in: Tagungsband der Fachtagung CAD'2000, Berlin, März 2000, pp. 51-67

Effektiver Informationsaustausch durch ORDBS-basiertes Web Content Management

Henrik Loeser, Budi Surjanto
Universität Kaiserslautern, FB Informatik, AG DBIS
Postfach 3049, D-67663 Kaiserslautern
E-Mail: {loeser | surjanto}@informatik.uni-kl.de

Zusammenfassung:

Web-Informationssysteme (WIS) sind aus keinem größeren Unternehmen mehr wegzudenken. In einem *Intranet* oder *Extranet* werden sie zum internen Informationsaustausch bzw. zur Kommunikation mehrerer Partnerunternehmen eingesetzt. Jedoch gibt es beim Betrieb eines WIS noch zahlreiche Probleme, wie etwa hohen Personal- oder Ressourcenaufwand, nicht aktualisierte Informationen, veraltete Navigationshilfen und Indexdaten und zunehmende Unübersichtlichkeit. In diesem Beitrag zeigen wir, wie sich durch das Zusammenspiel von objekt-relationalen Datenbanksystemen (ORDBS) und neuen Web-Technologien wie XML (*Extensible Markup Language*) die Probleme lösen bzw. reduzieren und damit ein effektiver, interner Informationsaustausch erreichen lassen. Die dabei angestrebte automatisierte Aktualisierung und Bereitstellung von Web-Dokumenten wird auch als *Web Content Management* (WCM) bezeichnet. Die daraus resultierenden Vorteile deuten darauf hin, dass durch die Kombination von Orts- und Plattformunabhängigkeit sowie automatische Wartung und Verteilung von Information eine Kerntechnologie für die Kommunikation und Kooperation in technischen Informationssystemen oder, ganz allgemein, in verteilten Anwendungen heranreift.

Abstract:

Web Information Systems (WISs) embody a core technology for all larger enterprises. WISs are employed in an intranet or extranet for internal information interchange or communication between associated companies. However, operating and maintaining a WIS means dealing with several problems, e. g., stressing human and computational resources or keeping documents and index data topical. In this paper, we discuss how the emerging object-relational database systems (ORDBS) and XML (Extensible Markup Language) can be used to eliminate resp. reduce these problems as well as to achieve an effective internal information exchange. This approach of automating maintenance as well as provision of Web documents is also denoted as Web Content Management (WCM). The resulting advantages indicate that, by combining location and platform independence together with automatic maintenance and distribution of information, a kernel technology is maturing for the communication and cooperation in technical information systems or, generally, in distributed applications.

1 Einleitung

Seit den ersten Tagen des *World Wide Web* (WWW oder Web) Anfang der 90er Jahre haben immer mehr Internet- und Web-Techniken Einzug in die Unternehmen genommen. Waren es zunächst nur die Unternehmenspräsentation auf einem Web-Server oder die Bereitstellung von E-Mail- und Internet-Zugängen für die Mitarbeiter, so werden heute Web-Informationssysteme (WIS, siehe Abb. 1) für verschiedene Zwekke eingerichtet. Diese WIS sind jedoch nicht immer für die breite Öffentlichkeit sichtbar, sondern viele werden entweder unternehmens-/abteilungsintern (Intranet) oder in einem abgeschlossenen Verbund mehrerer Unternehmen (Extranet) realisiert. Beispiele sind die interne Bereitstellung von CAD-Dokumenten, von aktuellen Mitteilungen zur Projektkoordination usw. oder der Web-basierte Informationsaustausch zwischen Handelspartnern (Automobilhersteller und Zulieferer, Kleidungshersteller und Einkäufer). In Verbindung mit der Eingabemöglichkeit und der Suche nach Daten können interne WIS auch zur Projektkoordination bzw. zum Wissens- und Informationsaustausch eingesetzt werden. Hierbei muss aber eine Vielzahl von wichtigen Problemen (siehe Abschnitt 2) gelöst werden, um Benutzbarkeit und bleibende Akzeptanz zu gewährleisten.

Nach ihrer Einführung liefern interne WIS aktuelle (und brauchbare?) Informationen und werden daher von den Benutzern auch angenommen. Im Laufe ihres Lebenszyklus entstehen jedoch in der Regel Probleme in so konträren Aspekten wie wachsendes Informationsangebot oder mangelnde Aktualität. Ersteres veringert in der Verbindung mit schlechter Strukturierung die Anwendbarkeit und damit das Auffinden der vom Benutzer gewünschten oder erwarteten Information, was mittelfristig eine sinkende Akzeptanz bewirkt. Aber auch die mangelnde Wartung und Aktualisierung der angebotenen Informationen führt zu abnehmender Akzeptanz und reduziert den praktischen Nutzen eines WIS.

Zwar werden bei allen größeren WIS Datenbanksysteme (DBS) zur Verwaltung und Speicherung der angebotenen Information sowie zur Realisierung von in das WIS integrierten Web-basierten Anwendungen eingesetzt (Applikations-Server oder CGI-Programme, siehe Abb. 1), jedoch lassen sich dadurch die Probleme nur mildern. Oft werden sie nur durch andere substituiert. Mit dem Aufkommen von ORDBS [5, 13, 22], die mittlerweile zum einen stabil und anwendungstauglich, zum anderen durch SQL:1999 (*Structured Query Language*, [19]) auch weitgehend standardisiert wurden, ist es nun möglich, die beim Betrieb von WIS auftretenden Probleme weiter zu reduzieren. Positive Auswirkungen hat zudem der neue Standard XML [4], der eine einheitliche Metasprache für Dokumentbeschreibungssprachen, wie etwa HTML (*Hypertext Markup Language*), definiert.

In diesem Beitrag wollen wir unsere Erfahrungen im Rahmen der Entwicklung eines technischen Informationssystems RITA (Rechnergestütztes Informationssystem für Technische Anwendungen, [10]) aufzeigen, das sich als ein WIS charakterisieren lässt. Die Vorteile der Web-Nutzung, also insbesondere Orts- und Plattformunabhängigkeit der Anwendungen, wurden bereits in [11] beschrieben. Sie werden verstärkt

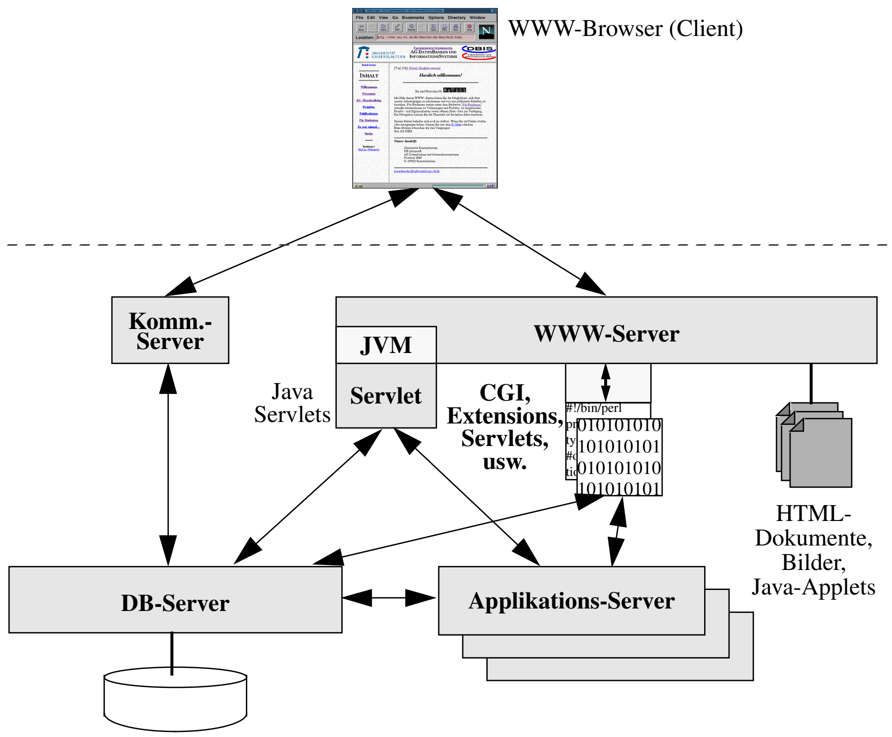

Abbildung 1: Aufbau eines Web-Informationssystems (vereinfacht)

durch den Einsatz von WCM, wodurch eine automatische Aktualisierung und Bereitstellung von Web-Dokumenten angestrebt wird. Unsere Lösung dazu verkörpert das auf einem ORDBS basierende System *iWebDB* (integrierte Web-Datenbank, [15, 16]), dessen Konzepte wir hier vorstellen und diskutieren wollen. Dazu motivieren wir im folgenden zunächst anhand von Beispielen typische bei der Administration und dem Betrieb von WIS auftretende Probleme. In Kapitel 3 geben wir einen kurzen Überblick über XML und damit zusammenhängende Techniken und Technologien. In Kapitel 4 stellen wir überblicksartig die wichtigsten Eigenschaften von ORDBS und ihre Erweiterbarkeit vor. Im Anschluss daran, in Kapitel 5, diskutieren wir auf dem Einsatz von ORDBS und XML beruhende Lösungen für die Verbesserung von WIS und dem internen Informationsaustausch. Unser Prototyp iWebDB und Details seiner Implementierung werden in Kapitel 6 besprochen. In Kapitel 7 diskutieren wir verwandte Arbeiten und schließen danach den Beitrag mit einer Zusammenfassung und einem Ausblick auf zukünftige Arbeiten ab.

2 Probleme

Wie oben schon angedeutet, können beim Betrieb eines WIS zahlreiche Probleme auftreten, deren Auswirkungen mit bisherigen “Standardlösungen” lediglich eingeschränkt werden können. Im folgenden wollen wir auf die bisherigen Lösungen sowie deren Nachteile eingehen. Dabei werden die allgemein vorhandenen technischen Probleme bei der Einrichtung und während des laufenden Betriebes eines WIS außer Betracht gelassen.

2.1 Bereitstellung aktueller Information

Soll ein WIS seinem eigentlichen, im Namen enthaltenen Zweck, also der Versorgung mit bzw. dem Austausch von Informationen dienen, so bedeutet dies, dass vor allem die Aktualität der Informationen gewährleistet sein muß. Ein häufiges Problem nach einer Erstinstallation und Inbetriebnahme eines WIS ist aber gerade die stetig nachlassende Aktualität des Angebots.

Um den für die Aktualitätserhaltung notwendigen Personaleinsatz zu begrenzen, werden schon seit längerem DBS zur Speicherung der Basisdaten, aus denen dann die Web-Dokumente erstellt werden, eingesetzt. Zur Generierung von HTML-Seiten auf Basis der in den Datenbanken (DB) gehaltenen Daten lassen sich bislang zwei verschiedene Techniken unterscheiden, die jedoch beide jeweils spezifische Nachteile aufweisen: Dynamische Seitengenerierung und periodische oder manuell angestoßene Seitengenerierung. Beim ersten Ansatz werden mit Hilfe von CGI-Programmen (*Common Gateway Interface*) oder einer Web-Server-Erweiterung (NSAPI, ISAPI, Apache API, *Servlet* usw.) die vom Benutzer angeforderten Dokumente "on-the-fly", also dynamisch, erzeugt und direkt vom Web-Server an den Benutzer weitergereicht [14]. Das eigentliche Dokument existiert nicht auf dem Server. Bei der zweiten Technik werden die Dokumente von Zeit zu Zeit mittels eines entsprechenden Werkzeuges generiert und im Dateisystem des Web-Servers abgelegt, wo sie dann für die Benutzer zum Abruf bereitstehen.

Ein Vorteil der dynamisch generierten Dokumente ist ihre Aktualität[1]. Jedoch werden für ein skalierbares und leistungsfähiges System beim Einsatz einer solchen Lösung zahlreiche Software-Komponenten auf dem Web-Server benötigt, die wiederum, je nach verwendeter Technik, entsprechende Systemressourcen binden. Gegenüber solchen Lösungen kommt der Ansatz der periodisch durchgeführten Seitenaktualisierung mit deutlich weniger Ressourcen aus. Jedoch besteht ein entscheidender Nachteil darin, dass die von den Benutzern zugreifbaren Web-Dokumente nicht unbedingt dem letzten Stand entsprechen. Wird das Aktualisierungsintervall verkürzt, so ist die Seitengenerierung unter Umständen grundlos. Bei einer Verlängerung des Zeitraumes lässt sich dagegen die Aktualität wiederum nicht immer gewährleisten.

2.2 Suche im aktuellen Dokumentenbestand

Um das schnelle Auffinden von gewünschten Dokumenten in einem großen Datenbestand zu ermöglichen, wird bei vielen WIS ein thematisch gegliederter Einstieg in die Web-Seiten oder eine Suchmöglichkeit angeboten. Zum Auffinden der Dokumente oder Information ist es aber notwendig, dass ein entsprechender Eintrag im Suchindex

1.) Man geht in diesem Fall von aktuellen Daten in der DB aus.

oder in den Übersichtsseiten existiert. Da jedoch in der Regel die Indexerstellung einer Suchmaschine relativ lange dauert, findet ein solche Aktualisierung nur in größeren zeitlichen Abständen statt. Zudem sind dynamisch generierte Seiten oft nicht im Index enthalten und die Daten zu periodisch generierten Seiten veraltet. Ein weiteres Problem betrifft Seiten in geschützen Bereichen des WIS, die nur von autorisierten Benutzern zugegriffen werden dürfen. Entweder sind, für alle sichtbar, Einträge im Suchindex enthalten, oder aufgrund fehlender Einträge findet auch ein Berechtigter nicht die gewünschte Information.

2.3 Datenaustausch und -mehrfachnutzung

In einem WIS bereitgestellte Dokumente und Daten liegen in der Regel im HTML-Format vor, damit sie direkt vom *Browser* angezeigt werden können. Sollen die in den Dokumenten enthaltenen Informationen von den Benutzern auch anderweitig eingesetzt werden, so stehen die Dokumente meist noch in anderen Formaten, wie z. B. als Textverarbeitungsdatei, zur Verfügung. Dies bedingt jedoch zusätzlichen Speicherbedarf sowie erhöhten Änderungsaufwand. Auch können nicht unbedingt alle Anwender die bereitgestellten Formate verarbeiten. Stellt man andererseits diese zusätzlichen Dokumentrepräsentationen nicht zur Verfügung, so können sowohl für die Manipulation als auch Darstellung der Dokumente Nachbearbeitungen bzw. Umformatierungen notwendig sein.

2.4 Benachrichtigung von Benutzern

Werden die auf einem WIS angebotenen Informationen regelmäßig aktualisiert, so besteht ein häufiges Problem darin, diese neuen Informationen den Nutzern auch bekannt zu machen. Oft müssen die Anwender selbst herausfinden, welche Dokumente neu sind oder aktualisiert wurden und was davon nun für sie von Bedeutung ist. Abhilfe ist dadurch möglich, entweder eine Liste der aktualisierten Web-Seiten anzubieten oder über persistente *Cookies* [20] sogar personalisierte Listen zu generieren. Eine weitere Technik sind die sogenannten *Active Channels* [18], die eine Art Push-Technik nach Bestellung eines Angebotes oder Themas realisieren.

3 XML

XML ist eine bedeutende Weiterentwicklung der Dokument(typ)beschreibungssprache SGML (*Standard Generalized Markup Language*), deren Akzeptanz aufgrund der hohen Komplexität stark eingeschränkt war. Entwicklungsziel war daher eine einfach aufgebaute und maschinell leicht zu verarbeitende Sprache. Ein XML-Parser sollte so klein sein, dass er ohne Probleme in Anwendungen, wie z. B. einen Web-Browser, integriert werden kann.

Im folgenden werden wir zunächst XML, SGML und HTML bzw. XHTML (*Extensible HTML*) miteinander vergleichen und dann einen Überblick über die wichtigsten, mit XML zusammenhängenden Techniken geben.

3.1 XML, SGML und (X)HTML im Vergleich

XML ist eine Metasprache zur Beschreibung von Dokumentbeschreibungssprachen, wie z. B. HTML, und ist somit einer übergeordneten Schicht über ihnen zuzuordnen. Während HTML aus einer festen standardisierten Menge von Dokumentbeschreibungselementen (*tags*) mit vordefinierter Bedeutung besteht, sind solche Tags in XML beliebig definierbar. Im Vergleich zu SGML wurde XML zwar mit weniger Funktionalität, dafür aber mit wesentlich einfacherer Struktur entworfen, um so effizient zu verarbeitende Dokumente zu erhalten. Als eine Anwendung von XML entstand XHTML [21], das eine XML-konforme Variante von HTML darstellt und erweiterbar ist.

3.2 Struktur und Inhalt

XML behandelt die Struktur und den Inhalt von Dokumenten separat. Eine DTD (*Document Type Definition*) stellt eine formale Menge von syntaktischen Regeln zur Verfügung, die eine Dokumentstruktur definiert. Sie legt fest, welche Tags zum Aufbau eines Dokumentes verfügbar sind und wie sie angewendet werden. Durch die Trennung zwischen Dokumentstruktur und -inhalt sowie durch die Modellierungsmächtigkeit von DTD erzielt XML einen hohen Grad an Flexibilität und Erweiterbarkeit.

3.3 Präsentation

In XML-Dokumenten haben Tags keinerlei Bedeutung bzgl. der Präsentation. Um ein XML-Dokument auf die gewünschte Weise präsentieren zu können, wurde XSL (*Extensible Stylesheet Language*, [7]) entwickelt. Die Grundidee ist es, die im Dokument enthaltenen Tags entsprechend einer geeigneten Formatvorlage (*stylesheet*) zu ersetzen, z. B. das Element "<ueberschrift>" bei einer Umwandlung für das Web durch "<h1>". Man kann Formatvorlagen für verschiedene Ausgabemedien, wie etwa das Web, Textverarbeitungssysteme, mobiles Telefon usw., entwickeln, sie dann auf ein XML-Dokument anwenden und dadurch eine Präsentation von demselben XML-Dokument auf verschiedenen Medien erhalten. Dadurch wird eine medienunabhägige Präsentation von XML-Dokumenten ermöglicht.

3.4 Hypertext-Verweiskonstrukt

Ein Verweiskonstrukt in Hypertext-Dokumenten, wie etwa in HTML-basierten Web-Seiten, ist unentbehrlich. Jedoch findet man in HTML nur ein einfaches Verweiskonstrukt, das nur einseitige Referenzierungen erlaubt. Im Vergleich dazu stellt XLink (*XML Linking Language*, [8]) weitaus mächtigere Verweiskonstrukte zur Verfügung, wie erweiterte Verweise (*extended link*s) und Verweisgruppen (*extended link groups*). Mit Extended Links sind bidirektionale Referenzen sowie multiple Referenzen, d. h. Verweise auf mehrere Dokumente, möglich. Eine Extended Link Group fasst Extended Links zusammen, wodurch mehrere Dokumente logisch verbunden werden, so dass sie von Anwendungen, z. B. einem Browser, entsprechend als Gruppe verarbeitet werden. Weiterhin lassen sich mit Hilfe von XPointers (*XML Pointer Language*, [6]) Teile oder einzelne Elemente eines XML-Dokuments, sogar einzelne Buchstaben, referenzieren, ohne dass diese einen expliziten Bezeichner besitzen müssen.

3.5 Metadaten

Metadaten sind “Daten über Daten”. Im Zusammenhang mit dem Web beschreiben Metadaten den Inhalt von Web-Dokumenten. Sie lassen sich dadurch für verschiedene Anwendungsgebiete einsetzen, beispielsweise bei Suchmaschinen zur Verbesserung der Suchfähigkeit oder bei intellegenten Agenten zum Auffinden, Austausch sowie Bereitstellen von Informationen. Als Ergänzung zu XML wurde RDF (*Resource Description Framework*, [17]) zur Beschreibung von Metadaten spezifiziert und soll so das Auffinden von Informationsquellen sowie den Datenaustausch erleichtern. Das von RDF verwendete Datenmodell ist vergleichbar mit dem Frame-Modell aus dem Gebiet Wissensrepräsentation.

3.6 XML Schema

Die Anforderung an ein ausdrucksstärkeres Konstrukt als DTD zur Definition einer Dokumentstruktur hat zur Entwicklung von XML Schema [2, 3] geführt. Analog zu DTD spezifiziert man mit XML Schema Dokumentstrukturen, wobei auf Dokumentenbestandteilen Bedingungen (*constraints*) bzgl. der Bedeutung, der Verwendung und der Beziehungen zwischen Dokumentbestandteilen definiert werden können. Auch die Spezifikation von impliziten Informationen wie z. B. Default-Werten ist möglich. Dies soll insbesondere den Austausch von Datenbankinhalten erleichtern, um so neben den eigentlichen Daten auch die Schemainformationen möglichst verlustfrei übertragen zu können.

4 Objekt-Relationale Datenbanksysteme

ORDBS, die seit wenigen Jahren kommerziell angeboten werden und mittlerweile durch SQL:1999 [19] zum Industriestandard geworden sind, bieten als Ergänzung zu rein relationalen DBS (RDBS) neben ihren objektorientierten Konzepten auch Möglichkeiten zur Erweiterung des DBS (*extensibility*). Im folgenden wollen wir zunächst kurz die objektorientierten Konzepte von ORDBS vorstellen, anschließend dann auf Aspekte der Erweiterbarkeit eingehen.

4.1 Objektorientierung

Als Ergänzung zu den von RDBS her bekannten Konzepten wie Tabellen und einfachen Datentypen (Integer, Zeichenkette, Datum usw.) bieten ORDBS objektorientierte Erweiterungen. Zum einen ist es nun möglich, eigene Typen zu definieren und als Basis für eine Tabellendefinition oder als Attributtyp zu verwenden, zum anderen existieren nun auch Mengenkonstruktoren zur Definition von Kollektionen. Mittels (einfacher) Vererbung können Typhierarchien aufgebaut werden. Anders als bei objektorientierten DBS (OODBS), bei denen direkt eine Instanz eines Typs erzeugt und gespeichert werden kann, sind Instanzen definierter Typen als Tupel zugehöriger Tabellen (*typed tables*) zu verwalten. Tabellen können dabei wie die Typen in einer Vererbungshierarchie aufgebaut werden (Tabellenhierarchie).

Mittels benutzerdefinierter Funktionen lassen sich Methoden (Verhalten) zu einem Typ definieren. Diese Methoden können in SQL-Anweisungen verwendet werden. Durch den in SQL:1999 aufgenommenen REF-Typ ist es nun möglich, Objektreferenzen zu realisieren und innerhalb einer SQL-Anfrage Zeigerketten zu traversieren.

4.2 Erweiterbarkeit

Neben ihren zusätzlichen objektorientierten Konzepten zeichnen sich ORDBS insbesondere durch ihre Erweiterungsfähigkeit aus. Hierunter versteht man die Möglichkeit, das ORDBS um Datentypen (*user defined types*, UDTs), in einer 3GL (*third generation language*) programmierte Funktionen (*user defined functions*, UDFs) sowie neue Indexstrukturen so zu ergänzen, dass diese neuen Typen, Funktionen und Indexstrukturen vom Benutzer bzgl. der Sprachintegration und der Optimiererunterstützung wie die eingebauten, d. h. die bereits vom (nicht-erweiterten) DBS bereitgestellten, verwendet werden können. Als Ergänzung dieser Konzepte können über sogenannte *Table Functions* (IBM) bzw. über das *Virtual Table Interface* (Informix) externe Datenquellen in Form von Tabellen zur Verfügung gestellt werden, die dann mit Hilfe von SQL manipuliert werden können. Fasst man die benutzerdefinierten Erweiterungen aus Gründen der Administrierbarkeit anwendungs(bereichs)bezogen in Modulen zusammen, so spricht man je nach DBS-Hersteller von *DataBlade* (Informix), *Extender* (IBM) oder *Cartridge* (Oracle).

Im folgenden gehen wir kurz auf die für das Thema dieses Beitrags besonders relevanten Aspekte der Erweiterbarkeit von ORDBS ein.

4.3 Benutzerdefinierte Datentypen

Mit Hilfe des Konzeptes der benutzerdefinierten Datentypen ist es Anwendern möglich, eigene Datentypen dem DBS bekannt zu machen. Dies kann von der einfachen Umbenennung vorhandener Typen (*distinct types*) zur Einführung von höherer Semantik und strengerer Typisierung über die Definition komplexer Datentypen (*structured types*) bis hin zum Einbringen von Typen mit für das DBS unbekannter, interner Struktur (*opaque types* oder *black box types*) gehen. Damit das DBS auch mit letzteren arbeiten kann, ist die Definition von sog. S*upport Functions* notwendig, die u. a. die Konvertierung zwischen internem und externem Format übernehmen. Basierend auf den eingebauten Typen und den darauf erzeugten benutzerdefinierten Typen können wiederum neue, strukturierte Typen definiert werden.

4.4 Benutzerdefinierte Funktionen

Anders als bei bisherigen *Stored Procedures* können nun Funktionen in einer 3GL wie C, C++ oder Java entwickelt werden, wobei je nach Hersteller verschiedene objektorientierte Konzepte unterstützt werden. Das heisst, die Bindung einer Funktion bzw. Methode an einen Typ, das Überladen von Funktionen sowie spätes Binden sind möglich. Solche Funktionen können im Rahmen von SQL-Anweisungen aufgerufen werden.

4.5 Benutzerdefinierte Indexstrukturen

Neben der Definition eigener Datentypen und Funktionen bieten ORDBS auch die Möglichkeit zur Integration eigener Indexstrukturen in den DB-Server, um so für neue Datentypen speziell angepasste Indexstrukturen zur Verfügung zu haben. Hierzu ist es erforderlich, mittels UDFs die vom DBS für eine Zusammenarbeit benötigten Schnittstellenfunktionen bereitzustellen. Über diese kommuniziert das DBS mit der ihm ansonsten unbekannten Indexstruktur und regelt die Einbindung in die internen Verarbeitungsabläufe.

4.6 Zugriff auf externe Daten

Neben der Erweiterung des DB-Servers bieten ORDBS zudem die Möglichkeit der Einbindung extern, d. h. ausserhalb des DBS gespeicherter Daten in die SQL-Verarbeitung. Je nach Hersteller spricht man in diesem Zusammenhang von Tabellenfunktionen (IBM DB2 UDB) oder virtuellen Tabellen (Informix). Die Grundidee ist hierbei die Simulation des relationalen Scan-Operators, so dass die Daten zeilenweise angefordert werden können. Anders als bei IBM ist bei Informix eine Aktualisierung der Daten möglich (Einfügen, Ändern, Löschen), so dass sich mit Hilfe einer SQL-Anweisung auch extern in einer Datei oder auf einem anderen Server gehaltene Daten verändern lassen.

5 ORDBS-basierte Lösungen

Nachdem wir die wesentlichen Merkmale von ORDBS sowie von XML vorgestellt haben, wollen wir im folgenden, ausgehend von der Diskussion WIS-spezifischer Probleme (Kapitel 2), auf dem Einsatz von ORDBS und XML basierende Lösungen erörtern. Dabei setzen wir die mittlerweile übliche Verwaltung von Web-Dokumenten mit Hilfe von DBS voraus.

5.1 Bereitstellen aktueller Information

Durch die Ausnutzung der Möglichkeiten (O)RDBS ist es nun möglich, die bisherigen Probleme zu überwinden und eine Lösung anzubieten, die sowohl Web-Seiten mit aktueller Information bereitstellt als auch wenig Ressourcen benötigt. Hierzu ist es zum einen notwendig, über Datenänderungen informiert zu werden, zum anderen müssen anschließend alle betroffenen Dokumente aktualisiert werden. Während eine DB-interne Notifikation bei Änderungsoperationen in Form von Triggern schon länger in vielen RDBS vorhanden ist und Trigger in SQL:1999 nun genormt wurden, bedarf es für die Aktualisierung der HTML-Seiten eigener Lösungen. Hilfreich erweist sich hier die oben vorgestellte Erweiterbarkeit von ORDBS, insbesondere die Möglichkeit der Einbindung externer Daten sowie die ebenfalls in SQL:1999 in Teilen standardisierte Ergänzung um benutzerdefinierte Funktionen. Basierend auf Triggern und UDFs kann nun bei einer Änderung von Produktionsdaten wie im folgenden vereinfacht dargestellt verfahren werden:

- Die Änderungsoperation löst einen Trigger aus.
- Der Trigger ruft eine UDF zur Überprüfung der Dokumentabhängigkeiten auf, d. h., es wird eine Liste aller Dokumentvorlagen, die von der Datenänderung betroffen sind, ermittelt.
- Für jede Vorlage wird die entsprechende Konfiguration geladen, der Dokumentengenerator mit der entsprechenden Konfiguration aufgerufen und das erzeugte Dokument unter seinem vorgesehenen Namen in das Dateisystem des Web-Servers geschrieben.

Dieser Ablauf setzt voraus, dass beim Einbringen von Dokumentvorlagen die Abhängigkeiten der Vorlage zu DB-Tabellen spezifiziert werden, wozu eine Abhängigkeitenverwaltung notwendig ist. Dokumentvorlagen sind dabei Dokumente, in die mit Hilfe von z. B. Makroanweisungen Ergebnisse von Datenbankabfragen eingebettet werden [14]. Zwar ist eine automatische Abhängigkeitenerkennung prinzipiell durch Analyse der Vorlagen möglich, jedoch werden dann alle Abhängigkeiten erfaßt, was in der Praxis eher hinderlich ist. So ist die automatische Aktualisierung einer Web-Seite mit aktuellen (betriebsinternen) Mitteilungen nur bei neuen Nachrichten notwendig. Andere Änderungen, z. B. an einer Tabelle mit internen Web-Diensten, die als Querverweise mittels einer DB-Anfrage ebenfalls in die Mitteilungsseite eingebunden sind, können dagegen im Rahmen einer Neugenerierung bei neuen Mitteilungen nachgezogen werden.

Durch das vorgestellte Konzept der triggerbasierten Dokumentenaktualisierung kann die Aktualität der Dokumente gewährleistet werden, ohne dass diese dynamisch zum Zeitpunkt ihrer Anforderung generiert werden müssen. Wegen Benutzerinteraktion und potentieller zeitabhängiger Inhalte eignet sich dieses Verfahren zwar nicht für die Realisierung von Web-Applikationen, jedoch werden für diese personalisierte und damit nur von einem einzigen Benutzer zugegriffene Web-Seiten benötigt. Die von unserem Ansatz intendierte Ressourcen-Einsparung bei gleichzeitiger Beibehaltung der Aktualität ließe sich somit ohnehin nicht erzielen. Eine kleine Optimierung läßt sich aber über die Generierung von Dokumentkomponenten, wie z. B. Seitenköpfe oder Navigationsblöcke erreichen, so dass die dynamische Seitengenerierung bei den angebotenen Web-Applikationen aufgrund zum Zugriffszeitpunkt eingesparter Operationen verkürzt werden kann.

5.2 Verbesserte Suchmöglichkeiten

Werden bei traditionell betriebenen WIS Übersichtsseiten entweder sporadisch manuell aktualisiert oder beim Zugriff dynamisch generiert, so läßt sich Aktualität nun durch das vorgestellte Konzept der triggergesteuerten Generierung erreichen. Jedoch wird damit noch nicht die Aktualitätsproblematik lokaler Suchmaschinen entschärft.

Dies ist nun in der Verbindung von DB-Triggern mit der Erweiterbarkeit von ORDBS, insbesondere UDFs, eigenen Indexstrukturen und dem Zugriff auf externe Daten, möglich.

Wurde bislang der Suchindex nur von Zeit zu Zeit, dafür aber für alle Dokumente aktualisiert, so kann über die Nutzung von Triggern genau die Umkehrung erreicht werden: Nach einer Modifikation werden lediglich die Indexdaten der von einer Änderung betroffenen Dokumente angepasst. Dabei ist es egal, ob das DBS zur Dokumentspeicherung vor dem Publizieren auf dem Web-Server oder sogar zur direkten Dokumentenlieferung via Web-Server genutzt wird. Im ersteren Fall muss das Herausschreiben der Web-Seiten in das Dateisystem des Web-Servers protokolliert werden, im letzteren Fall sind Modifikationen an den Dokumenten in der DB zu protokollieren.

Durch die Integration eines entsprechenden, z. T. sogar schon als kommerzielle DB-Erweiterung erhältlichen, Suchindex als eigene Indexstruktur in das ORDBS und die Nutzung von SQL als Anfragesprache[2] kann in Verbindung mit einer Benutzeridentifikation oder geeigneten Zugriffsautorisierung eine Personalisierung der Suche erfolgen. Dies bedeutet, beispielsweise dass ein Mitarbeiter aus der Entwurfsabteilung nur Dokumente aus seiner Abteilung sowie allgemein verfügbare Informationen erhält (siehe Anfrage in Abb. 2). Anders als bei den traditionellen Suchmaschinen kann somit auch eine Integration von geschützen Dokumenten in den Suchindex unter Beibehaltung des Zugriffschutzes erfolgen. Die sonst z. T. nicht durchgängige Informationssicherheit durch die Aufnahme von geschützen Dokumenten in einen Suchindex und die Anzeige von Dokumentverweisen für auch nicht autorisierte Personen können somit vermieden werden.

```
SELECT d.path, d.filename
FROM internalDocs d, group g,
     user u, inGroup ig
WHERE u.id=4711 AND ig.uid=u.id
AND g.id=ig.gid
AND (d.owner=u.id OR d.gid=g.id)
AND contains(d.page,'%Treffen%');
```

Abbildung 2: Sichere Suche

Als Ergänzung zur Indexierung eines Dokumentes können zur gezielten Dokumentenanalyse UDFs realisiert werden, um so z. B. in einem Web-Dokument nach allen Überschriften oder Auflistungen suchen zu können. In Verbindung mit einem (Volltext-) Index ist so eine Dokumentsuche mit einem bestimmten Inhalt und zusätzlichen strukturellen Bedingungen möglich. Beispielsweise sucht die in der Abb. 3 angeführte Anfrage alle Dokumente, in denen das Wort “Volkswagen” enthalten ist und die nach einer Überschrift “Zulieferer” einen als “ZF Friedrichshafen AG” betitelten Hyperlink beinhalten. Zusammen mit einem Web-Formular, das einfache Bedienbarkeit bietet,

2.) Nach wie vor wird natürlich ein HTML-Formular, in dem wie üblich die Suchworte eingegeben werden, als grafische Benutzerschnittstelle verwendet. Intern erfolgt eine Abbildung auf SQL.

ist es auf diese Weise allen autorisierten Personen möglich, schnell und gezielt in der verwalteten Datenbasis zu suchen.

5.3 Konsistenz von Hyperlinks

In den Web-Dokumenten enthaltene Hyperlinks dienen zwar der einfachen Verzweigung und Navigation innerhalb eines Dokumentes oder zwischen vielen Dokumenten, jedoch führt gerade die Dynamik des WWW zu einer Vielzahl von "toten" Links, auch innerhalb einer einzigen Web-Site. Mit Hilfe der schon angesprochenen Analysefunktionen zur Extraktion von Link-Informationen sowie aufbauend auf einer internen Link-Verwaltung, in der die Verweise zwischen Dokumenten erfasst sind, ist es nun möglich, die Problematik deutlich zu begrenzen und z. B. Verweise auf nicht-existente Dokumente aufzuspüren und gegebenenfalls geeignet zu reagieren. Dies kann durch das Informieren der zuständigen Person oder durch Ablehnung der auslösenden Änderungsoperation geschehen.

```
SELECT path, filename
FROM internalDocs
WHERE contains(page,'%Volkswagen%')
AND EXISTS (
   SELECT h.pos,l.pos
   FROM headers(page) h, hrefs(page) l
   WHERE h.pos<l.pos
   AND h.content LIKE '%Zulieferer%'
   AND l.content LIKE
   '%ZF Friedrichshafen AG%'
);
```

Abbildung 3: Kombinierte Anfrage

Da die Konsistenzwahrung von (lokalen) Hyperlinks, insbesondere in Verbindung mit Dokumentenversionierung und dem sogenannten *Web Site Restructuring*, dem Umbau großer Teile einer Web-Site, eine sehr komplexe Thematik ist, können wir aus Platzgründen an dieser Stelle nicht weiter darauf eingehen.

5.4 Datenaustausch

Viele der auf einer Web-Site bereitgestellten Dokumente wurden entweder mit einem speziellen, auf HTML ausgelegten Editor erzeugt oder aus einer Textverarbeitung heraus nach HTML konvertiert. Benutzer, die auf ein entsprechendes Web-Dokument zugreifen, haben jedoch große Probleme, dieses ohne manuelle Nacharbeit und Verlust von Formatierungen in einer Anwendung, z. B. einer Textverarbeitung, weiter zu verwenden.

Abhilfe kann hier die interne Speicherung aller Dokumente in einem einheitlichen, auf XML aufbauenden Format schaffen, d. h., alle Dokumente verwenden eine einheitliche DTD. Um nun ein Dokument in eine HTML-Seite zu transformieren, muss eine entsprechende Vorlage definiert sein. Ein als UDF in den DB-Server integrierter

XSL-Prozessor kann nun beim Publizieren des XML-Dokumentes, d. h. beim Ausschreiben der Datei auf den Web-Server, anhand der in der Vorlage definierten Regeln das XML-Dokument in eine HTML-Seite umwandeln. Gleichwohl ist es aber auch möglich, weitere Abbildungen zu definieren und andere Dokumentformate, beispielsweise *PostScript*, PDF oder RTF, zu erzeugen, wie es auch schon von einigen Web-Werkzeugen angeboten wird [1]. Auf diese Weise kann den Benutzern eine Wahlmöglichkeit bezüglich des Dokumentformates und somit eine schnelle und einfache Übernahme der bereitgestellten Informationen in eigene Anwendungen ermöglicht werden.

Klassische relationale Daten, die bisher z. B. als HTML-Tabellen dargestellt wurden, können in auf Basis von XML Schema definierten Dokumenten ausgetauscht werden. Dabei lassen sich sowohl die Daten als auch das zugrundeliegende Schema übernehmen.

5.5 Benachrichtigung der Benutzer

Werden auf einer Web-Site Informationen zu einer Vielfalt verschiedener Themen angeboten und diese jeweils unregelmäßig aktualisiert, so fällt es den Benutzern schwer, einzelne, für sie evtl. sogar wichtige Änderungen wahrzunehmen. Zudem ist dies ein immer wiederkehrender und zeitaufwendiger Vorgang.

Eine einfache Lösung für dieses Problem lässt sich durch die Integration einer E-Mail-Komponente in Form von UDFs in das ORDBS erzielen, wodurch E-Mails direkt durch den DB-Server verschickt werden können. Dabei ist es möglich, ähnlich wie in den oben beschriebenen Lösungen, den E-Mail-Versand wiederum durch einen Trigger bzw. ein entsprechendes Änderungsereignis auszulösen. Mit der E-Mail-Funktionalität und Triggern läßt sich zunächst aber nur eine Notifikation aller (in einem Modul zur Benutzerverwaltung) eingetragenen Anwender realisieren, der Benutzer hat weiterhin selbst die eingehenden Nachrichten zu filtern. Basierend auf einer thematischen Gliederung der Web-Site bzw. mit Hilfe der Analysefunktionen oder zu spezifizierender Suchausdrücke, z. B. *Regular Search Expressions*, ist die Realisierung einer Besteller-Verwaltung und Notifikation bei thematisch interessanten Änderungen möglich. Dies bedeutet, dass ein Benutzer dem System die ihn interessierenden Themen mitteilt oder sogar entsprechende Suchausdrücke spezifizieren kann. Bei einer für ihn relevanten Änderung wird er nun via E-Mail benachrichtigt und das geänderte (kleine) Web-Dokument gleich eingebunden.

Greift man vor dem E-Mail-Versand auf die Datenaustauschfunktionen zurück, so können die neuen oder veränderten Dokumente den Benutzern sogar in den von ihnen gewünschten Formaten zur Verfügung gestellt werden. Auf diese Weise ist zum einen die Benachrichtigung der Anwender über für sie wichtige Ereignisse möglich, zum anderen können die in der E-Mail enthaltenen Daten gleich in den für einen Benutzer relevanten Applikationen verwendet werden, ohne dass ein aktives Kontaktieren des Web-Servers (und Suchen des Dokumentes) notwendig ist.

Das Filtern von relevanten Aktualisierungen und die Benachrichtigung der Anwender über E-Mail ist nur unter bestimmten Voraussetzungen praktikabel. Zum einen kann es bei zu vielen täglichen Änderungen zu einer wahren “Datenflut” kommen, d. h., ein Benutzer bekommt zu viele Benachrichtungen. Hiermit verbunden ist eine sinkende Akzeptanz und damit eingeschränkte Kenntnisnahme der eintreffenden Nachrichten. Es wird das Gegenteil des Erwünschten erreicht. Zum anderen kann aber auch die Analyse von neuen Dokumenten, insbesondere mit benutzerspezifischen Suchkriterien, bei einer großen Zahl an Benutzern zu viel Zeit in Anspruch nehmen. Somit erscheint eine zu feingranulare Spezifikation von Zuweisungskriterien nicht sinnvoll.

6 Implementierung

Die im vorangegangenen Abschnitt beschriebenen Konzepte zur Verbesserung des Informationsaustausches auf Basis von objekt-relationaler DB-Technologie wurden am System iWebDB basierend auf dem ORDBS *Informix Dynamic Server 2000* [12] prototypisch umgesetzt. Das System iWebDB besteht momentan aus den als DataBlade realisierten und in das DBS integrierten Komponenten Doc (Dokumentenverwaltung), ED (*External Data*: Dateisystemintegration), *eXtract* (Analysefunktionen und Index) und DG (*Document Generator*). Hinzu kommen das grafische Administrationswerkzeug SM (*Site Manager*) und der universell einsetzbare Makroprozessor MP (*Macro Processor*), auf dem auch der Dokumentengenerator beruht (siehe Abb. 4).

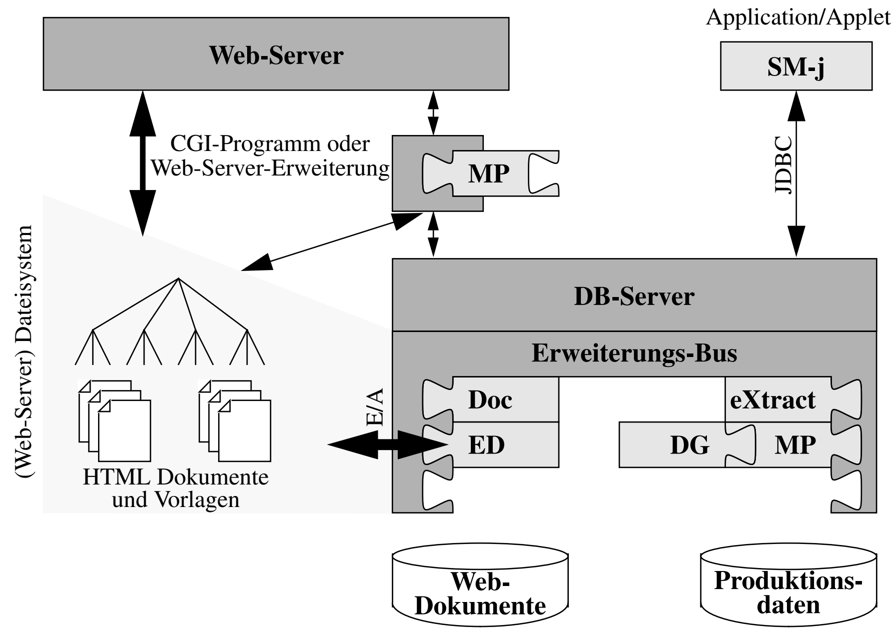

Abbildung 4: Architektur von iWebDB

Neben der Kernkomponente Doc, die alle wesentlichen Datentypen und grundlegenden Verwaltungsfunktionen bereitstellt, nimmt das Modul ED eine besondere Stellung ein. Auf Basis des *Virtual Table Interface* stellt es eine nahtlos in SQL integrierte Einbindung des lokalen Dateisystems zur Verfügung (siehe Abb. 4, “E/A”). Dies bedeutet, dass mittels SQL sowohl Anfragen an als auch Manipulationen am Dateisystem möglich sind. Da für das (externe) Dateisystem und das intern simulierte dieselben Datentypen verwendet werden, können Dateien bequem zwischen internem und externem Dateisystem ausgetauscht werden. Das Administrationswerkzeug nutzt dies, um mit über JDBC (*Java DataBase Connectivity*) ausgeführten SQL-Anweisungen die vom Benutzer gewünschten Dateien zu verschieben. Der in das DBS integrierte Dokumentengenerator DG nutzt ebenfalls die Dateisystemintegration und schreibt mit Hilfe eines INSERT die generierten Dokumente auf den Web-Server.

Zur besseren Unterstützung von Strukturanfragen auf XML-Dokumenten wird z. Zt. auf Basis des *Virtual Index Interface* ein Strukturindex in das DBS integriert. Hierzu wurde in einem ersten Schritt ein SAX-kompatibler (*Simple API to XML*) XML-Parser in Form einer UDF eingebunden, so dass für alle in einem Dokument enthaltenen Markierungen der Name, die Attribute sowie die Textposition der Tags extrahiert werden kann. Die Implementierung der daruf aufbauenden Indexstruktur wird momentan fertiggestellt.

Der ORDBS-basierte E-Mail-Versand wird seit 1997 im Rahmen des internen *Web-Notes*-Systems eingesetzt, um aktuelle Nachrichten an alle Besteller einer bestimmten Nachrichtenart zu verschicken. Dabei dient das DBS als suchbares Archiv, so dass auch neue Benutzer für sie wichtige Nachrichten nachlesen können.

Bei beiden Systemen wurden die benutzerdefinierten Funktionen in der Sprache C entwickelt, da z. T. Java in älteren Versionen des eingesetzten ORDBS noch nicht zur Verfügung stand bzw. auch Geschwindigkeitsgründe gegen Java sprachen. Bedingt durch Kodierungsrestriktionen und nur wenige Debugging-Möglichkeiten gestaltet sich die UDF-Entwicklung in C meist schwierig. Anders verhält es sich mit Java. Hier sind die Entwicklungszeiten und auch die Fehleranfälligkeit relativ gering, so dass Java zunehmend eingesetzt wird.

7 Verwandte Arbeiten

In den vorangegangenen Abschnitten haben wir, z. T. überblicksartig, die beim Betrieb eines WIS typischerweise auftretenden Probleme und ORDBS-basierte Lösungen vorgestellt. Zu vielen der angesprochenen Probleme existieren Einzellösungen, die jedoch wirklich nur das einzelne Problem selbst oder einen begrenzten Bereich abdecken. Eine durchgängige und in einem System integrierte Lösung, wie sie in diesem Beitrag vorgestellt wurde, gibt es bisher nicht.

So konzentrieren sich viele Ansätze und Produkte auf die Optimierung der DB-basierten, dynamischen Web-Seitengenerierung (siehe [14] für einen Überblick). Mit der in diesem Beitrag vorgestellten, DBS-gesteuerten Dokumentengenerierung werden jedoch viele dieser Ansätze überflüssig. Anders verhält es sich bezüglich der Speicherung und Suche von HTML- bzw. XML-Dateien. So wird derzeit in verschiedenen Projekten untersucht, wie die Speicherung und Indexierung von bzw. die Suche in XML-Dokumenten optimal von DBS unterstützt werden kann [9]. Allerdings basieren diese Ansätze auf der Verwendung von klassischen RDBS oder auch OODBS, die von uns eingesetzte Erweiterbarkeit ORDBS wird nicht betrachtet. Hingegen haben einige ORDBS-Hersteller Unterstützung und Produkte für die Verwaltung und Analyse von XML-Dokumenten angekündigt. Dieses sind entweder zusätzliche (externe) Komponenten (IBM, Oracle), die mit dem eigentlichen DBS interagieren, oder es sind noch keine genauen Angaben oder Pläne des Herstellers vorhanden. Desweiteren gibt es spezielle sogenannte XML-Server (Software AG, Object Design, Poet Software, etc.), die besondere Funktionalität und Geschwindigkeit bei der Verwaltung von XML-Dokumenten versprechen.

Insgesamt lässt sich festhalten, dass zwar für einzelne Probleme entsprechende Lösungsansätze existieren, wenn auch sich diese immer nur auf Teilaspekte beschränken. Eine in das DBS integrierte, auf die Gesamtheit der auftretenden Probleme abzielende Lösung, wie die hier vorgestellte, ist bisher einzigartig.

8 Zusammenfassung und Ausblick

In diesem Beitrag haben wir zunächst die typischen, beim Betrieb eines WIS auftretenden Probleme aufgezeigt. Anschliessend haben wir die Grundzüge der Dokumentbeschreibungssprache XML sowie der Erweiterbarkeit von ORDBS vorgestellt, um dann auf diesen Technologien beruhenden Lösungen zu diskutieren. Diese erlauben es, Web-Dokumente ohne die sonst notwendige Abwägung zwischen erhöhtem Ressourcenbedarf oder vermindeter Aktualität bereitzustellen, einen aktuellen, personalisierbaren und damit Informationssicherheit bietenden Suchindex zu betreiben sowie den internen Informationsfluss durch den Einsatz von E-Mails zu beschleunigen. Zudem ist eine Weiterverwendung der verwalteten Informationen, beispielsweise für gedruckte Informationsbroschüren, durch die Nutzung von XML als grundlegendem Dokumentformat möglich.

Setzt man die beschriebenen Technologien und Konzepte zur Realisierung eines WIS für die Verbesserung der internen Kommunikation, bzw. innerhalb eines abgeschirmten Unternehmensverbundes, ein, so lassen sich durch die zum einen aktuelleren Informationen, zum anderen durch deren schnellere Verbreitung sowohl eine bessere interne Kommunikation und Kooperation als auch ein daraus abgeleiteter wirtschaftlicher Erfolg erzielen.

Wenn auch die in diesem Beitrag vorgestellten Lösungen prototypisch weitestgehend validiert wurden, so bedarf es zum einen noch eines entsprechenden Praxistest, zum anderen ist aber auch eine bessere Unterstützung bestimmter ORDBS-Erweiterbarkeitsmechanismen durch mehr Hersteller und auch eine Vereinheitlichung der unterschiedlichen Ansätze durch einen entsprechenden SQL-Standard notwendig. Es bleibt also abzuwarten, wie sich die ORDBS und die auf ihnen beruhenden Lösungen weiterentwickeln.

9 Literatur

1. Bakken, S., Aulbach, A., Schmid, E., Winstead, J., Wilson, L., Lerdorf, R., Suraski, Z.: *PHP Manual*, PHP Documentation Group, http://www.php.net/manual, 1999.

2. Beech, D., Lawrence, S., Maloney, M., Mendelsohn, N., Thompson, H. S.: *XML Schema Part 1: Structures*, W3C Working Draft 6-May-1999, W3C, http://www.w3.org/TR/ xmlschema-1/, Mai 1999.
3. Biron, P. V., Malhotra, A.: *XML Schema Part 2: Data Types*, W3C Working Draft 6-May-1999, W3C, http://www.w3.org/TR/xmlschema-2/, Mai 1999.
4. Bray, T., Paoli, J., Sperberg-McQueen, C. M.: *Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998*, W3C, http://www.w3.org/TR/REC-xml, Februar 1998.
5. Chamberlin, D.: *Using the New DB2: IBM's Object-Relational Database System*, Morgan Kaufman, 1996.
6. De Rose, S., Daniel Jr, R.: *XML Pointer Language (XPointer), W3C Working Draft 9-July-1999*, W3C, http://www.w3.org/TR/WD-xptr, Juli 1999.
7. Deach, S.: *Extensible Stylesheet Language (XSL) Specification*, *W3C Working Draft 21-April-1999*, W3C, http://www.w3.org/TR/ WD-xsl/, April 1999.
8. De Rose, S., Orchard, D., Trafford, B.: *XML Linking Language (XLink), W3C Working Draft 26-July-1999*, W3C, http://www.w3.org/TR/xlink, Juli 1999.
9. Florescu, D., Kossmann, D.: *Storing and Querying XML Data using an RDBMS*, Data Engineering, Vol. 22, No. 3, IEEE Computer Society, September 1999.
10. Härder, T., Thomas, J.: *RITA - ein rechnergestütztes Informationssystem für technische Anwendungen*, in: ITG-Fachbericht 137, Softwaretechnik in Automation und Kommunikation (STAK'96), München, März 1996, S. 111-126.
11. Härder, T., Loeser, H., Zhang, N.: *Supporting Adaptable Technical Information Systems in Heterogeneous Environments - Using WWW and ORDBMS*, in: Proc. 8th Int. Workshop on Database and Expert Systems Applications (DEXA'97), Toulouse, Sept. 1997, pp. 295-303.
12. *Informix Dynamic Server 2000, Getting Started*, Informix Software Inc., http://www.informix.com, 1999.
13. Kim, W.: *Object-Relational - The unification of object and relational database technology*, UniSQL White Paper, 1996.
14. Loeser, H.: *Techniken für Web-basierte Datenbankanwendungen - Anforderungen, Ansätze, Architekturen*, in: Informatik - Forschung und Entwicklung, Springer, 13(4), Dezember 1998.
15. Loeser, H.: *iWebDB - Eine integrierte Web-Datenbank auf Basis objekt-relationaler DB-Technologie*, in: Proc. of BTW'99, Freiburg, März 1999.
16. Loeser, H., Ritter, N.: *iWebDB - Integrated Web Content Management based on Object-Relational Database Technology*, in: Proc. Int. Database Engineering and Applications Symposium (IDEAS'99), Montreal, Canada, August 1999.
17. Lassila, O., Swick, R. R.: *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation 22-February-1999, W3C, http://www.w3.org/TR/ REC-rdf-syntax, Februar 1999.
18. Active Channel Technology Overview, Microsoft Corporation, http://www.microsoft.com, 1999.
19. Mattos, N., Pistor, P., Dessloch, S.: *SQL3, Objektorientierung und Java als Standard: Ein Überblick über SQL3 und SQLJ*, Tutorial bei der BTW'99, Freiburg, März 1999.
20. *Persistent Client State HTTP Cookies*, Netscape Communications Corporation, http://www.netscape.com,1998.
21. Pemberton, S., et al.: *XHTML[tm] 1.0: The Extensible HyperText Markup Language - A Reformulation of HTML 4.0 in XML 1.0*, W3C Proposed Recommedation 24-August-1999, W3C, http://www.w3.org/TR/xhtml1, August 1999.
22. Stonebraker, M.: *Object-Relational DBMSs - The Next Great Wave*, Morgan Kaufman, 1996.