Seminararbeit

Synergiepotentiale bei der Ablage Dokumenten-orientierter Datenstrukturen in relationalen Datenbanken

Konzeption und Implementierung

Wirtschaftsinformatik 2 / Prof. Dr. Ludwig Nastansky

Betreuer: Dipl.-Wirt.-Inf. Ingo Erdmann

Sommersemester 2004

vorgelegt von:

Bernd Völlmeke
Studiengang: Wirtschaftsinformatik
Matr.Nr.: 6121060

Warburger Straße 60
33098 Paderborn

Inhaltsverzeichnis:

Abbildungsverzeichnis:

1. Einführung:

1.1. Szenario

Heutige Unternehmen sehen sich einem zunehmenden Kostendruck gegenüber. Die Konsolidierung und Zusammenführung von Systemen und Technologien wird hier vielfach gefordert, um den Management- und Administrationsaufwand zu reduzieren. Auf der Daten- und Informationsebene eröffnen sich hierdurch auch strategische Potenziale die sich zum Beispiel in einem verbesserten Wissensmanagement in Form einer verstärkten mehrfachen Nutzung von Informationen zeigen. Bezogen auf ein konkretes Szenario fallen in den meisten heutigen Unternehmen grundsätzlich zwei Typen von geschäftlichen Daten an. Zum einen so genannte „harte Daten", also hochstrukturierte Daten, (z.B. Stamm und Bewegungsdaten von Waren) die von betriebswirtschaftlichen Informationssystemen in relationalen Datenbanken verwaltet werden. Zum anderen existieren mit den so genannten „weichen Daten", semistrukturierte Daten, (z.B. Memos, Projektberichte) die meist dokumentenbasiert in Mailpostfächern oder anderen Dokumentendatenbanken abgelegt sind. Morschheuser spricht in diesem Zusammenhang von „konventionellen formatierten Daten und elektronischen Dokumenten (E-Dokumente)" [Morschheuser 1997].

"...it is almost hard to imagine any company that might have Domino that would not have a relational database system for some other application" [Tamura 2000]

Unter anderem um den Mehraufwand, der durch die Bereitstellung von zwei unterschiedlichen Systemen zur Datenspeicherung entsteht, zu reduzieren besteht für die neue Version 7 von Lotus Notes Domino[1] die Möglichkeit, alternativ eine (relationale) IBM Db2 als Datenspeicher für die serverseitigen (Dokument-)Datenbanken zu nutzen. Bisher kam hier ein proprietäres Datenformat zum Einsatz. Somit wird es möglich die gesamten Unternehmensdaten im Backend einheitlich relational zu halten und damit, sowohl die Kosten für administrativen Aufwand (z.B. Sicherungsstrategie) zu sparen, als auch von der Performance und Skalierbarkeit moderner relationaler Datenbanksysteme zu profitieren. Darüber hinaus entsteht durch diese Technologie eine neue relationale Schnittstelle, über die auf die in Notes Datenbanken enthaltenen Informationen zugegriffen werden kann. Damit ist eine Zusammenführung mit externen relationalen Daten schon auf Tabellenebene möglich, was eine Nutzung der Daten durch nahezu jede beliebige Anwendung ermöglicht.

[1] Lotus Notes Domino ist ein weitverbreitetes System zum dokumenten-orientiertem Workgroup Computing. Über die reine Message- und Organizer-Funktion hinaus lassen sich mit ihm dezentral dokumenten- und datenbank-orientiert Prozesse gestalten und unterstützen.

In dieser Seminararbeit sollen die Potentiale dieser neuen Möglichkeit, nämlich Lotus Notes Daten in womöglich schon vorhandenen, relationalen Datenbanken zu speichern, evaluiert werden. Besondere Aufmerksamkeit wird hierbei auf die neue relationale Schnittstelle gelegt, mit deren Hilfe eine prototypische Webanwendung implementiert wird. Sie ermöglicht es, Lotus Notes Daten, die in relationalen Datenbanken abgelegt sind in Kombination mit herkömmlichen relationalen Daten zu lesen und zu verändern. Die im Verlauf dieser Entwicklung angewandten Lösungskonzepte und die aufgedeckten Schwachstellen stellen ebenfalls Ergebnisse dieser Seminararbeit dar.

1.2. Aufbau der Arbeit

Nach der Einführung, werden im Kapitel 2 zunächst die Grundlagen dieser Seminararbeit, Datenbanken und Groupware, vorgestellt. In Kapitel 3 werden mit den unterschiedlichen Datenbanktypen und Schnittstellen zu Lotus-Notes Daten die Konzepte zur Umsetzung der im Kapitel 4 vorgestellten und diskutierten Webanwendung erläutert. Der Ausblick in Kapitel 5 beschäftigt sich zum Einen mit den technischen Aspekten der Umsetzung, aber auch mit noch zu klärenden Fragen im Hinblick auf Folgeprojekte. Abschließend werden in Kapitel 6 die während des Projektes gewonnenen Eindrücke, sowohl in Bezug auf die Nutzung von DB2 mit Domino im Allgemeinen, als auch die Möglichkeiten die sich durch die neuen relationalen Schnittstellen ergeben, zusammengefasst.

2. Vorstellung der Grundlagen

2.1. Datenbanken

“A database consists of some collection of persistent data that is used by the application system of some given enterprise.” [Date 2000]

Diese allgemeine Definition von Datenbanken gilt wohl für alle unterschiedlichen Arten von Datenbanksystemen, da sie auf den eigentlichen Zweck der Datenspeicherung abstellt. Die Vorteile einer Speicherung in standardisierten Datenbanken gegenüber der anwendungsspezifischen, proprietären Speicherung (z.B. im Dateisystem) liegen dabei zu allermeist im Bereitstellen der Information für mehrere Benutzer, im Sicherstellen der Zugriffskontrolle, sowie in der Koordination von Transaktionen.

Aus der Vielzahl der unterschiedlichen Datenbanksysteme (objekt-, objekt-relational-, hierarchisch-, netzwerk-, relational- und dokumentenorientiert) sollen hier nur die für das

Projekt relevanten Typen kurz vorgestellt werden, um damit die Grundlagen für die Diskussion Kapitel 3.1 zu legen.

2.1.1. Relationale Datenbanken

Relationale Datenbanken verdanken ihren Namen dem relationalen Datenmodell, dessen Prinzipien zuerst von E.F. Codd 1969/70 in „A Relational Model of Data for Large Shared Data Banks“ [Codd 1970] vorgestellt wurden. Für eine detaillierte theoretische Darstellung dieses Modells sei auf [Date 2000] verwiesen. Hier sollen die wichtigsten Aspekte des Datenmodells für den praktischen Einsatz kurz vorgestellt werden. Date unterscheidet drei entscheidende logische Aspekte dieses Datenmodells:

1) Struktureller- Aspekt: Alle verfügbaren Daten werden in Tabellen gehalten, die einer definierten Struktur entsprechen.
2) Integritäts- Aspekt: Oben genannte Tabellen erfüllen bestimmte Integritätsregeln, beispielsweise das jedes Tupel ein eindeutiges Schlüsselattribut besitzt.
3) Manipulativer- Aspekt: Alle Operatoren (z.B. restrict, project, join), welche dem Benutzer zur Verfügung stehen, produzieren als Ergebnisse wieder Tabellen.

Als spezielle Ziele relationaler Datenbanksysteme können dabei die Redundanzvermeidung, die Vermeidung von Inkonsistenzen sowie die Integritätserhaltung gesehen werden. Während die weitgehende Redundanzvermeidung durch Normalisierung der relationalen Datenbankstruktur beim Anlegen dieser erreicht wird, wird die Einhaltung der Integritäts- und Konsistenzregeln durch das DBMS als zentrale Ausführung- und Kontrolleinheit sichergestellt.

2.1.2. Dokumentenorientierte Datenbanken

Der Begriff Dokumentenorientierte Datenbanken ist in der Literatur und im allgemeinen Sprachgebrauch weniger scharf definiert als der Begriff der relationalen Datenbank. Ein Grund dafür ist, dass keine solide mathematische Grundlage – wie im relationalen Datenmodell – existiert. So werden als „Dokumenten-Datenbanken“ oft allgemein Anwendungssysteme bezeichnet die es ermöglichen, Dokumente zu speichern und aufzufinden (zum Beispiel diese Dokumenten-Datenbank zum Auffinden von Landesbeschlüssen). An diesem Beispiel wird deutlich, dass o.g. „Dokumenten-Datenbank“ ihren Namen aufgrund ihres Einsatzzwecks erhalten hat, und die Speicherung in dieser „Dokumenten Datenbank“ ohne Widerspruch zu ihrer Bezeichnung relational sein

kann. Eine dokumentenorientierte Datenbank soll deswegen im Rahmen dieser Seminararbeit nicht über ihre Funktion oder ihren Einsatzzweck, sondern vielmehr über die interne logische Struktur der Speicherung definiert werden. Diese interne logische Struktur der Speicherung besteht nicht wie bei der relationalen Datenbank aus zueinander in Beziehung stehenden Tabellen und den darin enthaltenen Datenfeldern, sondern geht davon aus, dass alle anfallenden Daten in Dokumenten gespeichert werden. Diese Dokumente können als Containerobjekte (vgl. Compound Documents) nahezu beliebige Datentypen aufnehmen. Die verschiedenen Dokumente einer Datenbank stehen a priori zunächst nicht in einer Beziehung zueinander. Diese wird entweder durch direkte Verlinkung einzelner Dokumente erreicht und/oder durch die (mehrdimensionale) Kategorisierung und Beschlagwortung von Dokumenten. Mit ihrer Hilfe kann durch Sichten (vgl. Bereitstellung von individuellen Sichten auf den Datenbestand) die auf eine oder mehrere der o.g. Ordnungskriterien abstellen eine sinnvolle Struktur in der Datenbank erstellt werden, die eine gezielte Navigation ermöglicht.

2.2. Groupware

Für den Begriff Groupware hat sich noch keine einheitliche Definition herausgebildet. Dies liegt u.a. daran, dass der Begriff Groupware sowohl einen konzeptionellen – Nutzung der Synergieeffekte des gemeinsamen Arbeitens – als auch einen technischen – die Software durch die der o.g. Effekt erzielt werden soll - Hintergrund untrennbar miteinander verbindet. Deutlich wird dies an der laut Dier ersten Definition von Groupware durch Peter und Trudy Johnson-Lenz im Jahre 1981:

„…intentional group processes plus software to support them. It has both computer and human components: software of the computer and "software" of the people using it." [Dier/Lautenbacher 1994]

Eine Zusammenstellung weiterer Definition findet sich ebenfalls bei Dier.

Im Rahmen dieser Seminararbeit, deren Ziel eine technische Implementierung ist, soll als Definition eine ebenfalls auf die technische Seite zielende Definition von Nastansky verwendet werden. Hiernach sollen als Groupware „Applikationen verstanden werden, welche die Computerunterstützung kooperativen Arbeitens ermöglichen" [Fischer et al 2000].

Diese Unterstützung gliedert sich nach Chaffey in die drei C's: „communication, collaboration and coordination" [Chaffey 1998] wobei dabei die collaboration

(Kooperation) und die coordination (Koordination) jeweils auf der communication (Kommunikation) aufbauen.

Im Folgenden sollen die Funktionalitäten von Groupware, die im weiteren Verlauf noch zu Sprache kommen kurz vorgestellt werden. Für weitergehende Betrachtung der unterschiedlichen Funktionalitäten sei auf [Fischer et al 2000] verwiesen.

Verteilte Datenbankarchitektur und Replikation

Die im Kapitel 2.1.2 beschrieben dokumentenorientierten Datenbanken bilden meist die Datenhaltung einer Groupwareanwendung. Die Daten werden dabei nicht zwingenderweise zentral gehalten sondern verteilt über mehrere Server an unterschiedlichen physischen Standorten, oder auch, meist nur Teile der Daten, auf stationären oder mobilen Arbeitsplätzen (Clients). Die Vorteile dieses Konzepts liegen nach Kremer (vgl. [Kremer 1999]) u.a. in der reduzierten Zugriffszeit, der erhöhten Verfügbarkeit und in der größeren Datensicherheit. Diese Vorteile werden durch Replikation erreicht, indem - aus der klassischen relationalen Sichtweise - die Nachteile der Datenredundanz sowie der (zumindest temporären) Dateninkonsistenz in Kauf genommen werden. Dies ist neben den im Kapitel 2.1 beschrieben und im Kapitel 3.1 diskutierten Unterschieden zwischen der dokumentenorientierten und relationalen Datenhaltung ein „elementar unterschiedliches Paradigma“ [Fischer et al 2000] und somit als Herausforderung für eine relationale Datenbank im dokumentorientierten Kontext zu sehen. Andere Anforderungen wie z.B. abgestufte Zugriffsmechanismen zur Nutzung einer Datenbank durch mehrere Benutzer sind, wenn auch technisch unterschiedlich realisiert, in beiden Datenbankarchitekturen ähnlich nutzbar.

Compound Documents

Diese bilden das zentrale Informationsobjekt in Groupware Umgebungen. Sie „enthalten als Container-Objekte strukturierte und unstrukturierte Informationen“ [Fischer et al 2000]. Somit können in Compound Documents als generischer Datentyp von einfachen Textfeldern bis zu multimedialen Inhalten alle Typen von Daten aufgenommen werden. Insbesondere muss es möglich sein, Daten anderer Anwendungen in die Compound Documents zu integrieren und innerhalb dieser, sie wieder mit ihrer Ursprungsanwendung zu bearbeiten. Die Darstellung der in den Compound Documents enthaltenen Daten sollte mit einer entsprechenden Maske so flexibel gestaltet sein, dass je nach Verwendungszweck nur bestimmte Inhalte angezeigt werden.

Bereitstellung von individuellen Sichten auf den Datenbestand

Um die Übersicht in einer gemeinsam genutzten dokumentenorientierten Datenbank mit tausenden Dokumenten zu gewährleisten, sind Sichten in zweierlei Hinsicht von entscheidender Bedeutung. Zum einen werden sie im Dokumenten- und Transaktionsmanagement eingesetzt um dem jeweiligen Bearbeiter seine aktuellen Aufgaben kenntlich zu machen, zum anderen dienen sie, in Kombination mit einer geeigneten Kategorisierung und Referenzierung, dem Knowledge Management in dem die vorhandenen Informationen jeweils im richtigen Kontext stehen und gefunden werden können. Sichten dienen also dazu dem Nutzer eine effektive Nutzung der vorhandenen Informationen zu ermöglichen und sollten daher „dem unterschiedlichen Informationsbedarf der Anwender durch Funktionen zur Strukturierung, Detaillierung und Komprimierung entgegenkommen“ [Fischer et al 2000].

3. Konzepte zur Umsetzung

3.1. Diskussion der unterschiedlichen Datenbankstrukturen

3.1.1. Dokumentenorientierte Datenbanken

Der größte Vorteil von dokumentenorientierten Datenbanken ist die Flexibilität der Datenhaltung. Es können in ihnen, ähnlich wie bei einem Dateisystem, nahezu beliebige Datenstrukturen abgelegt werden. Im Vergleich zu einem Dateisystem bieten dokumentenorientierte Datenbanken jedoch zahlreiche Vorteile. So ermöglicht die dokumentenorientierte Datenbank sowohl die Bereitstellung der Informationen für mehrere Benutzer, und deren Verteilung durch Replikation, als auch eine Rechteverwaltung, die im Allgemeinen über die Möglichkeiten eines Dateisystems hinausgehen. Weitere Vorteile liegen vor allem im Bereich der Informationsstrukturierung und der Informationssuche. Neben einer Volltextsuche bieten dokumentenorientierte Datenbanken die Funktion, strukturierte Informationen zu verwalten mit Hilfe derer es möglich wird eine Ordnung über die vorhandenen Daten (Dokumente) zu definieren.

Die Nachteile einer dokumentenorientierten Datenbank haben ihren Ursprung in der oben als Vorteil dargestellten Flexibilität. Durch das dokumentenorientierte Datenmodell, bei denen die einzelnen Dokumente grundsätzlich unabhängig voneinander gespeichert werden, ist es nicht möglich die Datenqualität durch eine automatische Kontrollinstanz sicherzustellen. Hierzu zählen insbesondere die referenzielle Integrität und die Vermeidung von Redundanzen, aber auch, durch die Möglichkeiten temporär unabhängige Repliken zu

halten, die automatische Konsistenzhaltung der Daten. Als Nachteil im Vergleich zu einem Dateisystem lässt sich noch anführen, dass zum Zugriff auf die dokumentenorientierte Datenbank zumeist ein eigener Client benötigt wird.
Dokumentenorientierte Datenbanksysteme sind meist eng mit der Anwendung für die sie entwickelt wurden verwoben. Es existiert keine einheitliche Definitions- und Abfragesprache und somit sind dokumentenorientierte Datenbanken nicht untereinander austauschbar. Als Schnittstellen werden verschiedene proprietäre Lösungen angeboten. Dies hat zur Folge, dass ein eigener Markt nur für dokumentenorientierte Datenbanken nicht besteht, sondern diese meist in Verbindung mit entsprechender Anwendungs- oder Middleware verkauft werden.

3.1.2. Relationale Datenbanken

Konkreten Ausprägungen von relationalen Datenbanken liegt immer eine zuvor genau definierte Datenbankstruktur zugrunde. Sie legt fest, welche Tabellen mit welchen Feldern in der Datenbank vorhanden sind und wie diese über Relationen zusammenhängen. Das Design dieser Datenbankstruktur (Datenbankschema) ist von besonderer Wichtigkeit, da dieser Bereich oft die Möglichkeiten der darauf aufbauenden Anwendung determiniert, und spätere Änderungen am Datenbankdesign nur mit erheblichem Aufwand möglich sind. Diese im Vergleich zu dokumentenorientierten Datenbanken deutlich höheren Anforderungen an die Strukturierung der Daten ermöglicht es, dass das DBMS die automatische Sicherung der Datenqualität – in logischer, nicht inhaltlicher Hinsicht – gewährleistet. So kontrolliert die Datenbank die Redundanz von Informationen, und kann die Einhaltung weiterer Integritätsregeln die sich entweder aus dem verwendeten relationalen Datenmodell ergeben oder explizit durch den Benutzer der Datenbank gefordert werden („business rules“ [Date 2000]), sicherstellen. Ein weiterer Vorteil der relationalen Datenbanken ist die Performance der Systeme gerade im Umgang mit großen Datenmengen, die sich ebenfalls aus der festen, einem mathematischen Modell folgenden, Struktur der Daten ergibt. So existieren innerhalb der Datenbanksysteme Optimierungsalgorithmen, die mit Hilfe von mathematisch-statistischen Algorithmen Anfragen so umstellen, dass möglichst wenige Rückgriffe auf Sekundärspeicher nötig werden. Große kommerzielle Datenbanksysteme ermöglichen es, die Leistung der Datenbank etwa durch clustern zu skalieren um so, eine auf für den jeweiligen Anwendungsfall passende Leistung bereitzustellen.

Im Vergleich zu dokumentenorientierten Datenbanken ist der Markt für relationale Datenbanken erheblich breiter, d.h. es existieren sehr viele unterschiedliche Anbieter für Datenbanksysteme unterschiedlichster Größen. Ein Grund hierfür ist sicherlich, dass mit SQL eine Datenbankdefinitions- und Abfragesprache existiert die in ihrer Grundform von allen Herstellern unterstützt wird, und eine Datenbank – zumindest in der Theorie – problemlos durch eine andere ersetzt werden kann.

3.1.3. Probleme beim Abbilden von Dokumentenorientierten Daten in relationalen Datenbanksystemen

In den vorhergehenden Abschnitten sind die jeweiligen Eigenschaften der verschiedenen Datenbanksysteme aufgezeigt worden. Es stellt sich nun die Frage wie diese unterschiedlichen Eigenschaften auf die Problemstellung wirken. Es ist deutlich geworden das sich dokumentenorientierte Datenbanken – und damit auch die in ihnen enthaltenen Daten – durch eine nahezu uneingeschränkte Flexibilität auszeichnen, während relationale Datenbanken ihre Stärken eher in der Verwaltung großer eindeutig strukturierter Daten haben (vgl. [Tamura 2000]). Um dieses Flexibilität in einen relationalen Datenbanksystem abzubilden bleibt, da das Datenbankschema zur Laufzeit nicht angepasst werden kann, nur die Möglichkeit, die Individualität der einzelnen Dokumente durch ein Binäres Feld (BLOB) in der Datenbank abzubilden. Durch ein solches Design können dann zwar beliebige Dokumente abgelegt werden, allerdings verliert die Datenbank damit sehr deutlich an Struktur, da ein Großteil der eigentlichen Nutzdaten eines Dokuments - ähnlich wie bei einem Dateisystem - in einem einzigen Objekt zusammengefasst ist. Reinhard Theis beschreibt in seinem Artikel diesen Zustand mit den Worten, dass, die dokumentenorientierten Daten „schräg im Datastore liegen“ [Theis 2004]. Diese Tatsache bleibt nicht ohne Folgen für das Daten Retrieval. Die Suche und das Sortieren nach Inhalten (beispielsweise Feldern in einem Dokument) die sich in diesem BLOB-Feld befinden sind nur mit SQL nicht mehr zu leisten, da es sich bei einer SQL Anfrage immer um eine deterministische Anfrage handelt, d.h. die Suchkriterien sind immer ein Teil des Ergebnis-Records (vgl. [Morschheuser 1997]) und ein genaues Wissen über die Struktur der angefragten Daten ist daher erforderlich. Somit wird zusätzlicher Implementierungsaufwand notwendig um mit einer relationalen Datenbank als Datastore die gleichen Funktionen (z.B. Volltextsuche) anbieten zu können wie mit einer herkömmlichen dokumentenorientierten Datenbank. Dies macht in vielen Fällen den durch die Verwendung von relationalen Datenbanksystemen erwartenden Performancegewinn

zunichte, da eben nicht die oben angesprochenen optimierten Wege der Anfragen benutzt werden können, sondern innerhalb der BLOB-Felder ähnliche Algorithmen wie bei einer dokumentenorientierten Datenbank verwendet werden müssen.

Diese Struktur der Speicherung macht es auch weitgehend unmöglich auf die Daten in der relationalen Datenbank mit externen Anwendungen zuzugreifen. Zwar ist es theoretisch denkbar die Daten aus dem BLOB-Objekt auszulesen, allerdings entspricht dieser Versuch ziemlich exakt dem Auslesen von Daten aus proprietären Dateiformaten. Es müsste also eine entsprechende Schnittstelle implementiert werden welche die Daten aus dem Objekt in eine verwertbare Form extrahiert. Ein Vorteil eines relationalen Datenbanksystems, die Datenunabhängigkeit (vgl. [Date 2000]), wäre somit also trotz Verwendung einer relationalen Datenbank nicht gegeben.

3.2. Vorstellung der bestehenden Möglichkeiten Domino Daten in Web-Applikationen zu präsentieren

In diesem Abschnitt sollen die wichtigsten Möglichkeiten kurz vorgestellt werden. Diese lassen sich in drei Unterkategorien aufteilen. Die erste Kategorie besteht aus der Möglichkeit die Notes standardmäßig zur Verfügung stellt, der Darstellung durch den integrierten Webserver. Zur zweiten Kategorie zählen die Nutzung des von speziellen Datenbanktreibern wie LDDJ oder NotesSQL sowie die Nutzung von APIs für verschiedene Programmiersprachen. Hierbei geschieht die Datenspeicherung weiterhin komplett in Notes und diese Daten werden über die erwähnten Schnittstellen ausgelesen und weiterverwendet. Lösungen der dritten Kategorie setzen schon einen Schritt früher an, so werden ausgewählte Daten bei DECS oder LEI basierten Lösungen erst gar nicht in der Notes Datenbank vorgehalten sondern gleich relational gespeichert. Diese relational gespeicherten Daten können dann durch externe Anwendungen genutzt werden.

Domino Webserver (Designer)

Seit dem Jahr 1996 (ab der Version 4.5) bietet der Lotus Domino Server die Möglichkeit die Domino-Anwendungen nicht nur im Notes Client sondern auch auf Webbrowsern anzuzeigen. Das Design der Web Applikation ergibt sich dabei automatisch aus der mit dem zugehörigen RAD-Tool – dem Domino Designer – entwickelten Anwendung. Allerdings ist es zumeist nötig die Anwendung durch zusätzliche Einstellungen oder eigene Designelemente für das Web anzupassen bzw. zu optimieren. Obwohl diese Möglichkeit sicherlich die schnellste und einfachste Methode darstellt, ist der Entwickler

hierdurch jedoch gleich in doppelter Hinsicht eingeschränkt, da er sowohl in Bezug auf die Entwicklungsumgebung (Domino Designer) als auch auf den Webserver (Lotus Domino) festgelegt ist.

Lotus Domino Driver for JDBC (LDDJ)

Dieser JDBC Treiber ermöglicht eine relationale Sicht auf Domino Daten aus einer Java Anwendung heraus. Er stellt Notes Forms als SQL Tabellen dar und Notes Views als SQL Views. Daneben bietet LDDJ mit der so genannten universellen Relation eine SQL Tabelle die alle Inhalte der Datenbank in einer Tabelle darstellt. LDDJ ermöglicht auch Joins zwischen verschiedenen Notes Views und mit „echten“ relationalen Daten. Schreibende Operationen sind hierbei auf die aus den Forms generierten Tabellen beschränkt. Geeignet ist diese Art des Zugriffs vor allem für in Domino Datenbanken abgelegte strukturierte Daten, wie z.B. Adressbücher. Als Typ 2 SQL Treiber stellt er dem zugreifenden Java-Programm die JDBC API zur Verfügung und benutzt zum Zugriff auf die Notes/Domino Datenbanken die Notes C API (vgl. Abbildung 1).

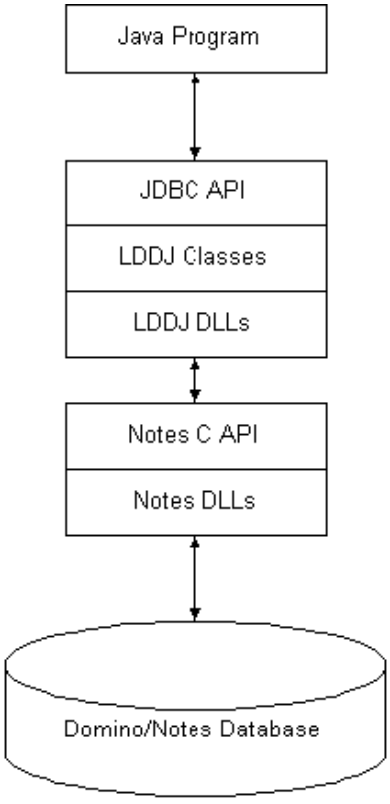


Abbildung 1 - Aufbau des LDDJ (Quelle [IBM3 2002])

Diese Struktur setzt voraus, dass eine Notes/Domino Installation (Client oder Server) auf dem den JDBC Treiber benutzenden Rechner vorhanden ist.

Ein Nachteil ist die Notwendigkeit, dass die Notes Datenbank an die Benutzung mit LDDJ in den meisten Fällen angepasst werden muss, da zahlreiche Eigenschaften der Datenbank

Einfluss auf die Funktion und Performance von LDDJ nehmen. Beispielweise gibt es keine Möglichkeit ein Zuordnung zwischen Notes Feldnamen und LDDJ Spaltennamen zu definieren was dazu führt, dass die Notes Feldnamen so gewählt werden müssen, dass sie nicht mit der JDBC Spezifikation in Konflikt stehen. Weiterhin bestimmt die Indexierung der Notes Datenbank maßgeblich die Performance der LDDJ Lösung, und es kann so u.U. nötig werden neue Views mit anderer Sortierung in die Datenbank einzufügen, um die entsprechenden LDDJ Operationen schneller ablaufen zu lassen. Problematisch ist weiterhin, dass es sich bei LDDJ um einen Treiber handelt der den JDBC 1.0 Standard nicht vollständig unterstützt. Zum einen fehlen einige Klassen und Methoden und zum anderen ist LDDJ auf kleine Datentypen beschränkt, d.h. es können keine BLOB Felder geschrieben oder ausgelesen werden. Dies führt dazu, dass die automatische Generierung von Datenobjekten auf Grundlage des Treibers in externen Entwicklungstools fehlschlägt, da diese Tools bestimmte Methoden voraussetzen um das Design der Datenbank zu ermitteln.

NotesSQL

NotesSQL bezeichnet einen Datenbank Treiber der genau wie der schon vorgestellte JDBC Treiber eine relationale Sicht auf Domino Datenbanken ermöglicht. Die im vorherigen Abschnitt beschrieben Eigenschaften und Einschränkungen treffen auf diesen ODBC Treiber analog zu. Unterschiedlich ist nur die den zugreifenden Programmen angebotene Schnittstelle, die bei NotesSQL, mit einigen Einschränkungen, der ODBC 2.0 Spezifikation entspricht (vgl. [IBM9 2003]).

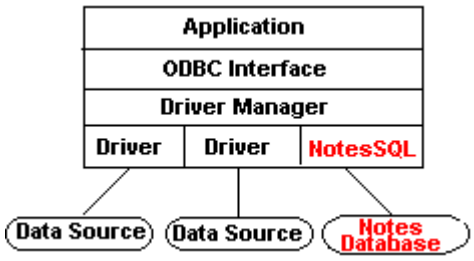


Abbildung 2 - NotesSQL: Aufbau ODBC (Quelle [IBM9 2003])

Notes Klassen für C, C++ und java

Eine weitere Schnittstelle die Lotus/Domino zum Zugriff auf seine Datenbanken anbietet, sind die APIs zu den Programmiersprachen Java und C(++). Über sie ist es möglich eine Verbindung zur Domino Datenbank herzustellen und so nicht nur auf die eigentlichen Dokumente lesend und schreibend, sondern auch auf weitere Eigenschaften der auf dem Server befindlichen Datenbanken zuzugreifen, so z.B. auf die ACL (vgl. [IBM1 2004]).

Die C-API bildet als älteste Schnittstellen die Grundlage für die objektorientierten C++ und Java Schnittstellen. So basiert die C++ Schnittstelle auf der C Schnittstelle (vgl. [IBM5 2004]), während es sich bei der Java-Schnittstelle um eine Menge von Wrapper-Klassen um die C-API handelt (vgl. [Huth/Erdmann/Hahnl 2003]). Die Schnittstellen können dabei sowohl mit Datenbanken die auf dem Domino Server abgelegt sind, als auch mit Datenbanken auf Notes Clients umgehen. Es ist ebenfalls möglich auf entfernte Datenbanken auf Domino Servern über eine CORBA Schnittstelle zuzugreifen. Besonders einfach ist dies mit den Java API zu implementieren, da die entsprechende Klasse (NCSO.jar), in denen u.a. die benötigten Stubs schon implementiert sind, nur in das eigene Programm eingebunden und genutzt werden muss.

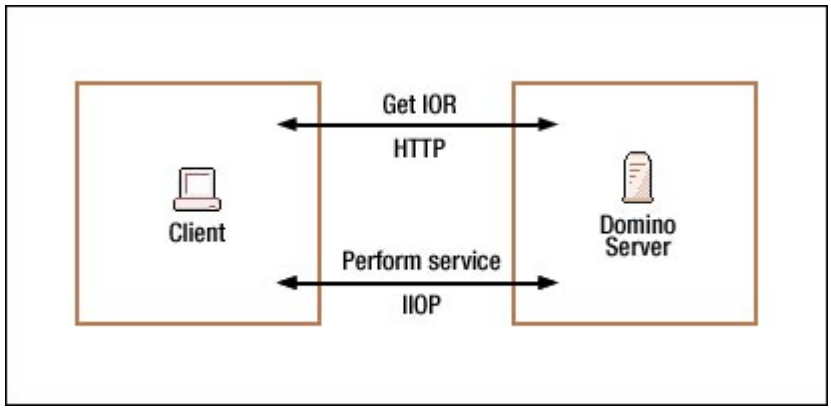


Abbildung 3 - Kommunikation Client - Server (Quelle: [Peron/Nikopoulos/Smith 2003])

Die Nutzung der CORBA Schnittstelle durch andere Programmiersprachen ist ebenfalls möglich, erfordert im Vergleich zur Java Implementierung allerdings einen etwas höheren Aufwand, da die Stubs Methoden mit Hilfe eines geeigneten IDL Compilers selbst erstellt werden müssen. (vgl. [IBM4 2001]).

COM

Hierbei handelt es sich um eine Microsoft Technologie die es ermöglicht, auf einzelne Komponenten einer Software zuzugreifen. So bildet COM z.B. die Grundlage für OLE mit der u.a Compound Documents realisiert werden (vgl. [Wikipedia 2004]). Im Bereich der Schnittstellen zu Domino/Notes Datenbanken ermöglicht COM den Zugriff auf Domino Objekte, durch Programmiersprachen die Zeiger unterstützen wie z.B. Visual Basic, VB for applications, VBScript, C++ und LotusScript. Hierdurch ist z. B. der Zugriff auf Notes Daten aus Office Anwendungen die VB unterstützen möglich (vgl. [Nielsen/Andenæs/ Smith 2000]). COM unterstützt nur den lokalen Zugriff auf Notes Datenbanken, d.h. es ist grundsätzlich nicht möglich auf entfernte Notes Daten nur Mittels COM zuzugreifen.

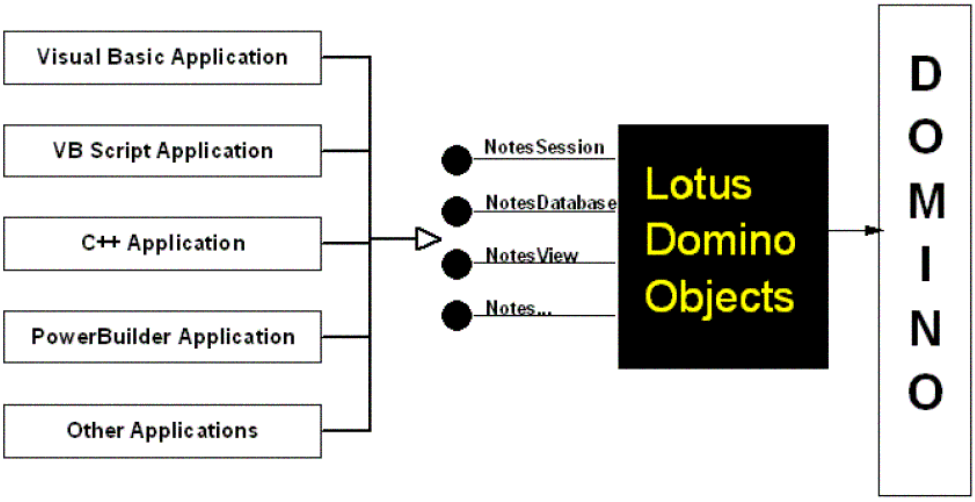


Abbildung 4 - Aufbau der Domino COM Schnittstelle (Quelle: [Nielsen/Andenæs/ Smith 2000])

Ein Zugriff für Programmiersprachen die keine Zeiger unterstützen – wie z.B. Java – ist nur durch Nutzung von CORBA möglich welche, die nativen COM Aufrufe kapselt und somit gleichzeitig auch einen verteilten Zugriff erlaubt.

Domino Enterprise Connectivity Service (DECS)

DECS ermöglicht eine bidirektionale Echtzeit Integration von Notes Daten in externe Datenbestände. Die Verbindung zu diesen Datenbeständen erfolgt über so genannte „Lotus Connectors“ die u.a eine Reihe von relationalen Datenbanken (z.B. DB2, Oracle und Sybase) unterstützen (vgl. [IBM7 2004]). In einem Notes Dokument werden virtuelle Felder definiert, die durch den auf dem Domino Server laufenden DECS Dienst überwacht werden. Lesende und schreibende Zugriffe werden so abgefangen und somit nicht in den Notes Datenbanken, sondern direkt im externen Datastore abgewickelt. Für den Anwender bleibt dieser Mechanismus völlig transparent, d.h. er arbeitet mit seiner Anwendung ohne zu wissen, welche Felder direkt in Notes Datenbanken gespeichert sind und welche in externen Datenquellen vorgehalten werden (vgl. [IBM2 2004]).

Somit wird ein Zugriff auf die Notes Datenbestände in der Art ermöglicht, dass auf per DECS, z.B. in relationale DB, ausgelagerte Daten mit externen Anwendungen ohne Notes Unterstützung zugegriffen werden kann. Allerdings müssen dafür für alle extern benötigten Daten Virtuelle Felder definiert werden, was einen nicht zu unterschätzenden Aufwand bedeutet, da jedes virtuelle Feld über sein eigenes „Activity Document“ verfügt in dem seine Auslagerungseigenschaften definiert sind.

IBM Lotus Enterprise Integrator (LEI)

Der Lotus Enterprise Integrator ermöglicht ganz allgemein das Kopieren und Verschieben von Daten zwischen Notes und unterschiedlichen externen Datenquellen. LEI arbeitet

dabei genau wie DECS mit den oben erwähnten „Lotus Connectors“ und ermöglicht damit die Anbindung von unterschiedlichsten Datenbanken. Zu diesem Zweck können die verschiedensten Aktivitäten definiert werden, die der LEI Server ausführt. So ist es u.a. möglich, geplant Daten zwischen verschiedenen Datenbanken zu kopieren und so Domino Daten für externe Anwendungen zur Verfügung zu stellen. Nachteilig an dieser Lösung ist, dass die Daten hierbei nur lesend weiterverarbeitet werden können, der Austausch nicht in Echtzeit geschieht und die Daten doppelt gehalten werden.

Eine Lösung für die oben beschriebenen Probleme, sind die „Advanced RealTime Activities“. Sie ermöglichen das Auslagern von Notes Daten in externe Datenspeicher. Dabei bietet der LEI zwei Varianten an. Zum einen besteht die Möglichkeit, analog zu DECS, virtuelle Felder in Dokumente zu integrieren, also Teile eines Dokuments z.B. relational zu speichern. Zum anderen unterstützt LEI virtuelle Dokumente, also Dokumente die komplett im externen Datenspeicher abgelegt sind. Virtuelle Dokumente haben gegenüber virtuellen Feldern eine Reihe von Vorteilen, so sind die Daten aus virtuellen Dokumenten auch in Views abrufbar. Ein weiterer Vorteil ist, das die eigentlich Notes Datenbank (nsf Datei) sehr klein gehalten wird, da sämtliche Felder im externen Datenspeicher abgelegt sind und, im Gegensatz zur Benutzung von virtuellen Feldern, keine „Key-Documents“ abgelegt werden müssen um den Bezug zu den entsprechenden Datenbankeinträgen herzustellen (vgl. Abbildung 5).

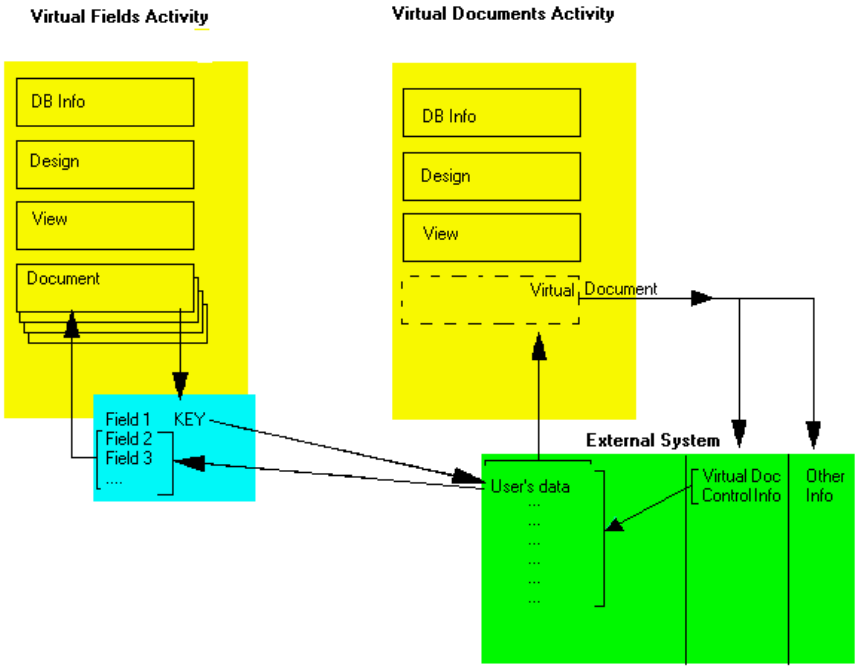


Abbildung 5 - Virtuelle Felder vs. Virtuelle Dokumente (Quelle: [IBM6 2004])

Darüber hinaus ist eine gleichzeitige Nutzung von virtuellen Feldern in virtuellen Dokumenten möglich. In diesem Fall wird zuerst das virtuelle Dokument aus der Datenbank erstellt und anschließend die virtuellen Felder aus einer (anderen) Datenbank bearbeitet. Damit ist es möglich auch in einem virtuellen Dokument auf Daten aus mehreren Backend Datenspeichern zuzugreifen (vgl. [IBM6 2004]).

4. Beschreibung der Beispielimplementierung

In diesem Kapitel wird die prototypische Beispielimplementierung erläutert. Während im Kapitel 4.1 zunächst kurz die Funktionalität der Anwendung gezeigt wird und mögliche Einsatzszenarien für diese Funktionalität dargestellt werden, befasst sich Kapitel 4.2 mit dem technischen Aufbau der Lösung. Anschließend werden im Kapitel 4.3 Schwachstellen und Probleme der implementierten Lösung aufgezeigt und mögliche Lösungsansätze diskutiert.

4.1. Fähigkeiten der prototypischen Lösung und denkbare Einsatzszenarien

Der Prototyp arbeitet mit Adressdaten aus dem Pavone Enterprise Office System, die in einer Notes Datenbank abgelegt sind. Sowohl innerhalb des Notes Clients als auch über die Webanwendung ist es möglich, die vorhanden Adressen einzusehen, zu verändern und zu löschen, als auch neue anzulegen. Da es sich um einen Prototyp handelt, der in erster Linie die Funktionsweise darstellen soll, wurde nur eine Auswahl der Daten die in Enterprise Office für eine Adresse gespeichert werden in die Webanwendung übernommen. Die in der Webanwendung zugreifbaren Angaben beschränken sich auf: Name, Vorname, E-Mail Adresse, Telefonnummer, PLZ, Stadt und Strasse. In der Web Anwendung besteht darüber hinaus die Möglichkeit den jeweiligen Personen (Adressen) ein Gehalt zuzuweisen, zu verändern und wieder zu löschen. Diese Information ist, als Beispiel für die Kombination von Notes Daten mit anderen Anwendungen/Daten, in einer separaten Tabelle abgelegt. Es ist also möglich die Informationen zweier unterschiedlicher Datenquellen unter der Oberfläche der Webanwendung einheitlich zu bearbeiten. Weiterhin bietet die Webanwendung die Möglichkeit eines Logins unter Angabe des Benutzernamens und des Passworts. Diese Benutzer, die sich ursprünglich auf die DB2 Datenbank beziehen, werden in Notes einem Notes User zugeordnet, was dazu führt, dass der entsprechende Benutzer in der Webanwendung über zu denen in der Notes Anwendung analogen Zugriffsrechten verfügt. Für eine detaillierte Erläuterung dieses Sicherheitskonzeptes sei auf Kapitel 4.2.2.1 verwiesen.

Die Webanwendung beweist somit die grundsätzliche Möglichkeit, über die neu geschaffene relationale Schnittstelle auf Domino Daten wahlfrei zuzugreifen, und darüber hinaus diese Daten mit anderen zusammenhängenden Daten unter einer Oberfläche zu bearbeiten. Somit ist es nicht nur möglich die Datenhaltung für „harte“ und „weiche“ Unternehmensdaten in einem einheitlichen System abzuwickeln, sondern zusätzlich von

der Verknüpfung ausgewählter Teildaten der beiden Datentypen zu profitieren. Durch diese Verknüpfung kann im Idealfall eine vollständige Integration der Daten erreicht werden, so dass beispielsweise die doppelte Pflege von Mitarbeiterstammdaten entfallen kann. Kritisch ist hierbei anzumerken, dass die Möglichkeiten der Verknüpfung der harten und weichen Daten auch von dem zur Verwaltung der harten Daten eingesetztem System abhängig sind. Diese Seite wurde allerdings im Rahmen der Seminararbeit nicht betrachtet, da sie aufgrund der Vielzahl unterschiedlicher Systeme jeweils für den konkreten Einzelfall untersucht werden muss.

4.2. Aufbau der Lösung

Nach der im vorherigen Kapitel vorgenommen Beschreibung der Fähigkeiten der Anwendung soll in diesem Kapitel insbesondere auf die technische Realisierung eingegangen werden. In den einzelnen Unterkapiteln wird zunächst die zugrunde liegende Notes Anwendung vorgestellt, um anschließend auf die drei Schichten der Webanwendung: die Daten-, Modell- und Präsentationsschicht, einzugehen. Zur Entwicklung der Anwendung wurden hauptsächlich drei Softwareprodukte verwendet. Zunächst einmal die DB2 Datenbank mit ihren enthaltenen Werkzeugen zur Konfiguration der Datenbank sowie die Domino Produkte: Server, Admin, Designer und der Notes Client. Die Entwicklung der Webanwendung wurde mit dem Websphere Applikation Developer (WSAD) 5.1.2 vollzogen, da dessen Oberfläche eine grafische Benutzung des JSF-Frameworks unterstützt. Auf dem im WSAD integrierten Testserver wird die Anwendung auch ausgeführt (vgl. Abbildung 6).

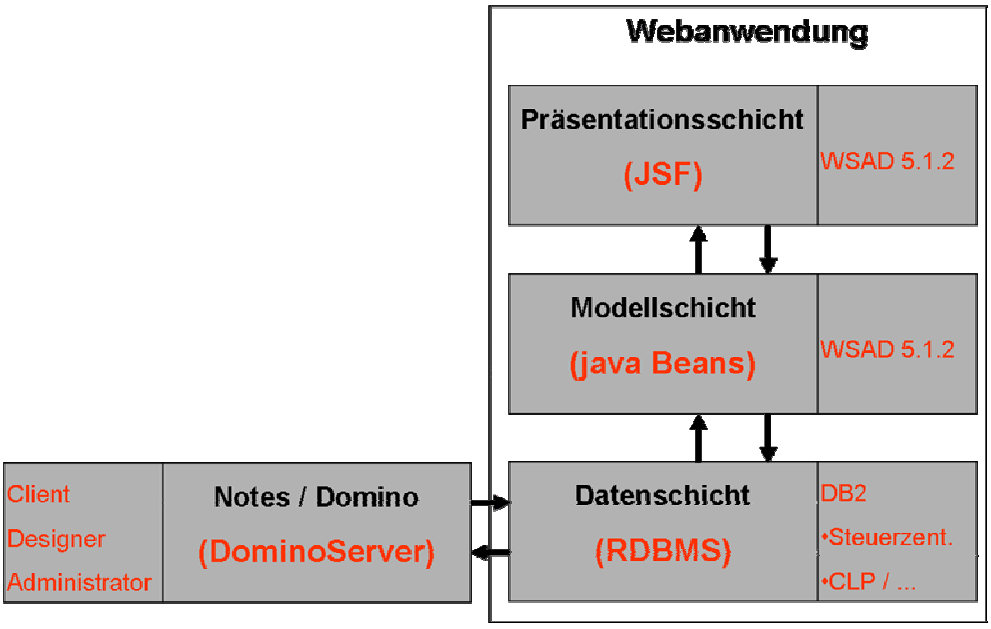


Abbildung 6 – Aufbau des Prototyps und benutzte Entwicklungswerkzeuge

Bei der folgenden Beschreibung liegt das Hauptaugenmerk auf den Teilen der Anwendung welche neue Möglichkeiten, also insbesondere die DB2 Anbindung, nutzen. Die übrigen Bestandteile der Anwendung werden soweit erklärt, wie es zum Verständnis des Gesamtkonzepts notwendig ist.

4.2.1. Zugrunde liegende Notes Anwendung / Datenbank

Ausgangspunkt der gesamten Anwendung ist dass schon erwähnte Enterprise Office System. An diesem aus mehreren Notes Datenbanken bestehenden System mussten im Rahmen dieser Seminararbeit lediglich zwei Änderungen vorgenommen werden. Zunächst wurden die Datenbanken auf einen DB2 fähigen Domino Server kopiert um die klassischen nsf Datenbanken in DB2 abzuspeichern. Anschließend wurde in die Hauptdatenbank, der Office Datenbank, ein Access View eingefügt, der die für die Webanwendung relevanten Daten SQL Anwendungen zugänglich macht. Die Access Views sind ein in Domino 7 beta neu enthaltenes Designelement, welches nur für in DB2 gespeicherte Domino Datenbanken zur Verfügung steht. Auf die genaue Funktionsweise wird im folgenden Kapitel 4.2.2 eingegangen.

4.2.2. Konzept der DB2 Anbindung

Der prinzipielle Aufbau der Domino – DB2 Anbindung ist aus Abbildung 7 ersichtlich. Der Domino Server greift auf eine in DB2 abgelegte Domino Datenbank zu, indem er sich mit der DB2 Datenbank verbindet, und die entsprechenden Information aus der DB2 Datenbank liest bzw. schreibt. Für den zugreifenden Notes Client ist dieser Vorgang völlig transparent, da der Domino Server den Zugriff für den Client maskiert.

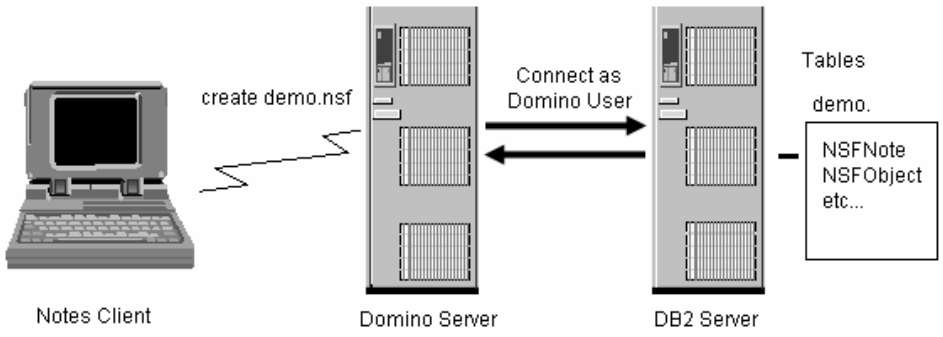


Abbildung 7 - Aufbau Domino Server mit DB2 (Quelle [IBM8 2004] (leicht angepasst))

Diese Eigenschaft wird an zwei Stellen besonders deutlich, zum einen benötigt der Domino Client keinen DB2 Zugang, da der Domino Server die Zugriffe und deren Kontrolle übernimmt, und zum anderen ist auf dem Client die Replikation auch mit DB2 basierten Domino Datenbanken weiterhin uneingeschränkt möglich.

Die Speicherung der Daten erfolgt in der DB2 Datenbank durch eine spezielle Tabellenstruktur. Insbesondere werden die individuellen Felder der einzelnen Dokumente in einer BLOB Struktur gespeichert. Aus diesem Grund ist es wie schon in Kapitel 3.1.3 angedeutet, wenig sinnvoll den Versuch zu unternehmen, auf diese Struktur mit einer externen Anwendung zuzugreifen, da die Schnittstelle zu dieser Struktur jedes Mal neu implementiert werden müsste. Daher bietet Domino mit den im nächsten Kapitel beschrieben Access Views eine einheitliche Schnittstelle an die es externen Applikationen per SQL ermöglicht auf diese Daten zuzugreifen.

4.2.2.1. Access Views als Schnittstelle zu SQL

Access Views dienen dazu Teile der Domino Daten externen Anwendungen als DB2 – also einem relationalen - View zur Verfügung zu stellen. Darüber hinaus sind sie Grundlage für die in Kapitel 4.2.2.2 vorgestellten Query Views die eine View Generierung über SQL-Abfragen auch für Notes Views ermöglichen. Diese relationale Sicht auf die Domino-Daten ist prinzipiell Analog zu der die auch in LDDJ bzw. NotesSQL verwendet wird. Grundsätzlich stellen Dokumente (Notes) eine Zeile in einer Tabelle da und die Spalten beinhalten die in den Dokumenten enthaltenen Felder.

Der Eingliederung der Access Views in die oben beschriebene Domino - DB2 Struktur verdeutlicht Abbildung 8

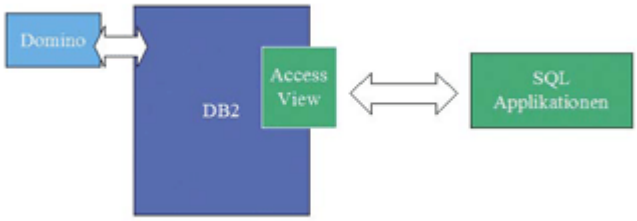


Abbildung 8 - Eingliederung der Access Views (Quelle [Theis 2004])

Access Views lassen sich in allen Domino-Datenbanken die DB2 als Datenspeicher verwenden, mit Hilfe des Designers anlegen. Hierbei werden zunächst die Datenfelder der Domino-Datenbank die im Access View enthalten sein sollen ausgewählt. Es können Datenfelder die aus Forms und Subforms stammen und darüber hinaus Shared Fields in den Access View einbezogen werden, auch ist es nicht erforderlich das die enthaltenen Datenfelder aus dem gleichen Form stammen.

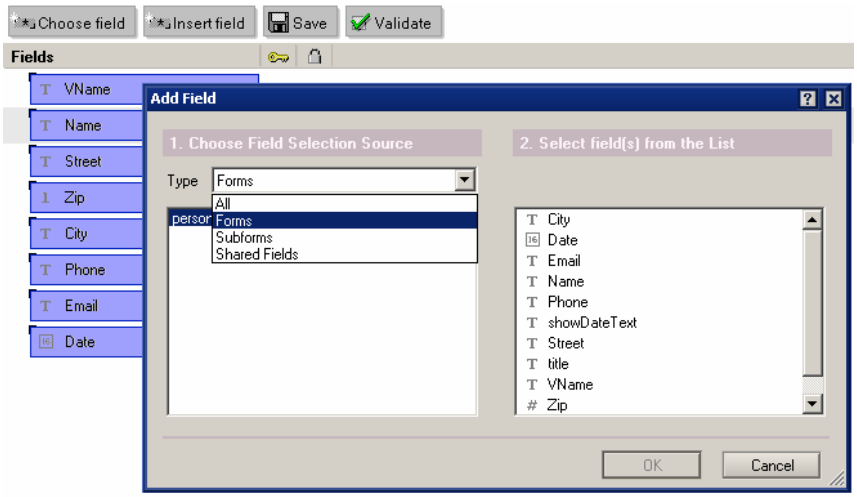


Abbildung 9 - Auswahl der Felder für einen Access View

Für jedes ausgewählte Feld wird automatisch eine Zuordnung auf einen passenden DB2 Datentyp durchgeführt, diese Zuordnung lässt sich zusätzlich von Hand anpassen. So kann bei Zeitangaben beispielsweise bestimmt werden ob nur das Datum oder die Zeit in den Access View übernommen werden oder sämtlich Informationen in Form eines Timestamps. Eine Einschränkung stellen die RichText Felder dar, sie können zwar in Access Views einbezogen werden (bei der Auswahl erscheinen sie allerdings rot), bleiben aber bei der späteren Nutzung des Access Views immer leer, selbst wenn sich im entsprechenden Notes Feld Daten befinden da dieser Datentyp nicht auf einen DB2 Datentyp abgebildet werden kann.

Für den Access View muss nun noch definiert werden welche Dokumente grundsätzlich in ihm enthalten sein sollen. Diese Festlegung wird über die Auswahl von mindestens einem bis zu allen in der Datenbank enthaltenen Forms realisiert. Jedoch stellt diese Auswahl nur eine Vorauswahl dar, die bei der späteren Nutzung durch entsprechende SQL Statements weiter eingeschränkt werden kann. Außerdem können zusätzlich zu den bereits ausgewählten Feldern noch weitere, für jedes Dokument verfügbare Felder, hinzugefügt werden. Hierbei handelt es sich um die UNID, die später als Schlüssel verwendet werden kann und die „modified time“. Bei Nutzung der Access Views durch eine externe Anwendung ist es von besonderer Wichtigkeit, dass die Schlüsselgenerierung konsistent zu der nativen Notes Umgebung verläuft damit keine Konflikte auftreten. Zu diesem Zweck kann im Access View festgelegt werden durch welches in der Notes Datenbank vorhandene Form, einfüge und update Operationen die auf Seite des DB2 – Views erfolgen, verarbeitet werden sollen. Dies hat zu Folge, dass die im DB2-View, also extern, erzeugten Dokumente zu vollständigen Notes-Dokumenten werden, d.h. insbesondere eine

NoteID sowie eine UNID bekommen. Damit ist die Konfiguration des Access Views abgeschlossen.

Nachdem die Erstellung und Konfiguration der Access Views dargestellt wurde soll im diesem Abschnitt auf deren Sicherheitskonzept und dessen Realisierung eingegangen werden. Ziel der Access Views ist es die in der Notes Datenbank festgeschriebene Sicherheits-Policy auch beim Zugriff von außen zu erhalten. Das bedeutet, dass sämtliche Rechte die in der ACL definiert sind auch für den externen Zugriff gelten sollen, so beispielsweise das Recht, neue Dokumente zu erstellen oder Vorhandene zu ändern. Zu diesem Zweck werden in einem ersten Schritt den vorhanden Notes Benutzern DB2 Benutzernamen zugeordnet. Dies geschieht im Domino Directory über den Eintrag des DB2 Benutzernamens als „Network Account Name“ im Profil des jeweiligen Benutzers. Die Überprüfung der Rechte erfolgt dann zur Laufzeit über einen extra einzurichtenden UDF-Server (vgl. [IBM8 2004]). Dieser UDF-Server stellt das Bindeglied zwischen DB2 und Domino dar. Sobald über SQL auf die Domino Daten eines Access Views zugegriffen wird, baut dieser UDF-Server eine Verbindung zum Domino Server auf und stellt die Zugriffsrechte des DB2 Benutzers mit Hilfe der im Domino Directory gespeicherten Zuordnung, DB2 zu Domino Benutzer, fest. Hieraus ergibt sich, dass, obwohl die Daten vollständig in DB2 abgelegt sind, Zugriff auf den Domino Server vorhanden sein muss, um die Zugriffsrechte zu garantieren. In der Praxis führt dieser Mechanismus dazu, dass zwei gleiche SQL Anfragen - in Abhängigkeit vom für diese SQL Anfrage verwendeten Benutzer - unterschiedliche Ergebnisse liefern können. Dieser Sicherheitsmechanismus kann, wenn für bestimmte Anwendungen Zugriffsrechte auf Benutzerebene nicht erforderlich sind, durch einen Konfigurationseintrag im Domino Server auch abgeschaltet werden. In diesem Fall sind nur die direkt in DB2 vergebenen Rechte für den Zugriff auf die Access Views maßgeblich.

4.2.2.2. Exkurs: Query Views – Neue Möglichkeiten der View Definition in Notes

Query Views bilden innerhalb von Notes das Gegenstück zu den oben vorgestellten Access Views. Sie sind damit Teil einer Notes Datenbank und erscheinen im Designer als spezielle Art eines Views. Mit ihrer Hilfe können Notes Views erstellt werden die auf SQL Abfragen basieren, also Daten beinhalten die in relationalen Datenbanken abgelegt sind. Insbesondere können erstellte Access Views somit nicht nur dazu genutzt werden Domino Daten für externe Anwendungen zur Verfügung zu stellen, sondern es ist mit Query Views

ebenfalls möglich diese Access Views zur Generierung von Views in Domino zu verwenden.

Query Views haben gegenüber klassischen Notes Views zahlreiche Vorteile. So ist es nicht nur möglich externe Daten in Notes darzustellen sondern ebenfalls dynamische Abfragen zu generieren in dem bestimmte Teile des SQL Ausdrucks erst zur Laufzeit der Notes Applikation eingefügt werden. Da diese Abfragen dynamisch generiert und ausgeführt werden, verbrauchen Query Views keinen Platz in der Datenbank zur Speicherung eines Indexes. Die Funktion auf externe Daten zuzugreifen ermöglicht es, einen Query View zu erstellen der Daten aus mehreren Notes Datenbanken zusammenfasst indem er auf die in den Notes Datenbanken erstellten Access Views zugreift und diese zusammenführt.

Im Gegensatz zur einfachen Nutzung von DB2 als Backend für Domino Datenbanken, bei der die Zugriffstruktur der Notes Anwendung vollständig erhalten bleibt, ist die Zugriffskontrolle bei der Verwendung von Query Views die auf Access Views basieren ein Aspekt der beachtet werden muss, damit diese nicht die Notes Zugriffsstruktur unterlaufen. Aus diesem Grund gelten für die Zugriffe auf die Access Views durch Query Views die gleichen Zugriffsregeln wie für die externe Nutzung der Acces Views. Die späteren Nutzer des Query Views müssen also, genau wie bei der externen Nutzung der Access Views (siehe Kapitel 4.2.2.1), über einen DB2 Zugang verfügen der mit ihrem Notes Benutzer verbunden ist um auf die entsprechenden Inhalte zugreifen zu können (vgl. Abbildung 10).

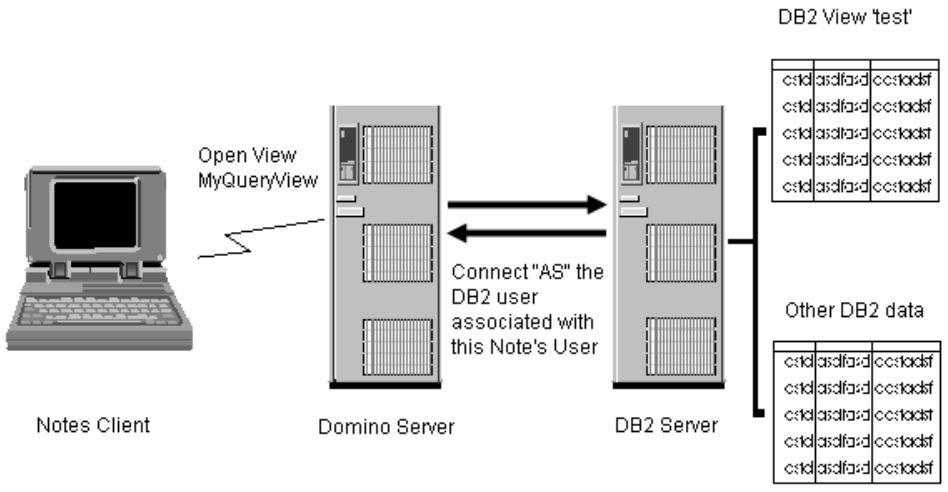


Abbildung 10 - Zugriff auf Query Views (Quelle [IBM8 2004] (leicht angepasst))

Aufgrund der oben beschriebenen Vorteile erweitern Query Views die Fähigkeiten von Views in Notes Anwendungen ganz entscheidend. Die größte in der Beta 2 vorhandene Funktionseinschränkung ist allerdings, dass das Öffnen von Dokumenten die im Query View angezeigt werden nur möglich ist, wenn in der entsprechenden Select Anfrage das Feld „NOTEID“ inkludiert ist und, was schwerer wiegt, die entsprechenden Dokumente

Teil der Notes Datenbank sind aus der der entsprechende Query View aufgerufen wurde. Damit lassen sich Dokumente aus mehreren Datenbanken in einem View zwar zusammenfassen, allerdings entfällt die Möglichkeit diese Dokumente auch zu bearbeiten. Unter Umständen wird diese deutliche Einschränkung bereits in zukünftigen Versionen durch eine Verlinkung gelöst, die in der finalen Version bereits ein öffnen aller Dokumente in einem Query View erlaubt.

4.2.3. Das Modell der Webanwendung

Aufgabe des Modells ist die Kommunikation mit der darunter liegenden Datenschicht; im konkreten Fall also mit den Access Views, und die Bereitstellung dieser Daten für die darüber liegende Präsentationsschicht. Die im Rahmen dieser prototypischen Anwendung zum Einsatz gekommene Entwicklungsumgebung WSAD 5.1.2 bietet sowohl Unterstützung für JSF als auch für Datenobjekte, den SDOs, die dem Entwickler beim Umgang mit relationalen Datenquellen unterstützen. Die Nutzung dieser Datenobjekte erwies sich allerdings im konkreten Fall als unmöglich, da die in der Version WSAD 5.1.2 nutzbaren SDOs grundsätzlich nur mit Datenbank-Tabellen als Datenquelle zusammenarbeiten. Bei den Access Views handelt es sich allerdings, wie der Name schon vermuten lässt, um echte DB2, also relationale, Views. Aufgrund dieser Tatsache wurden für die Anwendung als Modell einfache java Beans genutzt. WSAD unterstützt die Verwendung von java Beans insoweit, dass diese ebenfalls als Datenquelle grafisch in die Oberfläche eingebunden werden können und so, genauso wie bei den SDOs ein grafisches Binding an JSF Komponenten möglich ist. Allerdings müssen diese Beans, im Gegensatz zu den SDOs, vorher selbst erstellt werden, d.h. dass neben den Attributen mit den entsprechenden Getter und Setter Methoden, auch die Datenbankoperationen per Hand geschrieben werden müssen.

In der Anwendung kommen insgesamt zwei Beans zum Einsatz. Zum einen die „Login“ Bean die lediglich zur Speicherung und Überprüfung der Identität des aktuell angemeldeten Benutzer eingesetzt wird. Zum anderen die „Person“ Bean in der die Attribute einer Adresse vorgehalten werden. Über das Attribut „all“ kann ein Array aller in die Datenbank eingetragenen Adressen abgerufen werden. Des Weiteren enthält die „Person“ Bean eine „Login“ Bean. Hierdurch kann der aktuelle Benutzer für den Aufbau der Verbindung zur Datenbank benutzt werden, und somit, in Kombination mit der Rechteverwaltung durch den UDF Server, nur Datenbankoperationen ausführen zu denen

er nach der ACL berechtigt ist. Für eine detaillierte Betrachtung des Aufbaus der Beans sei auf die Klassendiagramme und den Quellcode im Anhang verwiesen.

4.2.4. Die Präsentationsschicht der Webanwendung

Die Oberfläche der Webanwendung ist, da der Focus auf der Funktionalität des Lösungsansatzes liegt, bewusst schlicht gehalten, ermöglicht aber Zugang zu allen Funktionen die nötig sind, um die durch den Access View zugänglich gemachten Adressdaten zu manipulieren. Zusätzlich sieht sie zur Authentifizierung noch ein Login vor und ermöglicht das Hinzufügen von Gehaltsinformationen als Beispiel für die Modifikation von externen Daten.

Bei den aktiven Elementen der Oberfläche handelt es sich ausschließlich um die im WSAD bereitgestellten JSF Komponenten, so dass diese komplett grafisch erstellt werden konnten. Die Logik der Anwendung besteht grundsätzlich aus der Navigation, der Initiierung von Datenbankoperationen und dem Anzeigen und Verstecken von JSF-Komponenten entsprechend dem Status der Anwendung.

Die Navigation läuft wie bei JSF üblich über die Definition von Navigations-Regeln, die sich in WSAD über eine entsprechende Oberfläche definieren lassen. Aufgrund der geringen Größe der Anwendung wurden alle Navigations-Regeln global, d.h. für alle Seiten gültig, definiert. Zum Aufruf dieser Regeln müssen nun noch in den entsprechenden Aktionen, beispielsweise im Code der beim Betätigen eines Buttons abgearbeitet wird, die String Rückgabewerte definiert werden.

Die Aufrufe von Modell-Methoden, die Datenbankoperationen ausführen, konnten, da es sich um ein selbst erstelltes Modell handelt, nicht mit Unterstützung durch Assistenten erstellt werden, sind aber aufgrund der wenig komplexen Struktur der zugrunde liegende Bean sehr einfach aufgebaut. So geben alle Datenbankoperationen einen String zurück der bei erfolgreicher Ausführung null ist und ansonsten die von der Datenbank zurückgegebene Fehlermeldung enthält. In der Anwendung muss dann nur dieser Ausgang der Operation überprüft werden und gegebenenfalls die Fehlermeldung als neue Fehlermeldung dem FacesContext – der Umgebung einer Faces Anwendung – hinzugefügt werden. Das Framework sorgt dann automatisch für ein erneutes Anzeigen der Seite wobei die Fehlermeldung im entsprechenden Fehlermeldungsbereich angezeigt wird.

Das Verstecken und Anzeigen von Bedienelementen kommt vor allem beim Hinzufügen und Entfernen von Gehaltsinformationen zur Anwendung. So muss bei nicht vorhandenen

Gehaltsinformationen, ein Button angeboten werden um diese anzulegen und umgekehrt bei vorhandenen Gehaltsinformationen die Möglichkeit geschaffen werden, diese wieder zu entfernen. Die dafür nötige Logik ist bereits in der Bean vorhanden. So existiert ein Boolean Attribut „sal“ das true ist wenn dieser Person ein Gehalt zugeordnet ist. Dieses wird dann als rendered Attribut in dem entsprechenden Button (negiert) benutzt, und bestimmt somit, welche dieser Elemente angezeigt werden. Diese Definition kann im WSAD, wenn auch etwas versteckt, bis auf die Negation vollständig grafisch durchgeführt werden, da sich die entsprechende Information in der Bean befindet die dem WSAD schon zum Entwicklungszeitpunkt bekannt ist.

4.3. Kritik der Lösung und der an der Lösung beteiligten Software – Komponenten

Die oben vorgestellte Lösung beweist die Einsatzfähigkeit der DB2 Integration und insbesondere der Access Views, als Schnittstelle zu SQL basierten Anwendungen. Auch ist eine Kombination mit JSF als Darstellungstechnologie unter Verwendung des WSAD 5.1.2 möglich. Neben den bei der Verwendung von Beta Software zu erwartenden Fehlern die in der finalen Version behoben sein müssten, bleiben zusätzlich einige Aspekte die noch Entwicklungspotential für zukünftige Versionen der eingesetzten Software bergen. Diese sollen in diesem Abschnitt vorgestellt werden.

Rich Text Felder

Der Zugriff auf diese Felder durch Access Views bleibt in der vorliegenden Version, völlig unmöglich, da sie zwar in Access Views im Designer eingebunden werden können, aber die entsprechenden Spalten zu Laufzeit grundsätzlich leer bleiben. Das Problem welches sich an dieser Stelle ergibt ist grundsätzlich vergleichbar mit dem Problem Domino Dokumente in einer relationalen Datenbank zu verwalten. Auch bei den Richt Text Feldern steht deren Flexibilität im Gegensatz zu der Strukturiertheit einer relationalen Datenbank. In diesem Fall lässt sich ein Richt Text Feld nicht einem DB2 Datentyp zuordnen. Eine Lösung für dieses Problem ist nur schwer zu finden. Möglich wäre es z.B. die enthaltenen Inhalte in einen HTML String umzuwandeln. So könnte ein solches Feld im Access View vom DB2 Datentyp LongVarchar sein und das Zuordnungsproblem wäre gelöst. Allerdings würde auf diese Weise nur ein Teil des möglichen Inhalts des Richt Text Feldes – die Formatierung – in DB2 dargestellt werden. Auch ist eine Darstellung als HTML sicherlich nicht für jede Anwendung die ideale Darstellungsweise. Eine andere Möglichkeit wäre die Darstellung als BLOB im Access View, welche dann in eine java Objekt überführt werden kann in dem die einzelnen Inhalte des Richt Text Feldes abfragbar sind. Diese Möglichkeit

würde allerdings wieder speziell auf eine Programmiersprache ausgelegt sein und eine Nutzung von Rich Text Feldern mit automatischen Datenobjekten behindern, da diese ebenfalls speziell für solche Objekte konzipiert werden müssten. Abschließend lässt sich sagen, dass, obwohl eine optimale Lösung des Problems nicht möglich zu sein scheint, eine wie auch immer geartete Möglichkeit des Zugriffs auf Rich Text Felder dem bisherigen Zustand, in dem ein Zugriff grundsätzlich nicht möglich ist, vorzuziehen wäre.

Zugriffsrechte in DB2

Wie in Kapitel 4.2.2.1 erläutert gelten für die in den Access Views dargestellten Domino Daten die gleichen Zugriffsrechte wie sie auch für den zugeordneten Domino Benutzer gelten. Problematisch ist hierbei allerdings, das seitens des externen Programms, in diesem Fall der Webanwendung, der Umfang dieser Rechte nicht bestimmt werden kann. Ob ein bestimmtes Recht vorhanden ist kann somit nicht im Vorhinein geprüft werden, sondern zeigt sich erst beim Versuch die entsprechende Datenbankoperation auszuführen, die dann entweder ausgeführt wird oder scheitert. Das Scheitern einer Datenbankoperation wird durch eine Fehlermeldung mit einem zugehörigen Fehlerstring signalisiert. Abbildung 11 zeigt eine solche Fehlerausgabe in der Webanwendung, in diesem Fall wurde versucht ein Dokument zu verändern obwohl der Benutzer nicht über die erforderlichen Rechte verfügt.

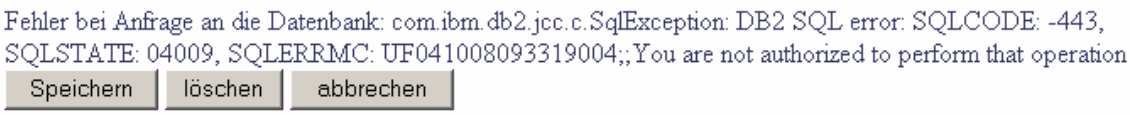


Abbildung 11 - Fehlermeldung bei nicht vorhandenen Zugriffsrechten

Die Möglichkeit die Rechte des betreffenden Benutzers für die in einem Access View befindlichen Daten im Vorhinein abzufragen, würde vor allem die Benutzbarkeit der Webanwendung verbessern, da Aktionen zu denen der betreffende Benutzer nicht berechtigt ist, in der Oberfläche erst gar nicht angezeigt werden müssten. So könnte in der Webanwendung z.B. der Button der es erlaubt die Daten eines Mitarbeiters zu ändern, ausgeblendet werden falls der betreffende Nutzer nicht über die erforderlichen Rechte verfügt.

Unterstützung von Views durch SDOs

In der hier vorgestellten Beispielimplementierung wurden für das Modell der Webanwendung java Beans verwendet. Die Verwendung der in WSAD 5.1.2 vorhandenen Datenobjekte, den SDOs, war unmöglich, da diese grundsätzlich nicht auf relationale Views, wie den Access Views, angewendet werden können. Auch wenn die java Beans in die grafische Entwicklungsumgebung des Application Developers komfortabel

eingebunden werden können, so bleibt trotzdem der Aufwand für die Erstellung der Beans erhalten, und der in der Anwendung nötige Code der die Bean-Funktionen wie z.B. das Abspeichern aufruft kann aufgrund der Individualität der Bean nicht mit Hilfe von Assistenten generiert werden. Aus diesen Gründen wäre also eine Unterstützung von Views durch die SDOs eine sinnvolle Weiterentwicklung und könnte die Erstellung einer auf Access Views basierenden Webanwendung noch mal deutlich beschleunigen.

5. Ausblick

5.1. Technische Weiterentwicklung

Die im Rahmen dieser Seminararbeit hauptsächlich eingesetzten Softwarekomponenten befinden sich wie Notes/Domino entweder noch in der Beta Phase oder es handelt sich wie bei JSF und deren Unterstützung im WSAD noch um sehr neue Entwicklungen. Beide Bereiche entwickeln sich sehr dynamisch weiter, so dass die hier noch aufgetretenen technischen Probleme, wie die fehlende Unterstützung der SDOs für relationale Views oder das Öffnen verteilter Dokumente in Query Views schon bald gelöst sein werden.

Anders wird dies für bisherige Schwachstellen aussehen bei denen es sich weniger um rein technische Probleme als viel mehr um Designentscheidungen handelt. Beispiele hierfür wären die Umsetzung der Rich-Text-Felder oder die Abfrage der Zugriffsrechte bei der Verwendung von Access-Views. In diesem Bereich müssen Entscheidungen getroffen werden, die immer einen Kompromiss zwischen Standardkonformität und Funktionalität darstellen, so dass hier wahrscheinlich keine eindeutige technische Lösung gefunden wird, sondern sich mit der Zeit Best-Practices herausbilden werden.

Eine vom Zeithorizont deutlich über die beiden zuvor angesprochenen Punkte hinausgehende technische Entwicklung ist die strategische Ausrichtung der Notes/Domino Plattform. Konkret die Frage was der Schritt, eine DB2 Datenbank als optionales Datenbackend für Domino anzubieten, für die weitere Entwicklung von Notes/Domino zu bedeuten hat. Zum einen besteht dadurch die in der Seminararbeit angedeutete Möglichkeit einer Integration der Notes Daten in andere Anwendungen und andersherum, die Nutzung von externen Daten in Notes. Zum anderen erreicht man durch eine Speicherung der Domino Daten in DB2, zumindest bei schon genutzter DB2 Datenbank, eine einheitlich Datenhaltung für alle Unternehmensdaten. Dies erleichtert die Administration und hilft, damit Kosten zu sparen, sorgt durch Auslagerung der Datenhaltung für eine dreischichtige Domino Architektur und ermöglicht es gleichzeitig von der Skalierbarkeit der DB2

Datenbank zu profitieren. Negativ ist allerdings in der bisherigen Implementierung, dass diese Vorteile nur in Kombination mit einer DB2 Datenbank nutzbar sind, und an diesem Umstand wird sich auch bis zur finalen Version 7 nichts ändern. So ist zu erwarten, das hauptsächlich Kunden die schon eine DB2 Datenbank in Verwendung haben von diesen Möglichkeiten Gebrauch machen werden und somit viele Installationen weiterhin nur nsf benutzen werden (vgl. Abbildung 12)

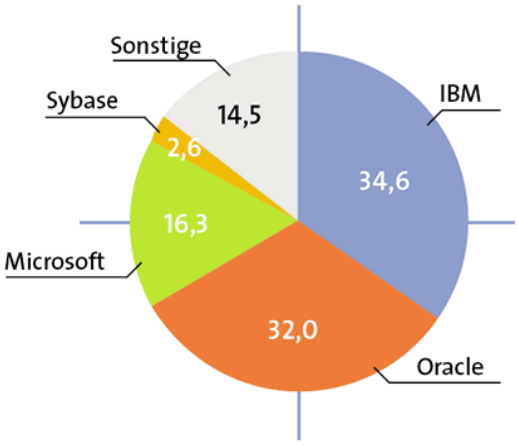


Abbildung 12 - Weltweites Volumen des Datenbankmarkts im Jahr 2001 in % (Quelle [Fritsch 2003])

Daher sollte es das langfristige Ziel sein, möglichst viele RDBMS als Datenbackend zu unterstützen und somit nahezu jedem Kunden einen Umstieg zu ermöglichen. Wenn dieses Ziel erreicht werden kann, wäre langfristig sogar eine Ablösung des nsf Dateiformates denkbar, da der einzige Bereich in dem nsf Dateien dann noch eingesetzt würden der Offline-Betrieb auf den Clients wäre, welcher durch eine fortschreitende Vernetzung allerdings ebenfalls zunehmend bedeutungsloser wird. Der Vorteil für IBM ist hierbei, dass die Pflege und Weiterentwicklung des nsf Formates entfallen könnte, und die ohnehin stattfindende Weiterentwicklung von DB2 direkt positiv auf Domino wirken würde.

5.2. Ziele für Folgeprojekte

Verbleibende Aufgaben für Folgeprojekte können neben der Nutzung und Evaluierung der kommenden Versionen insbesondere Performanceanalysen der vorgestellten Lösungen sein. Untersuchungen in diesem Bereich waren für diese frühe Version der Domino-DB2 Anbindung noch nicht sinnvoll durchführbar, da die Ergebnisse, nach Angaben der Entwickler (vgl. [IBM10 2004]), aufgrund der stetigen Entwicklung deutlich unterhalb der finalen Version liegen sollten, und somit für den praktischen Einsatz irrelevant sein werden. Sollte sich im Bereich der Performance eine Stabilisierung zeigen so sind insbesondere zwei Vergleiche interessant. Zunächst einmal interessiert der direkte Vergleich zwischen den beiden unterschiedlichen Datenspeicherungmöglichkeiten DB2

und nsf. Also eine Aussage darüber, ob und wenn wie viel eine in DB2 vorgehaltene Notes Anwendung schneller arbeitet. Der zweite Aspekt ist der Vergleich zwischen einem Zugriff auf reguläre, also nicht mit Notes in Verbindung stehenden, Tabellen und einem Zugriff auf einen, einen Access View repräsentierenden, DB2 View. Konkret gilt es hierbei also die Frage zu beantworten, wie viel Performance durch die in den Views implementierte Notes Rechteverwaltung verloren geht. Diese Daten können helfen, eine Entscheidung für einen Wechsel des Domino Datenbackends zu fundieren, und den erforderlichen Hardwareaufwand für eine solche Lösung zu bestimmen.

Eine weitere denkbare Folgeaufgabe wäre die Konzeption und Implementierung des Einsatzes in einer gegebenen Infrastruktur. Zu diesem Zweck müsste eine Konstellation gefunden werden, die die in dieser Version erforderlichen technischen Anforderungen erfüllt. Insbesondere also die Kombination aus einer Domino Installation bei gleichzeitiger Nutzung eines auf einer DB2 Datenbank basierenden relationalen Systems. Von besonderem Interesse ist bei einer solchen Aufgabenstellung natürlich die Realisierung auf Seiten der vorhandenen relationalen Daten, da diese im Rahmen dieser Seminararbeit lediglich ohne Bezug auf ein konkretes System gezeigt wurde. Auch die Integration von einer größeren Anzahl relationaler Daten oder die Nutzung von relationalen Daten aus verschiedenen Quellen könnte in diesem Zusammenhang evaluiert werden.

6. Fazit

Die Möglichkeit den Domino Server mit einem optionalen relationalen Datenbackend auszustatten bedeutet eine weitere Annäherung von Notes an relationale Systeme. Hieraus erwachsen vor allem Vorteile im Bereich der Administration und Skalierbarkeit, da nicht mehr zwei Systeme verwaltet werden müssen. Das hier in der Beta Version 2 vorliegende Produkt ist in seiner Funktionalität zwar noch mit zahlreichen Einschränkungen bei speziellen Funktionen behaftet, wie beispielsweise die räumliche Verteilung von Domino- und Datenbankserver, oder die unterstützenden Betriebssysteme, doch funktioniert die grundsätzliche Überführung einer nsf gestützten Notes Anwendung in eine DB2 gestützte für nahezu beliebige Datenbanken selbst in dieser Version schon ohne Probleme. Dies zeigt, dass das gewählte Konzept für die Abbildung nsf auf DB2 flexibel genug ist, und somit in zukünftigen Versionen nur noch wenige Änderungen zu erwarten sind. Die größte Einschränkung, die auch in der finalen Version noch bestehen wird, ist die Festlegung auf DB2 als unterstütztes Datenbanksystem. Dieser Umstand wird der größte Hinderungsgrund für eine schnelle Verbreitung von relationalen Datenbackends für bestehende

Notes/Domino Systeme sein, da die positiven Synergieeffekte nur bei einer vorhanden DB2 Datenbank voll zur Geltung kommen, und ein Wechsel zu DB2 auch keinen entscheidenden Performance Vorteil für Notes Anwendungen bringen wird. Weitere Einflussfaktoren für die Akzeptanz dieser Lösung wird letztendlich die Preisgestaltung für die Kombination Domino/DB2 sein und inwieweit es in der weiteren Entwicklung noch gelingt, die in der bestehenden Beta Version vorhandene Bindung an eine sehr exakt festgelegte DB2 Version aufzuheben.

Die Möglichkeit über die Access Views durch eine relationale Schnittstelle auf Notes Daten zuzugreifen, bedeutet eine signifikante Steigerung der Flexibilität ohne zusätzlichen Platzverbrauch in der jeweiligen Datenbank, sowohl für interne Notes Views (Query Views) als auch für externe Anwendungen. Der entscheidende Nachteil innerhalb von Notes ist, dass auf Dokumente die über mehrere Datenbanken in einem View aggregiert wurden bisher nicht zugegriffen werden kann. Für externe Anwendungen stellt dieser Umstand kein Problem dar, da die logische Zusammenfassung der einzelnen Felder zu einem Dokument generell in der externen Anwendung geschehen muss. Ein, aufgrund der in der Beta Phase stehenden Entwicklung, noch nicht abschließend geklärter Punkt ist die Performance von SQL-Abfragen auf großen Datenbeständen.

Die Entscheidung für oder gegen eine Nutzung des relationalen Datenbackends kann also nur differenziert getroffen werden. Zu empfehlen ist sie grundsätzlich wenn eine DB2 Datenbank schon vorhanden ist, da sich somit alle aus dem DB2 Einsatz ergebenden Vorteile ohne großen Aufwand nutzen lassen. In anderen Fällen sollte jeweils abgewogen werden, ob eine bestimmte Funktion, die durch den DB2-Einsatz ermöglicht wird, den entstehenden Aufwand rechtfertigt, oder ob die gleiche Funktionalität nicht auch, durch Nutzung anderer technischen Mittel, mit weniger Aufwand erreicht werden kann. Hierbei sollten auch die zukünftigen Anforderungen an das System betrachtet werden und die Entscheidung für oder gegen DB2, wenn nicht quantitativ, so zumindest qualitativ, beeinflussen. Ein Beispiel hierfür wäre die Darstellung von Daten aus einer konkreten Notes Datenbank in einer externen Anwendung. Bei isolierter Betrachtung dieser einen Anforderung ist die Nutzung der im Kapitel 3.2 vorgestellten technischen Möglichkeiten oftmals günstiger als die Nutzung von Access Views. Wenn jedoch zusätzlich bekannt ist, dass in Zukunft eine solche Anforderung für weitere Datenbanken besteht, kann eine mit der Nutzung der Access Views einhergehende Umstellung auf ein DB2 Datenbackend lohnend sein, da die hierbei entstehenden Umstellungskosten durch einen geringeren Aufwand bei den folgenden Projekten aufgewogen wird.

7. Literaturverzeichnis

[Benz/Oliver 2003] Benz, B.; Oliver, R. (2003): Lotus Notes and Domino 6 programming bible.

Wiley, Indianapolis 2003.

[Chaffey 1998] Chaffey, D. (1998): Groupware, Workflow and Intranets – Reengineering the Enterprise with Collaborative Software.

Digital Press (Butterworth-Heinemann), Boston 1998.

[Codd 1970] Codd E.F (1970).: A Relational Model of Data for Large Shared Data Banks.

In: Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

Aus: http://www.acm.org/classics/nov95/toc.html

am 12.08.2004

[Date 2000] Date, C. J. (2000): An introduction to database systems.

7. Ausgabe, Addyson Wesley Longman Inc., New York 2000.

[Dier/Lautenbacher 1994] Dier, M., Lautenbacher, S. (1994): Groupware: Technologien für die lernende Organisation.

Computerwoche-Verlag , München 1994.

[Fischer et al 2000] Fischer, J.; Herold, W.; Dangelmaier, W.; Nastansky, L.; Suhl, L. (2000): Bausteine der Wirtschaftsinformatik.

2. Aufl., Erich Schmidt Verlag, Berlin 2000.

[Fritsch 2003] Fritsch, W. (2003): Datenbanktrio im Gleichschritt.

Informationweek.de 6/03, 2003.

Aus:
http://www.informationweek.de/print.php3?/channels/channel12/030648.htm

am 08.07.2004

[Huth/Erdmann/Hahnl 2003] Huth C.; Erdmann I.; Hahnl O. (2003): Recycling von Notes-Java-Objekten und Durchführung der Java Garbage Collection.

Groupware Competence Center, Paderborn 2003.

Aus:

http://gcc.upb.de/www/WI/WI2/wi2_lit.nsf/0/dc7a59ae2202f8f6c1256ced00338691?OpenDocument

am 01.09.2004

[Kremer 1999] Kremer, R. (1999): Replikatives Informationsmanagement in verteilten Groupware Umgebungen.

Shaker Verlag, Aachen 1999.

[Morschheuser 1997] Morschheuser S. (1997): Integriertes Dokumenten- und Worklflow-Management. Deutscher Universitäts Verlag, Wiesbaden 1997.

[Nielsen/Andenæs/ Smith 2000] Nielsen S. P., Andenæs K. M., Smith K. P. (2000): COM Together – with Domino - How to use your BASIC skills to build collaborative applications.

IBM Redbook, New York 2000.

Aus: http://www.redbooks.ibm.com/redbooks/pdfs/sg245670.pdf

am 01.09.2004

[Peron/Nikopoulos/Smith 2003] Peron R.; Nikopoulos S.; Smith K. (2003): Java access to the Domino Objects

Aus: http://www-10.lotus.com/ldd/today.nsf/lookup/Java_access_pt1

am 01.09.2004

[Tamura 2000] Tamura, R. A. (2000): Domino 5 web programming with XML, Java and JavaScript. Que Publishing, Indianapolis 2000.

[Theis 2004] Theis, R. (2004): Domino und DB2: Paradies mit Baustellen.

Notes Magazin 4/04 , 2004.

Aus: http://www.notes-magazin.de/index.php3?page=04-04/grundlagen.html

am 08.07.2004

Ohne Verfasser:

[IBM1 2004] IBM (2004): C API 6.5 User's Guide for Domino and Notes (Dokumentation)

Aus: http://doc.notes.net/tools/c/6.5/api65ug.nsf

am 01.09.2004

[IBM2 2004] IBM (2004): DECS Installation and User Guide (Dokumentation)

Aus: http://doc.notes.net/lei/70/decsdoc7.nsf

am 02.09.2004

[IBM3 2002] IBM (2002): Domino Driver for JDBC 1.5 Guide (Dokumentation)

Aus: http://doc.notes.net/drivers/jdbc/1.5/lddj_help.nsf

am 26.08.2004

[IBM4 2001] IBM (2001): Domino Toolkit for Java/CORBA Guide (Dokumentation)

Aus: http://doc.notes.net/uafiles.nsf/docs/java21/$File/dtjava.exe

am 01.09.2004

[IBM5 2004] IBM (2004): IBM Lotus C++ API Toolkit 3.0 User's Guide (Dokumentation)

Aus: http://doc.notes.net/tools/cplus/3.0/lncpp30.nsf

am 01.09.2004

[IBM6 2004] IBM (2004): LEI Activities and User Guide (Dokumentation)

Aus: http://www-12.lotus.com/ldd/doc/lei/70/leidoc7.nsf

am 02.09.2004

[IBM7 2004] IBM (2004): Lotus Connector and Connectivity Guide (Dokumentation)

Aus: http://doc.notes.net/lei/70/lccon7.nsf

am 02.09.2004

[IBM8 2004] IBM (2004): Notes, Domino, and Domino Designer 7 (Beta 2) Release Notes (English)

Aus: http://doc.notes.net/domino_notes/7.0Beta2/readme_beta.nsf

am 30.09.2004

[IBM9 2003] IBM (2003): NotesSQL 3.02e Reference (Dokumentation)

Aus: http://www-12.lotus.com/ldd/doc/notessql/3.0.2e/notessql.nsf

am 01.09.2004

[IBM10 2004] IBM (2004): Beitrag im Domino 7 beta Forum (Performance)

Aus http://www-10.lotus.com/ldd/beta/nd7pubbeta.nsf/5f27803bba85d8e285256bf10054620d/f53ca831ab68717385256e830065566b?OpenDocument

am 20.10.2004

[Wikipedia 2004] Wikipedia: Component object model.

Aus: http://en.wikipedia.org/wiki/COM

am 01.09.2004

8. Anhang

8.1. Glossar/Abkürzungsverzeichnis:

API: Application Programming Interface

BLOB: Binary Large Object

COM: Component Object Model

CORBA: Common Object Request Broker Architecture

DBMS: Datenbank Management System

HTML: Hypertext Markup Language

HTTP: Hyper Text Transfer Protocol

IDL: Interface Definition Language

IIOP: Internet Inter ORB Protocol

IOR: Interoperable Object Reference

JSF: Java Server Faces

RDBMS: Relationale Datenbank Management System

SDO: Service Data Object

SQL: Structured Query Language

UDF: User Defined Functions

UI: User Interface (Benutzungsoberfläche)

WSAD: Websphere Application Developer

8.2. Downloads der Online Quellen

[Codd 1970] A Relational Model of Data for Large Shared Data Banks.

http://www.acm.org/classics/nov95/toc.html

Codd.mht

[Fritsch 2003] Datenbanktrio im Gleichschritt.

http://www.informationweek.de/print.php3?/channels/channel12/030648.htm

Infoweek.zip

[Huth/Erdmann/Hahnl 2003] Recycling von Notes-Java-Objekten und Durchführung der Java Garbage Collection.

http://gcc.upb.de/www/WI/WI2/wi2_lit.nsf/0/dc7a59ae2202f8f6c1256ced00338691?OpenDocument

GCC K-Pool.mht

[Nielsen/Andenæs/ Smith 2000] COM Together – with Domino - How to use your BASIC skills to build collaborative applications.

http://www.redbooks.ibm.com/redbooks/pdfs/sg245670.pdf

Sg245670.zip

[Peron/Nikopoulos/Smith 2003] Java access to the Domino Objects

http://www-10.lotus.com/ldd/today.nsf/lookup/Java_access_pt1

Java access to the Domino Objects.mht

[Theis 2004] Domino und DB2: Paradies mit Baustellen.

http://www.notes-magazin.de/index.php3?page=04-04/grundlagen.html

Notes Magazin.mht

[IBM1 2004] C API 6.5 User's Guide for Domino and Notes (Dokumentation)

http://doc.notes.net/tools/c/6.5/api65ug.nsf

Api65ug.exe

[IBM2 2004] DECS Installation and User Guide (Dokumentation)

http://doc.notes.net/lei/70/decsdoc7.nsf

Decsdoc.exe

[IBM3 2002] Domino Driver for JDBC 1.5 Guide (Dokumentation)

http://doc.notes.net/drivers/jdbc/1.5/lddj_help.nsf

lddj_help.exe

[IBM4 2001] Domino Toolkit for Java/CORBA Guide (Dokumentation)

http://doc.notes.net/uafiles.nsf/docs/java21/$File/dtjava.exe

Dtjava.exe

[IBM5 2004] IBM Lotus C++ API Toolkit 3.0 User's Guide (Dokumentation)

http://doc.notes.net/tools/cplus/3.0/lncpp30.nsf

Lncpp30.exe

[IBM6 2004] LEI Activities and User Guide (Dokumentation)

http://www-12.lotus.com/ldd/doc/lei/70/leidoc7.nsf

Leidoc.exe

[IBM7 2004] Lotus Connector and Connectivity Guide (Dokumentation)

http://doc.notes.net/lei/70/lccon7.nsf

Lccon.exe

[IBM8 2004] Notes, Domino, and Domino Designer 7 (Beta 2) Release Notes (English)

http://doc.notes.net/domino_notes/7.0Beta2/readme_beta.nsf

readme_beta2.zip

[IBM9 2003] NotesSQL 3.02e Reference (Dokumentation)

http://www-12.lotus.com/ldd/doc/notessql/3.0.2e/notessql.nsf

Notessql.exe

[IBM10 2004] Beitrag im Domino 7 beta Forum (Performance)

http://www-10.lotus.com/ldd/beta/nd7pubbeta.nsf/5f27803bba85d8e285256bf10054620d/f53ca831ab68717385256e830065566b?OpenDocument

IBM Lotus Notes-Domino 7 Public Beta.mht

[Wikipedia 2004] Wikipedia: Component object model.

http://en.wikipedia.org/wiki/COM

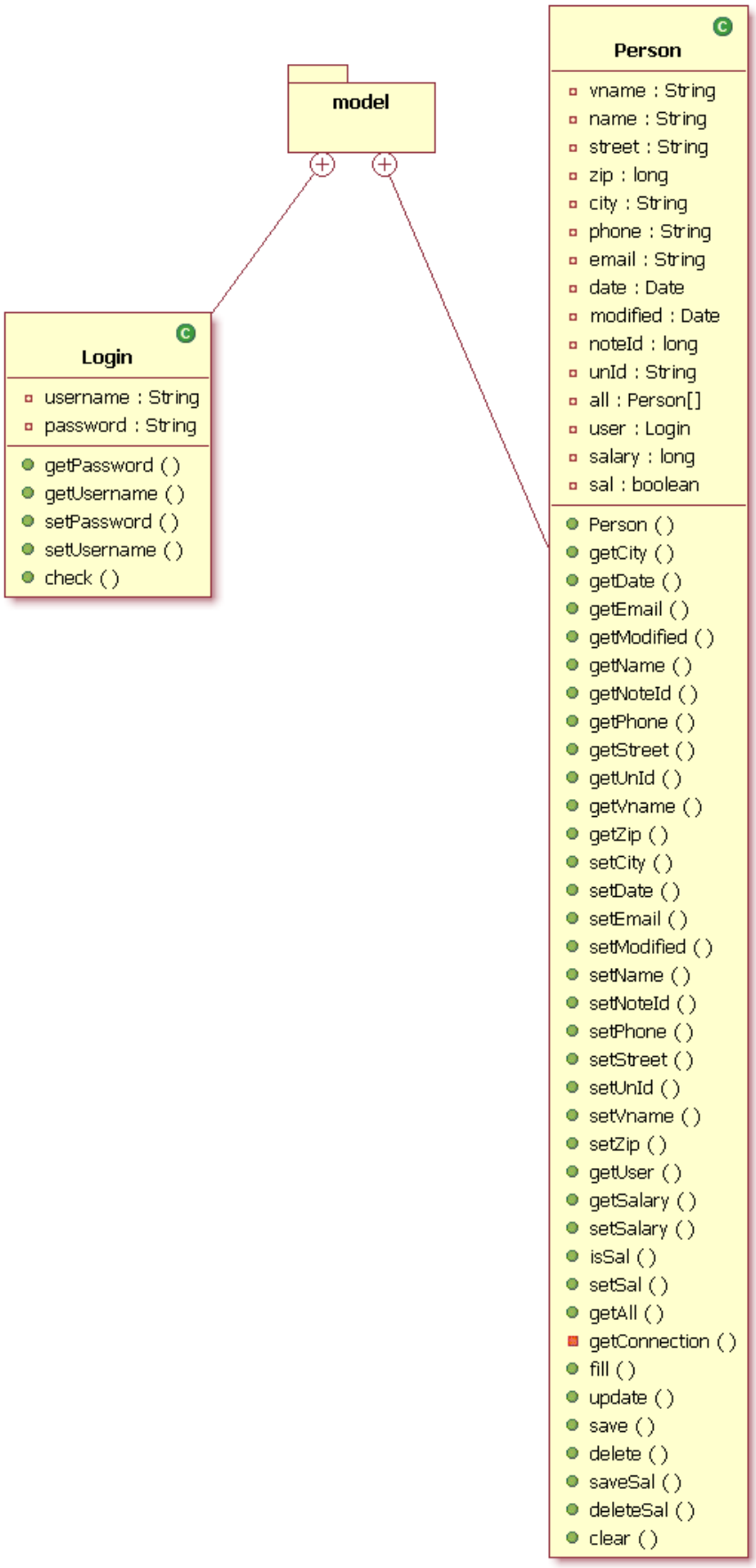
COM - Wikipedia.mht

8.3. Die Beispielanwendung

8.3.1. EAR Datei des WSAD

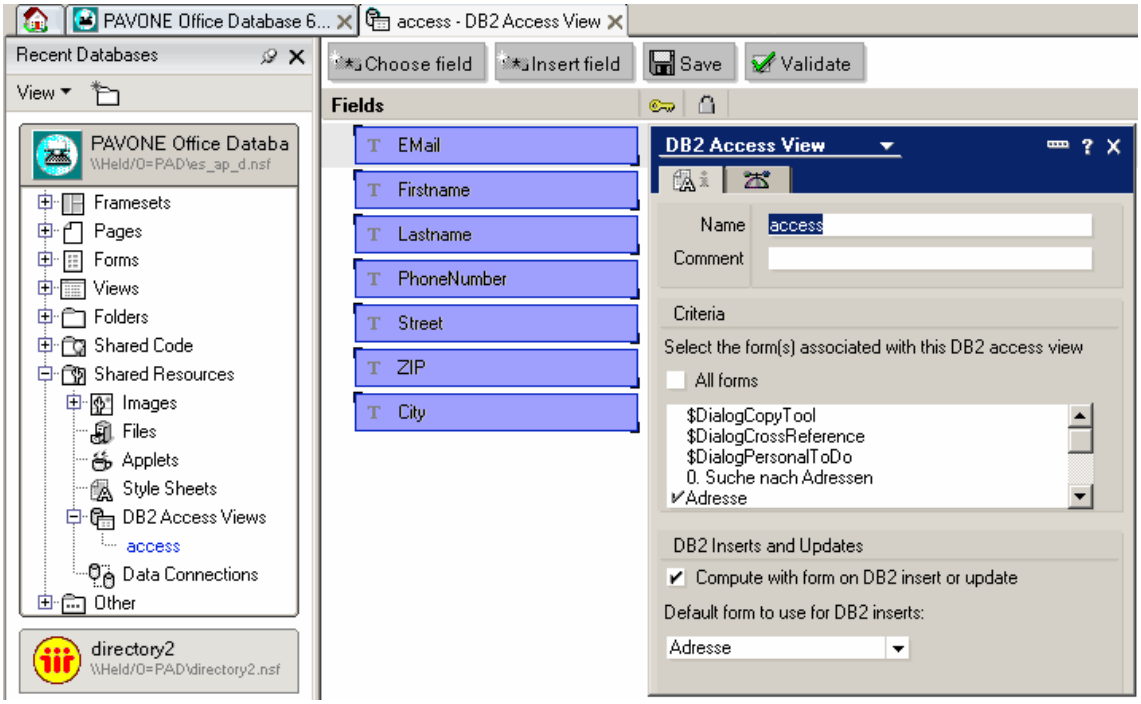
Db2notes.ear

8.3.2. Klassendiagramme der Java Beans

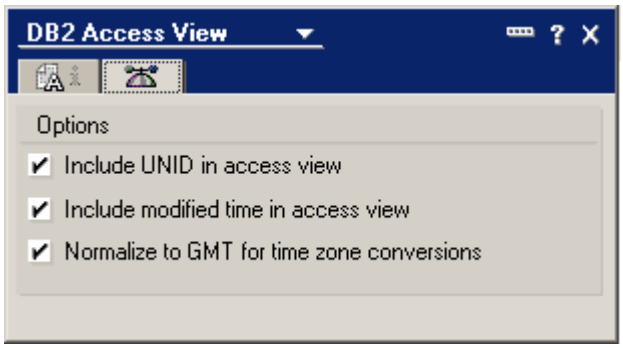


8.3.3. Angaben zum Access View in der Pavone OfficeDatenbank

Neben dem kopieren der Datenbanken auf einen DB2 fähigen Domino Server musste in der Office Datenbank ein Acces View angelegt werden. Die nachfolgenden Screenshots zeigen die enthaltenen Felder und die Konfigurationseinstellungen des verwendeten Access Views.



Felder und Einstellungen des Access Views



Erweiterte Einstellungen des Access Views