PHP

PHP wird oft als Konkurrent zu ColdFusion genannt, da PHP3 und die Vorversion PHP/FI ebenfalls große Verbreitung gefunden hat. PHP ist freie Software, verfügbar für alle Unix Plattformen und Windows NT. Linkbare Module existieren für Apache Web-Server unter Unix. Besonders diese Variante ist für kurze Ausführungszeiten bekannt[6].

PHP kann wie ein Template verwendet werden, d.h. PHP-Code wird in Form von HTML-Kommentaren in eine Seite eingebettet. Daneben kann es jedoch auch wie ein CGI-Programm verwendet werden, d.h. die HTML-Seite kann auch vollständig generiert werden. In HTML-Formularen definierte Eingabefelder können in einem weiterverarbeitenden PHP-Programm entweder über die traditionelle Parameterzerlegung gemäß CGI angesprochen werden, oder

direkt mittels des Feldnamens, der als Programmvariable zur Verfügung steht.
Über Inclulde-Befehle können Programmteile oder eigene Funktionsbibliotheken eingebunden werden.

```
<?php_track_vars;
  include "makefield.inc";
  $conn = mk_genericEdit($psqlcnn, "updatetable.phtml", $argv[0]);
?>
```

Für alle wichtigen Datenbanken existieren eingebaute Schnittstellenfunktionen, die direkt, ohne externe Treiber verwendet werden können, sofern sich die Datenbank auf der selben Maschine befindet. Die Sprache ist eine Mischung aus C und PERL, Variablen werden dynamisch erzeugt und haben keine Typenbindung. Selbst in Zeichenketten können Variablen verwendet werden. Nachfolgende Zeilen verwenden Variablen für den Tabellennamen und die Identnummer des Datensatzes zur Erzeugung eines Query-Strings für eine PostgreSQL Datenbank. Das Abfrageergebnis liegt als zweidimensionaler Array vor, der über Spaltennamen oder Nummern die Werte zugänglich macht.

```
$qstr   = "SELECT * FROM $basetname where ident = $itemident ;";
$resultdata = pg_Exec($conn, $qstr);
```

Die Funktionsbibliothek von PHP ist riesig und erweiterbar. PHP ist jedoch bislang eine reine Programmierumgebung, die keine graphischen Werkzeuge, wie etwa ColdFusion anbietet.

3.3 Active Server Pages (ASP)

Active Server Pages (ASP) ist Template Lösung und Scripting Umgebung des Microsoft Internet Information Servers. Realisiert als dynamisch ladbare Module, findet die Verarbeitung in der Laufzeitumgebung des Servers statt. Als Scriptsprache können verschiedene Programmiersprachen verwendet werden. ASP definiert einen Rahmen, in den Scripting-Engines eingebettet werden können. Mitgeliefert werden die Interpreter für `JScript` und `VBScript` in Form von serverseitigen ActiveX-Komponenten, als Standard Scriptsprache wird `VBScript` benutzt. Es ist auch möglich verschiedene Script-Sprachen innerhalb einer ASP - Seite zu verwenden.
Unter einer ASP-Anwendung wird eine Menge von Seiten, die in einem virtuellen Verzeichnis liegen verstanden.
Zur Kennzeichnung der ASP-Tags wird `<% Anweisung, Anweisung %>` verwendet.
ASP stellt fünf eingebaute Objekte zur Verfügung.

- Das Application Objekt enthält Zustandsinformationen für die Anwendung. Dieses Objekt wird vom System automatisch beim ersten Aufruf einer Anwendung erzeugt, d.h. sobald eine Seite aus einem virtuellen Verzeichnis aufgerufen wird. Das Objekt terminiert beim Stop des Servers.
- Das Session Objekt enthält Zustandsinformationen über die jeweilige Benutzersitzung. Hier könnten z.B. Warenkörbe, oder user tracking Informationen abgelegt werden. Ein Session Objekt wird für jede aktive Benutzer session erzeugt. Terminiert werden sie über einen timeout (default 20 Minuten), oder scriptgesteuert.
- Das Request Objekt stellt alle Informationen zur Verfügung, die von einem Browser Request übermittelt wurden.
- Das Response Objekt enthält Methoden, mit denen die Anwort an den Client gesteuert werden kann.

- Das Server Objekt dient als Schnittstelle zum Aufruf von Server Methoden.

Der Zugriff auf die vordefinierten Objekte erfolgt aus einem Programmscript über den Klassenbezeichner.

Neben den internen Objekten stehen fünf vordefinierte serverseitige ActiveX Komponenten zur Verfügung, die unter anderem ein Database-Access-Objekt, eine File-Access- und eine Content-Linking Komponente enthalten. Content-Linking erlaubt die Definition einer Abfolge von Seiten, die nicht in den Seiten verankert sein muß. Damit unterstützt die Microsoft Lösung die Wiederverwendung von Seiten in anderen Zusammenhängen oder Abläufen.

3.4 Servlets: Serverseitiges JAVA

Servlets sind in Java geschriebene Module, die generell als Server Erweiterung dienen. Der Einsatz ist dabei nicht grundsätzlich auf die Erweiterung von Web-Servern beschränkt. JavaSoft gab 1997 die Servlet Spezifikation heraus und inzwischen gibt es Module oder Zusätze für alle wichtigen Webserver, um Java Servlets auszuführen[7]. Die Servlets selbst laufen dabei in einer JAVA-VM und sind damit so portabel wie Java.

Servlets wurden auch als Ersatz für CGI konzipiert, um die vielen Probleme der klassischen CGI-Programmierung zu beheben. Sie können jedoch auch wie wie SSI verwendet werden.

Serverseitig läuft eine Java VM, die ein aufgerufenes Servlet lädt, weitere Request werden an Threads verteilt, d.h. es werden keine weiteren Prozesse gestartet.
Ein gestartetes Servlet bleibt so lange im Speicher, bis das Programm geändert, oder der Server heruntergefahren wird. Dadurch wird die Ausführung beschleunigt. D.h. einmal geladen ist ein Servlet immer bereit den nächsten Aufruf zu bearbeiten. Aufgrund dieser Tatsache, können auch Informationen im Server gehalten werden, die einen User Request überdauern, d.h. das Session handling wird mit Servlets sehr viel einfacher und von Hintergrunddatenbanken unabhängiger. Mit dieser Persistenzfähigkeit sind Anwendungen, die einen Warenkorb implementieren oder einen Connection Pool zu einer Datenbank verwalten sehr viel einfacher zu realisieren.

Datenbankzugriffe können über JDBC oder objektorientierte Kommunikationsobjekte wie Object Request Broker (ORB) oder Remote Method Invocation (RMI) realisiert werden. Bei Objektorientierten Datenbanken kann das JAVA Binding der Datenbank benutzt werden, um die Datenbank zu manipulieren.

Zur Clientseite hin, kann traditionell HTML Code generiert werden, oder als Alternative eine Applet-Servlet Kommunikation aufgenommen werden.

Servlets können auf jeder Plattform erstellt werden, auf der ein JDK und ein JSDK installiert ist. Nach dem coding wird das Servlet mit dem Java compiler in Bytecode übertragen und in das vorgesehene Servlet - Verzeichnis auf dem Server kopiert.

4 Clientseitige Möglichkeiten

4.1 ActiveX

Mit ActiveX bezeichnet Microsoft Programmkomponenten, die durch Einbettung in Benutzerprogramme, die als Container fungieren wiederverwendet werden können. ActiveX wurde 1996 entwickelt und hat seitdem Funktionen aus Object Linking and embedding übernommen. ActiveX ist an die Windows Plattform als Ausführungsort gebunden.

Im WWW werden ActiveX Controls durch <OBJECT....> Tags in HTML Seiten gekennzeichnet. Ist die hier bezeichnete Komponente auf dem lokalen Rechner vorhanden, wird sie ausgeführt, ist sie nicht vorhanden wird sie von der unter codebase angegebenen URL heruntergeladen. Die Container-Anwendung ist in diesem Fall der Browser, der Fähigkeiten, die vom ActiveX Control angeboten werden benutzen kann.

ActiveX hat eine traurige Berühmtheit aufgrund der einhergehenden Sicherheitsrisiken erlangt. ActiveX Controls sind ausführbarer Binärcode für Windows Plattformen. Damit besteht die Möglichkeit beliebige Aktionen auf der ausführenden Maschine durchzuführen. Dies kann von Manipulationen im Dateisystem, bis zu Systemneustart alles sein[8]. Für die Realisierung einer Datenbankverbindung könnte z.B. ein remote Procedure Call auf eine Server Datenbank erfolgen.

Microsoft begegnete diesem Mißstand mit der Einführung von Zertifikaten, d.h. Echtheitsbestätigungen. Der empfangende Browser prüft die Garantie, daß das ActiveX Steuerelement von einem vertrauenswürdigen Server kommt.

Vorteil des ActiveX Ansatzes ist, daß die Funktionen der Windows Oberfläche genutzt und zur Entwicklung die visuellen Programmierumgebungen aus dem Hause Microsoft verwenden werden können. Als Binärcode bietet ActiveX darüberhinaus sehr viel schnellere Ausführungsgeschwindigkeit als die konkurrierenden Java-Applets.

4.2 Java Applets

Applets sind Java Programme, die im Bytecode auf einen Browser übertragen und auf der dortigen VM ausgeführt werden. Ursprüngliche Intention von Applets ist es, die Belastung des Netzwerks zu reduzieren, die entsteht, wenn permanent Informationen vom Server nachgeladen werden müssen, weil HTML keine aktiven Elemente enthält. So wurden und werden Applets häufig verwendet um animierte Bildchen im Browser ausführen zu lassen, d.h. heutzutage zumeist Werbeinformation zucken und hüpfen zu lassen. Applets können jedoch auch ganze Applikationen enthalten.

Beim Öffnen einer HTML-Seite, die einen Applet-Tag enthält, wird die angegebene .class Datei vom Server übertragen und im Java Interpreter des Browsers gestartet.

```
<APPLET CODE="Programm.class" WIDTH=400 HEIGHT=300>
<PARAM NAME="xyz" VALUE="zzzz">
Ihr Browser scheint keine Java Applets zu m&ouml;gen.
</APPLET>
```

Standardmäßig sind Applets untrusted, d.h. als nicht vertrauenswürdige Objekte, besitzen sie nur sehr eingeschränkte Zugriffsrechte auf lokale und entfernte Resourcen. Eingriffe in Dateien des lokalen Rechners sind ebenso unzulässig, wie die Eröffnung von Netzwerkverbindungen auf andere Server, als der von dem sie geladen wurden. Dieses restriktive Sicherheitskonzept spielt bei Datenbankverbindungen eine Rolle. Wurde ein Applet mit einem Typ4 JDBC-Treiber ausgestattet und in die Lage versetzt eine Datenbankverbindung zu öffnen, so bedeutet dies für untrusted Applets, daß sich die Datenbank auf der selben Maschine befinden muß, wie der Web-Server, der das Applet ausgeliefert hat. Soll die Datenbank von der Web-Server Maschine entkoppelt werden, muß eine Zwischenschicht als Applikationsserver hinzugenommen werden. Manche Datenbankhersteller bieten deshalb zusätzliche Connect-Programme an, die auf der Web-Server Maschine die Existenz der Datenbank vorspiegeln und Anfragen an die Datenbank weiterleiten.

Zertifikate und der, ab Version 1.2 von Java verfügbare Security Manager sollen diesem Mangel abhilfe Schaffen. Wie bei ActiveX stellt die vergrößerte Mächtigkeit eines Applets auch ein höheres Sicherheitsrisiko dar.

Verbindung zu Datenbanken kann über RMI, Corba, oder vermittelt über ein Servlet mithilfe von Objektserialisierung realisiert werden.

5 Anhang

5.1 Programmbeispiele

5.1.1 Klassisches Perl CGI-Script mit Zugriff auf dbm Dateien

```
#!/usr/bin/perl
#-----------------------------------------
# KMMT Telefonbuch sample 1999 fuer
# - Themen:
# -- Parameterzerlegung für get und post
# -- Zusammenarbeit Script und Formular
# -- persistente hash Daten
# -- Parameterencoding
#-----------------------------------------
#-----------------------------------------
#parameter zerlegen
#-----------------------------------------
if ($ENV{'REQUEST_METHOD'} eq "GET")
{
    $Daten =  $ENV{'QUERY_STRING'};
}
else
{
    read(STDIN, $Daten, $ENV{'CONTENT_LENGTH'});
}

if (defined $Daten)
{
    @felder = split(/&/,$Daten);
    foreach $fl (@felder)
    {
      ($n, $v) = split(/=/, $fl);
      $v =~ tr/+/ /;
      $v =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
      $v =~ s/<!--(.|\n)*-->//g;
      $params{$n} = $v;
    }#params stecken jetzt im hash params
}

$name = $params{'name'};
$nummer = $params{'nummer'};

#-----------------------------------------
# persistenten hash oeffnen
#-----------------------------------------
dbmopen(%telinfo, "telinfo", 0666);
if( $params{'cmd'} eq "update")
{
  $telinfo{$name} = $nummer;
}

if( $params{'cmd'} eq "delete")
{
```

```
   delete($telinfo{$name});
}
#-------------------------------------------
#  liste anzeigen
#-------------------------------------------
print "Content-type: text/html\n\n";
print "<html><head>\n";
print "<title>Telefon Liste</title> </head>\n";
print " <body bgcolor=#CCFFF> <h1>Telefonliste</h1>\n";

#-------------------------------------------
# dialog script per post aufrufen
#-------------------------------------------
print "<form action=\"/cgi-bin/teledit\" method=\"post\">\n";

print " <table>\n";

while(($name, $nummer) = each %telinfo)
{

  #-------------------------------------------
  # encoding für teledit get Aufruf!!!
  #-------------------------------------------
  $encname = $name;
  $encname =~ s/ /+/g;
  $encname =~ s/Ä/\%C4/g;
  $encname =~ s/Ö/\%D6/g;
  $encname =~ s/Ü/\%DC/g;
  $encname =~ s/ä/\%E4/g;
  $encname =~ s/ö/\%F6/g;
  $encname =~ s/ü/\%FC/g;
  $encname =~ s/ß/\%DF/g;

  print " <tr>\n";

  #-------------------------------------------
  # get Aufruf Dialog Script
  #-------------------------------------------
  print "<td><a href=\"teledit?cmd=update&name=$encname\">$name:</a></td>";
  print "<td> $nummer </td>";

  #-------------------------------------------
  # sich selber einen befehl geben
  #-------------------------------------------
  print "<td><a href=\"telmanager?cmd=delete&name=$encname\">entfernen</a></td>";
  print " </tr>\n";
}
#-------------------------------------------
# hashdatei wieder schliessen
#-------------------------------------------
dbmclose(%telinfo);

print " </table>\n";
print "<input type=submit value=\"Neuer Eintrag\">\n";
print "</form>\n";
print " </html>\n";


#!/usr/bin/perl
#-------------------------------------------
# KMMT Telefonbuch sample 1999 HTML Formular
#-------------------------------------------
```

```
if ($ENV{'REQUEST_METHOD'} eq "GET") {$Daten =  $ENV{'QUERY_STRING'}; }
else {read(STDIN, $Daten, $ENV{'CONTENT_LENGTH'}); }

if (defined $Daten)
{
  @felder = split(/&/,$Daten);
  foreach $fl (@felder)
  {
    ($n, $v) = split(/=/, $fl);
    $v =~ tr/+/ /;
    $v =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    $v =~ s/<!--(.|\n)*-->//g;
    $params{$n} = $v;
  }
#-------------------------------------------
#-Hash oeffnen und Wert raussuchen
#-------------------------------------------
  dbmopen(%telinfo, "telinfo", 0666);
  $name = $params{'name'};
  $nummer = $telinfo{$params{'name'}};
  dbmclose(%telinfo);
}

#-------------------------------------------
#-Formular generieren
#-------------------------------------------
print "Content-type: text/html\n\n";
print "<html><head>\n";
print "<title>TelefonEintrag</title> </head>\n";
print " <body bgcolor=#CCFFF> <h1>Telefonbuch Eintrag</h1>\n";

#-------------------------------------------
#- Hauptprogramm wieder aufrufen cmd in hidden field
#-------------------------------------------
print "<form action=\"/cgi-bin/telmanager\" method=\"post\">\n";
print "<input type=hidden name=\"cmd\" value=\"update\">\n";

#-------------------------------------------
# Tabelle mit Eingabefeldern erzeugen
#-------------------------------------------
print "<table>\n";
print "<tr>\n";
print "<td align=right>Name:</td>\n";
print "<td><input type=text name=\"name\" value=\"$name\" size=40 ></td>\n";
print "</tr>\n";
print "<tr>\n";
print "<td align=right>Nummer:</td>\n";
print "<td><input type=text name=\"nummer\" value=\"$nummer\" size=40></td>\n";
print "</tr>\n";
print "</table><br>\n";

print "<input type=submit value=\"Absenden\">\n";
print "<input type=reset  value=\"Abbrechen\">\n";

print "</form> </body> </html>\n";
```

5.1.2 Netscape Serverside Java-Script

```
<HTML>
<HEAD>
<TITLE>Department Data</TITLE>

<SERVER>
       // Connect the database COMPANYDATABASE on server MYSERVER
      database.connect("INFORMIX", "myServer", "informix", "informix", "CompanyDa-
tabase");
</SERVER>

</HEAD>
<BODY>

<TABLE BORDER=2>
<TR>
   <TH>Department No.</TH>
   <TH>Manager</TH>
   <TH>Location</TH>
   <TH>Annual Budget</TH>
</TR>

<SERVER>
      // Establish a cursor on the Department table
      deptCursor = database.cursor("select * from DEPARTMENT");


      // Iterate across every record in the result set
      // and write out selected fields in a table row.
      // Note that next() returns true as long as there are more rows
      while ( deptCursor.next() )
      {
            write("<TR>");
            write("<TD>" + deptCursor.deptNumber + "</TD>");
            write("<TD>" + deptCursor.manager + "</TD>");
            write("<TD>" + deptCursor.location + "</TD>");
            write("<TD>" + deptCursor.budget + "</TD>");
            write("</TR>");
       } deptCursor.close();
</SERVER>

</TABLE>

</BODY>
</HTML>
```

5.1.3 PHP Beispiel:

Beim Klick auf Neuanlage aus der zentralen Auswahl oder auf Bearbeiten aus einer Liste editierbarer Einträge.

```
<?php_track_vars;
  include "makefield.inc";
  $conn = mk_genericEdit($psqlcnn, "updatetable.phtml", $argv[0]);
?>
```

Die Funktion genericEdit erzeugt eine Eingabe oder Bearbeitungsmaske für einfache Tabellen.

```
/*-----------------------------------------------------------*/
function mk_genericEdit($psqlcnn,     // connectioninfo aus Cookie
                        $postaction,  // Eintrag in Post ACTION=
                        $tbshort,     // Kürzel des Tabellenname Kürzel
                        $itemident=0  // Ident eines Eintrags, nur bei EDIT
                        )
{
   $conn      = mk_connect($psqlcnn);
   $basetname = mk_getbasetname($conn, $tbshort);
   $pagetitle = "Daten Neuerfassung";
   if($itemident > 0)
   {
    $pagetitle = mk_getdocinfo($conn, $basetname, "HTML_STD_EDIT_LIST");
   }
   $hres = mk_createHeader($pagetitle);
   $hres = mk_setpostaction($postaction);

   echo "<input type=hidden name=tbname value='$basetname'>";
   echo "<table>";

   if($itemident > 0)
   {
      $result = mk_gettbmeta($conn, $basetname, "tbmetaall");
      $qstr   = "SELECT * FROM $basetname where ident = $itemident ;";
      $resultdata = pg_Exec($conn, $qstr);
      if (!$resultdata) {
         echo "An error occurred.\n";
         exit;
      }
   }
   else
   {
      $result = mk_gettbmeta($conn, $basetname);
   }

   $num = pg_NumRows($result);

   $i = 0;
   while ($i < $num)
   {
     if($itemident > 0)  /* mit contents  */
     {
       $trres = mk_maketrfield( pg_Result($result, $i, "anm" ),
                                pg_Result($result, $i, "tnm" ),
                                pg_Result($result, $i, "atln"),
                                pg_Result($result, $i, "atmod"),
                                pg_Result($resultdata, 0, $i));
     }
```

```
      else
      {
        $trres = mk_maketrfield( pg_Result($result, $i, "anm" ),
                                 pg_Result($result, $i, "tnm" ),
                                 pg_Result($result, $i, "atln"),
                                 pg_Result($result, $i, "atmod"));
      }
      $i++;
    }

    pg_FreeResult($result);
    pg_Close($conn);
    $trres = mk_maketrspeichern();
    echo "</table> </form> ";
    echo mk_html_href("kmmtserv.diff.uni-tuebingen.de/dbhelp.html", "Hilfe in neuem
Fenster &ouml;ffnen");
    echo "</BODY> </HTML>";

    return $conn;
}
```

5.1.4 ColdFusion Markup Beispiel

Das Template vor der Auswertung durch den CF Server

```
<CFQUERY NAME="ProductList"
DATASOURCE="CompanyDB">
SELECT * FROM Products
</CFQUERY>
<HTML>
<HEAD>
<TITLE>Product List</TITLE>
</HEAD>
<BODY>
<H1>Company X Products</H1>
<CFOUTPUT QUERY="ProductList">
#ProductName# $#Price# <BR>
</CFOUTPUT>
</BODY>
</HTML>
```

Das HTML Ergebis

```
<HTML>
<HEAD>
<TITLE>Product List</TITLE>
</HEAD>
<BODY>
<H1>Company X Products</H1>
Widgets $4.00 <BR>
Diggers $5.00 <BR>
Dingers $20.00 <BR>
Dongers $15.00 <BR>
</BODY>
</HTML>
```

5.1.5 Servlet mit JDBC zugriff und Sessiongebundenes Blättern

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstConnect extends HttpServlet {

public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException
{    doGet(req, res); }

public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException
{
  String url = "jdbc:postgresql://kmmtserv.diff.uni-tuebingen.de:XXX/dbname";
  String         usr = "xxxxx";
  String         pwd = "xxxxx";
  int maxdisplines   =  5;
  boolean morelines   = false;

  HttpSession session = req.getSession(true);

  res.setContentType("Text/html");
  PrintWriter out = res.getWriter();
  out.println("<HEAD><TITLE>Accessing PG</TITLE></HEAD><BODY BGCOLOR=#CCFFF>");
  out.println("<H1> Etwas zu Tabellen </H1>");

  Integer deja_lines = (Integer)session.getValue("test.deja_lines");
  if (deja_lines == null) // first Connect Session created
  {
     deja_lines = new Integer(0);
     session.putValue("test.deja_lines", deja_lines);
  }

  if (deja_lines.intValue() == 0)
  {
     Connection     db = null;
     Statement      st = null;
     ResultSet      rs = null;
     try
      {
        Class.forName("postgresql.Driver");
      }
     catch( Exception e)
      {
        out.println("<p> JDBC Treiber nicht gefunden <p>");
      }
     try
     {
        // Connect to database
        db = DriverManager.getConnection(url, usr, pwd);
        st = db.createStatement();
        ResultSet temp_rs = st.executeQuery("Select verfasser, titel from lit");
        session.putValue("test.the_rs", temp_rs);
      }
     catch( SQLException es )
      {
        out.println("<p> Connect oder Select misslungen <p>");
```

```
      }
    finally{ try{st.close();} catch(Exception e){}}
  }  // ende session und db-init

  ResultSet rs = (ResultSet)session.getValue("test.the_rs");

  out.println("<table border=1 cellspacing=2><tr>");

  try
  {
    ResultSetMetaData rsmd = rs.getMetaData();
    int cols = rsmd.getColumnCount();
    for(int i = 1; i < cols+1; i++)
    {
      out.println("<td><b>" + rsmd.getColumnLabel(i) + "</b></td>");
    }
    out.println("</tr>");
    morelines = rs.next();

    while(morelines)
    {
      out.println("<tr>");
      for(int i = 1; i < cols+1; i++)
      {
          Object o = rs.getObject(i);
          if(rs.wasNull())
             out.println("<td>{null}</td>");
          else
             out.println("<td>" + rs.getObject(i).toString()  + "</td>");
      }
      out.println("</tr>");

      morelines =rs.next();
      maxdisplines--;
      if(maxdisplines == 0)
      {
        session.putValue("test.deja_lines", new Integer(1));
        break;
      }
    } // end while
    out.println("</table>");
    rs.close();

    if(!morelines)  // no more lines
    {
     session.putValue("test.deja_lines", new Integer(0));
     out.println("<h3>nix mehr zu zeigen, beginne von Vorn!</h3>");
    }

   } // ende try
  catch( SQLException es )
   {
     out.println("<p> Connect oder Select misslungen <p>");
   }

 out.println("</BODY>");
 out.close();
}  // ende doGet
} // ende Class
```

5.2 Literatur und Informationsquellen

[Assfalg1998]
Assfalg, R, Goebels, U, Welter, H: Internet Datenbanken Konzepte, Methoden, Werkzeuge.Bonn, Addison-Wesley, ISBN: 3-8273-1230-2, 1998

Ein guter Überblick über Probleme, Schnittstellen und Werkzeuge im großen Feld des DB-publishing.

[Dicken1997]
Dicken, Hans: JDBC, Internet Datenbank Anbindung mit JAVA. Bonn, Thomson Publishing, ISBN: 3-8266-0343-5, 1997

Breiten Raum nimmt in diesem Buch eine allgemeine Einführung in das Thema und der Beispielcode ein. Die Codebeispiele beziehen sich auf den Zugriff auf eine ORACLE Datenbank und mSQL. Aufgrund des Erscheinungsdatums liegt ein Schwerpunkt auf JDBC-ODBC Bridge Treibern

[EILE1998]
Eilebrecht, Lars: Apache Web-Server; Bonn, International Thomson publishing, 2.Aufl. 1998.

Das Buch deckt ein breites Spektrum an Themen rund um den APACHE ab und gibt gute Anwendungsbeispiele für die besprochenen Themen. Die vorliegende zweite Auflage bietet einige Ergänzungen zur Version 1.3, orientiert sich jedoch an der UNIX Umgebung. Das Thema Apache unter Windows NT wird nur sehr randständig behandelt, serverseitiges JAVA nicht erwähnt. Das Buch ist das einzige gedruckte Werk, das die neue Autoconf Möglichkeiten im Anhang zu Apache 1.3 gut dokumentiert. Ein separates Kapitel geht auf frei verfügbare Indexsysteme, ein weiteres auf den Umgang mit Robots ein.

[Hall1998]
Hall, Marty: CORE Web Programming, PRENTICE HALL PTR CORE SERIES, Upper Saddle River, Prentice Hall PTR, ISBN: 0-13-625666-X, 1998

Lehrbuch mit 1279 Seiten für HTML, JAVA, CGI, JAVASCRIPT. CD-ROM mit Beispielen für Win95/NT und Macintosh.

[Hunter1998]
Hunter, Jason: JAVA Servlet Programming, Sebastopol, O'Reilly, ISBN 1-56593-391-X, 1998

Das Buch ist eine hervorragende Einführung ins Thema. Herausragend sind die Kapitel zu Session Trakking, Security und Database Connectivity. Die Beispielprogramme sind sehr praxisnah und können oftmals direkt eingesetzt werden.

[Kosafian1999]
Khoshafian, S., Dasananda, S., Minassian, N.: The Jasmin Object Database. Multimedia Applications for the Web. San Francisco, Morgan Kaufmann Publishers, ISBN: 1-55860-494-4, 1999

Für Entwickler eine gute Übersicht über die Funktionsweise und Schnittstellen (ODQL, C++, Java, WebLink) von Jasmine. Störend die unvermeidliche Debatte über Objektorientierung und fehlende Angaben zu Konfiguration des Systems. Dennoch bieten die Entwickler des Systems mit dem Buch einen guten Überblick über die Möglichkeiten und Eigenheiten (ODQL) von Jasmine. (Derzeit nur über den amerikanischen Buchhandel erhältlich)

[Laurie1997]
Laurie, Ben; Laurie, Peter: Apache - The Definitive Guide; California, O'Reilly & Associates, Inc,.1.Aufl. 1997.

Das Buch behandelt die Versionen 1.1 und 1.2 des Apache Webservers, damit lediglich die UNIX - Versionen. Themen wie Authentifizierung, Indexing, Proxy Funktion und Redirection werden ausführlich besprochen. Ein natürlicher Schwerpunkt liegt dabei auf Besprechung der Konfigurationsdateien. Zwei Kapitel widmen sich der Möglichkeit, eigene Apache module zu schreiben

[Meier1997]
Meier, Andreas, Wüst, Thomas: Objektorientierte Datenbanken, Ein Kompaß für die Praxis. Heidelberg, dpunkt.verlag, ISBN: 3-920993-95-0, 1997

Das Buch bespricht die wichtigsten kommerziellen objektorientierten und objektrelationalen Datenbanksysteme. Beginnend mit zwei Kapiteln, die Entstehung und Eigenschaften objektorientierter Systeme, sowie Fragen der Datenmodellierung berühren, liegt ein erster Schwerpunkt in Erläuterung und Vergleich verfügbarer Abfrage- und Programmiersprachen. Als praxisrelevant erweist sich das Werk wegen des direkten Vergleichs kommerzieller Systeme. Ein kurzes Abschlußkapitel berührt die Evaluation eines OODBS. Das Buch ist auch für Nicht-Datenverarbeiter lesbar, leidet jedoch ein wenig an der Geschwindigkeit, in der gegenwärtig Neuentwicklungen auf den Markt drängen.

[Moss1998]
Moss, Karl: JAVA Servlets. New York, McGraw-Hill, ISBN: 0-07-913779-2, 1998

Die Einführung zu warum und wie von Servlets ist erfreulich kurz. Ein separates Kapitel behandelt den JAVA-Webserver ein weiteres JRUN als Alternative für traditionelle Webserver. Im Fortgang des Buches läßt der Autor einfache Beispiele rasch hinter sich und behandelt das Zusammenspiel von Servlets untereinander, mit HTML, SSI, RMI, HTTP, HTTP Tunneling und natürlich JDBC. Sehr empfehlenswert.

[Räpple1998]
Räpple, Martin: Sicherheitskonzepte für das Internet, Grundlagen, Technologien und Lösungskonzepte für die kommerzielle Nutzung. Heidelberg, dpunkt.Verlag, ISBN: 3-932588-14-2, 1998

Das Buch beschreibt die verschiedenen Ebenen (entsprechen Schichtenmodell) auf denen "Angriffe" auf Rechnersysteme ausgeführt werden können. Ein zweiter Abschnitt beschäftigt sich mit Gegenmaßnahmen, Sicherheits- und Kryptographiesystemen. Anwendungsszenarien für das kommerzielle Umfeld und Anhänge runden das Buch ab. Obwohl das Buch technische Themen, z.T. detailliert, behandelt, liest es sich streckenweise wie ein Krimi. Durch die klare Gliederung findet das Werk auch als Nachschlagequelle sinnvolle Verwendung.

5.3 Anmerkungen

[1] http://www.omg.org

[2] Die ODMG ist ein Zusammenschluß von kommerziellen Softwareherstellern. Die Mitglieder werden in http://www.odmg.org aufgelistet. Die Adresse ist ein guter Einstiegspunkt in die Dokumentation der Standardisierungsbemühungen, gleichzeitig eine Orientierung, wer von den wichtigsten Herstellern sich diesen Standards verpflichtet fühlt.

[3] The following table lists products that presently support the ODMG Java Binding.

COMPANY	PRODUCT
Sun Microsystems	Java Blend
Computer Associates	Jasmine
Object Design	ObjectStore, PSE for Java
Versant	Versant ODBMS
POET	POET Object Server
Objectivity	Objectivity/DB
Ardent Software	O2 System
Objectmatter	BSF

Quelle: http://www.odmg.org, The Industry Standard for Java Object Storage,By Douglas Barry and David Jordan, Component Strategies - September 1998

[4] Information aus: Anmerkungen zu "CGI leicht gemacht"
http://www.jmarshall.com/easy/cgi/german/cgi_anmerkungen.html#getvspost

[5] Informationsseiten für Entwickler von Netscape:
http:// developer.netscape.com/viewsource/kuslich_javascript.html

[6] Eine sehr beliebte Kombination ist Linux-Apache-MSQL-PHP, welches deswegen das Kürzel LAMP erhalten hat. Diese Kombination ist z.B. beim Virtus-Projekt im Einsatz.

[7] Eine ganze Reihe von Webservern unterstützt von Haus aus die Ausführung von Servlets: Suns Java Web Server, W3C Jigsaw (freeware) WebSite Professional, Lotus Domino Go Webserver... Eine detail-

lierte Aufzählung findet sich in Hunter1998, S. 8ff und im Artikel : "Was es sein darf. Servlets als CGI-Alternative: Tools im Überblick in iX Magazin 11/98, Seite 64

[8] Mit Vorsicht zu genießende Demos der Fähigkeiten von ActiveX finden sich in : http://www.bubis.com/glaser/java_activex.htm