Analyse und Bewertung von Konzepten zur Anwendungsintegration eines Hochschulinformationssystems

Studienarbeit im Fach Informatik

vorgelegt von

Peter Ulbrich

geb. am 02.06.1980 in Tettnang

Angefertigt am

Institut für Informatik
Lehrstuhl für Informatik 4 - Verteilte Systeme und Betriebssysteme
Friedrich-Alexander Universität Erlangen-Nürnberg

Betreuer:

Prof. Dr.-Ing. habil. Wolfgang Schröder-Preikschat
Dr.-Ing. Jürgen Kleinöder

Beginn der Arbeit: 01. August 2005
Ende der Arbeit: 28. März 2006

Hiermit versichere ich, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich durch Angabe der Quelle als Entlehnung kenntlich gemacht.

Erlangen, den 28. März 2006

Übersicht

Hochschulen sind komplexe Organisationen. Um die Mitarbeiter zu entlasten und zu unterstützen, werden in zunehmendem Maße Anwendungssysteme eingesetzt. Für die Erfüllung ihrer Aufgaben befinden sich diese Systeme in einem Verbund. Eine reine Kopplung ist dabei keine befriedigende Lösung, da ein Wechsel zwischen den Anwendungssystemen die Arbeit der Nutzer erschwert. Sie benötigen ein umfassendes, einheitliches und hochschulweites Informationssystem. Aus der notwendigen Integration auf technischer und semantischer Ebene entstehen grundsätzliche Probleme wie die Standardisierung der Kommunikation und der Datenschemen.

Das Informationssystem UnivIS ist eines dieser Anwendungssysteme. Es ist nur lose gekoppelt und kann derzeit die Integrität seiner Daten in einem Verbund nicht gewährleisten. Das System weist einige Eigenarten innerhalb seiner Datenhaltung auf, die eine Integration erschweren. Für den Datenaustausch mit anderen Systemen fehlt es zudem an standardisierten Schnittstellen.

Diese Arbeit beschäftigt sich mit den Problemen die bei der Integration von Anwendungssystemen und im speziellen des UnivIS auftreten können. Es werden Lösungsmöglichkeiten für eine technische, wie auch für eine spätere semantische, Integration analysiert und bewertet.

Ziel ist es die Integrationsfähigkeit des UnivIS zu verbessern, seine Kommunikationsmöglichkeiten zu erweitern und den Datenzugriff von außen zu erleichtern.

Abstract

Higher education institutions are complex organisations. In order to support and unburden the employees, application systems are put in place to an increasing degree.For the purpose of performing relevant tasks, these systems are tightly interconnected. A pure coupling is no satisfactory solution, as switching between applications increases the workload of users. There is a need for a comprehensive and uniform information system across higher education academia. From the neccessity of integration on a technical and semantic level, fundamental problems arise, such as the standardisation of communication and data schemes.

The UnivIS information system is an example of such an application system. It is only loosely coupled and cannot guarantee the integrity of its data in an interconnected system at present. The system shows some unique characteristics regarding its data management, which complicates integration. In addition it falls short of standardized interfaces for the data exchange with other systems.

This paper concerns the problems which can arise from the integration of application systems, UnivIS in particular. Possible solutions are supplied for a technical and later also semantic integration, each of which are analyzed and evaluated.

The objective is the improvement of the degree of integration of the UnivIS system, to expand its communication capabilities and to facilitate external data access.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Programmausdrücke

1 Einleitung

1.1 Überblick & Motivation

Hochschulen bestehen aus einer Vielzahl von unterschiedlichen Organisationseinheiten, welche Leistungen für Forschung, Lehre und Verwaltung erbringen. Diese Leistungen umfassen eine große Zahl an Arbeitsabläufen, für deren Erfüllung ein umfangreicher Stab von Mitarbeitern und Einrichtungen benötigt wird.

Mit dem Aufkommen der elektronischen Datenverarbeitung wurden die ersten Anwendungsprogramme für die Unterstützung der Mitarbeiter und zur Optimierung einzelner Arbeitsabläufe eingeführt. Die Beschaffung und Wartung erfolgte in der Regel durch die Abteilung selbst, weshalb diese Systeme hauptsächlich für die eigenen Arbeitsabläufe konzipiert wurden. Eine Kommunikation zwischen den Systemen war nicht vorgesehen und sie entwickelten sich unabhängig voneinander, wie in Abbildung 1.1 (Links) dargestellt. Im Laufe der Zeit sind diese einfachen, dedizierten Anwendungsprogramme zu komplexeren Anwendungssystemen herangewachsen. Sie werden, ihrer Herkunft entsprechend, auch als Abteilungssysteme bezeichnet.

Durch die technologische Entwicklung, speziell durch das Aufkommen des Internet, und die daraus entstehenden Kommunikations- und Zugriffsmöglichkeiten konnten Informationen immer mehr Nutzern zugänglich gemacht werden. Für diesen Zweck wurden weitere Anwendungssysteme entwickelt, welche Informationen für spezielle Nutzergruppen verfügbar machen, so genannte Informationssysteme. Diese sind in der Regel keiner besonderen Abteilung zugeordnet und dienen der Unterstützung von dezentralen Arbeitsabläufen. An der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) wurde das Informationssystem *UnivIS* eingeführt. Es bietet den Mitarbeitern und Studenten über das Internet einen Zugang zu Informationen zu Forschung und Lehre. Das UnivIS verwaltet zum Beispiel das Vorlesungsverzeichnis und dient zur Raumplanung.

Wie ihre Pendants in den Abteilungen, sind Informationssysteme ebenfalls Anwendungssysteme mit einer eigenen Domäne, sie unterscheiden sich im Wesentlichen nur durch die Art der Nutzergruppen. Interne und externe Einflüsse führen zur ihrer Weiterentwicklung.

Interne Einflüsse

Die interne Entwicklung erfolgt kontinuierlich und in kleinen Schritten. Sie entsteht in erster Linie durch die voranschreitende technologische Entwicklung und die steigenden Anforderungen der Anwender. Die Systeme wachsen kontinuierlich und erfassen Informationen immer differenzierter und umfangreicher. Dies führt auch zu einer Annäherung oder sogar einer Überlappung

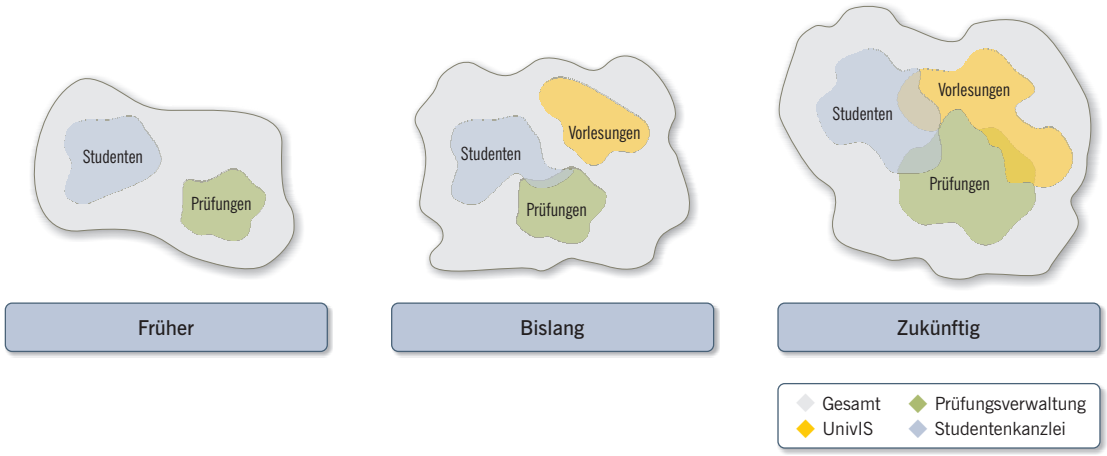


Abbildung 1.1: Entwicklung der Anwendungsdomänen in der Informationsverarbeitung an Hochschulen.

der Anwendungsdomänen und der enthaltenen Informationen wie es in Abbildung 1.1 (Mitte) dargestellt ist.

Überlappungen entstehen durch Arbeitsabläufe die sich über verschiedene Abteilungen beziehungsweise deren Anwendungssystemen erstrecken. Sie bedeuten für den Anwender einen Mehraufwand, da er Daten zwischen den Systemen austauschen oder sogar doppelt erfassen muss.

In der Informationsverarbeitung ergibt sich zudem das Problem der Redundanz, Informationen verteilen sich in unterschiedlichen Ausprägungen über mehrere Systeme. Da die beteiligten Systeme meist eine andere Sicht auf die Daten besitzen, ist ein Austausch schwierig. Auch fehlt es an standardisierten Schnittstellen für die Kommunikation. Um den Anwendern dennoch entgegen zu kommen wurden problemorientierte und systemspezifische Adapter eingeführt, ohne jedoch die grundlegenden Probleme der Informationsverarbeitung zu berücksichtigen.

Externe Einflüsse

Die Entwicklung wird zudem extern, über die politischen und betriebswirtschaftlichen Rahmenbedingungen beeinflusst. Dieser Wandel erfolgt in zeitlichen Abständen und in der Regel ruckartig. Die sich daraus ergebenden Änderungen sind oft ausgeprägter und unmittelbarer, als die evolutionäre Entwicklung im Inneren. Wodurch sich Aufgabenaufgabenstellungen einer Abteilung in kurzer Zeit stark erweitern und verlagern können.

Infolge neuer oder geänderter Arbeitsabläufe können sich Informationen großflächig überlappen. Abbildung 1.1 (Rechts) zeigt die möglichen Auswirkungen durch die damit verbundene Vergrößerung und Verschiebung der Anwendungsdomänen. Die Anwender sind in einem immer größeren Umfang dazu gezwungen ihre Aufgaben mit verschiedenen Anwendungssystemen zu erfüllen. Können diese Systeme nicht lückenlos miteinander kommunizieren führt dies zu Medienbrüchen. Es entsteht der bereits erwähnte Mehraufwand für den Wechsel zwischen den Systemen und die multiple Erfassung.

Die Probleme der Informationsverarbeitung steigen durch die Redundanz und Inkonsistenz der

Daten mit zunehmender Vernetzung überproportional an. Ein Ausgleich dieser Defizite durch systemspezifische Lösungen wird dadurch immer schwieriger und aufwendiger.

Um die Anwender auch weiterhin in ihrer Tätigkeit unterstützen zu können, müssen die Systeme in Zukunft nicht nur eine optimale Kommunikation zu ihren eigenen Nutzern, sondern auch untereinander bieten. Dies erfordert ein Zusammenwachsen der bisherigen Anwendungssysteme in ein integriertes, hochschulweites Informationssystem für die Bereitstellung der Informationen in einheitlicher Form.

1.2 Aufbau und Ziele dieser Arbeit

Generell beschäftigt sich diese Arbeit mit der Frage, wie sich das System *UnivIS* in ein solches, hochschulweites Informationssystem einbinden lässt. Der Aufbau gliedert sich hierzu in drei Schwerpunkte:

Einführung in den Themenbereich Integration: Ziel dieses Schwerpunktes ist es, die allgemeinen Probleme der Integration eines hochschulweiten Informationssystems aufzuzeigen. Zu diesem Zweck wird in Kapitel 2 ein allgemeiner Überblick über Themen *Informationssysteme* und deren *Integration* gegeben. Sowie die grundlegenden Mittel und Konzepte der Integration erläutert und am Beispiel des Universitätsklinikums Erlangen veranschaulicht. Schließlich wird der aktuelle Stand in der Hochschulverwaltung betrachtet.

Betrachtung des UnivIS im Kontext der Integration: Im Weiteren wird analysiert inwieweit das System UnivIS auf die Problematik der Integration vorbereitet ist, mit dem Ziel vorhandene Schwachpunkte und Defizite aufzudecken. Dazu wird in Kapitel 3 zunächst der grundlegende Aufbau des Systems und die künftigen Anforderungen beschrieben. In Kapitel 4 wird untersucht wie es um die Integrationsfähigkeiten des UnivIS steht und ob die gestellten Anforderungen mit dem aktuellen Stand erfüllt werden können.

Möglichkeiten das UnivIS zu integrieren: Die dritte Zielsetzung ergibt sich mit der Forderung nach einer verbesserten Integrationsfähigkeit. In Kapitel 5 werden Möglichkeiten gezeigt und bewertet, den Datenzugriff auf das UnivIS und die Kommunikation mit dem Verbund zu verbessern. Abschließend werden in Kapitel 6 Vorschläge für die Umsetzung einer Datenreplikation, als eine geeignete Lösung, gegeben.

2 Informationssysteme an Hochschulen - Grundlagen, Überblick, Stand der Technik

In diesem Kapitel werden zunächst einige technische Grundlagen von Datenbanksystemen eingeführt. Sie finden in praktisch allen Abteilungssystemen Anwendung und sind deshalb in der späteren Analyse wichtig.

Bereits in der Einleitung wurde vom Begriff der *Informationssysteme* Gebrauch gemacht. Dieser wird in Kapitel 2.2 detailliert erläutert und bildet die Überleitung zur Anwendungsintegration. Darauf folgend werden gängige Methoden und Konzepte der Integration beschrieben und am Beispiel der Entwicklungen im Universitätsklinikum Erlangen veranschaulicht.

Anschließend wird die Situation der IT-Systeme in der Hochschulverwaltung erörtert.

2.1 Datenbanksysteme

Ein *Datenbanksystem* (DBS) ist ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen, die von mehreren Anwendungsprogrammen benutzt werden. (aus [ECS93])

2.1.1 Aufbau

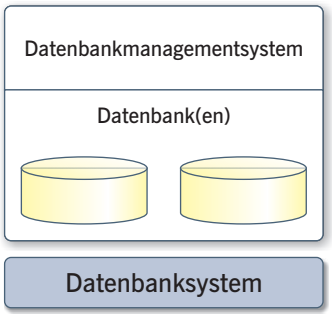
Ein Datenbanksystem besteht aus der Datenbank (DB), in der die Daten abgelegt werden und dem Datenbankmanagementsystem (DBMS). Welches die Daten entsprechend den vorgegebenen Beschreibungen abspeichern, auffinden oder weitere Operationen mit den Daten durchführen kann (vgl. Abb. 2.1.1) [ECS93].

In Abbildung 2.1.2 ist der Aufbau aus Sicht des Systementwurfs als vereinfachtes Schichtenmodell dargestellt. Durch die einzelnen Schichten werden die wesentlichen Abstraktionsschritte von der Externspeicherebene bis zur Benutzerschnittstelle charakterisiert, die das DBS dynamisch vorzunehmen hat (nach [Hä01] Kapitel 1.3).

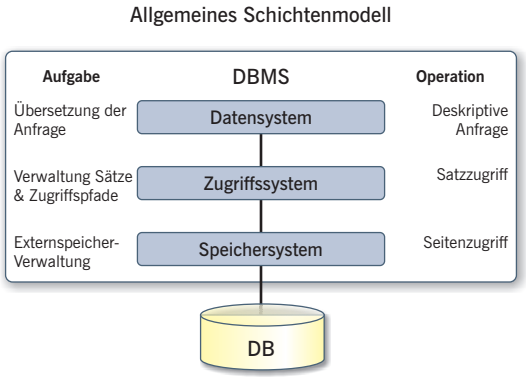
2.1.2 Eigenschaften

Ein Datenbanksystem muss folgende Vorraussetzungen erfüllen (nach [Hä01] Kapitel 1.2):

- Isolation von Anwendungen und Daten (Anwendungs- und Datenunabhängigkeit).



2.1.1: Aufbau



2.1.2: Allgemeines Schichtenmodell (nach [Hä01])

Abbildung 2.1: Aufbau eines Datenbanksystems

- Unterstützung verschiedener Benutzersichten.
- Das DBS umfasst neben der Datenhaltung auch die Daten- und Schemabeschreibung.
- Abstraktion der Speicherstrukturen (Datenabstraktion).
- Fehlerbehandlung
- Transaktionskonzept, Operationen nach ACID Prinzip: Atomicity (Ausführung ganz, oder gar nicht), Consistency (Die Transaktion endet in einem konsistenten Zustand), Isolation (Transaktionen beeinflussen sich nicht gegenseitig), Durability (Das Ergebnis ist dauerhaft).

2.1.3 Datenmodelle

Unter dem Datenmodell versteht man eine Menge von Begriffen mit denen das Schema (oder Struktur) einer Datenbank beschrieben werden kann:

Satzschnittstelle: Sätze (Datenobjekte) in verschiedenen Satztypen stehen in keiner Beziehung zueinander. Der Zugriff erfolgt satz- und zugriffspfadorientiert (GET #, STORE). Die Verknüpfung der Daten wird durch Lesen des Schlüssels aus einer Datei und Öffnen einer anderen Datei erreicht. Die Datenunabhängigkeit ist in diesem Modell aufgrund der expliziten Nutzung von Indexstrukturen nicht zu gewährleisten.

Hierarchisch: Sätze stehen in einer Eltern-Kind Beziehung. Der Zugriff erfolgt durch logische Zugriffspfade (GET PARENT, GET NEXT). Die Implementierung der Zugriffspfade bleibt verborgen. Auch das Codasyl- und das Netzwerkmodell verwenden logische Zugriffspfade.

Relational: Das relationale Datenmodell ist das bekannteste und weit verbreitet. Im Gegensatz zu den vorangegangenen Modellen besitzt es keine expliziten Zugriffspfade, sondern ist mengenorientiert. Die Beziehung zwischen den Datenobjekten (Relationen) geschieht über

Schlüsselattribute. Zugriffspfade und Reihenfolge sind nach außen nicht mehr sichtbar. Auf Datenobjekte wird durch eine Anfrage (SELECT, JOIN) zugegriffen. Die zugrunde liegende Relationenalgebra erlaubt eine effiziente Auswertung und Optimierung der Anfragen vor Erstellung des Zugriffspfades durch das System.

Objektorientiert: Verschiedene Implementierungen. Standardisierungsbemühungen durch die Object Management Group[1] (OMG): Datenobjekte sind Objekte mit eindeutigem Identifier, einem Zustand (Wertemenge), einer Menge an Methode und Beziehungen. Der Zugriff erfolgt durch Referenzen oder über eine mengenorientierte Anfragesprache. Objektorientierte Modelle eignen sich besonders gut für komplexe Gegenstände, wie zum Beispiel Landkarten oder CAD-Objekte[2].

Objektrelational: Erweiterung des relationalen Modells durch erweiterbares Typensystem und Unterstützung komplexer Datentypen als Attribute.

XML: Für XML (eXtended Markup Language) stehen Zugriffspfade über XPath[3] und mit XQuery[4] auch eine Anfragesprache zur Verfügung. Datenmodelle mit XML eignen sich für die Erfassung semi-strukturierter Daten. Die meisten großen (objekt-)relationalen DBS beherrschen heute den Umgang mit semi-strukturierten Daten, es gibt aber auch native XML DBS (siehe [Mey05]).

2.1.4 Datenmodellierung

Der Datenbankentwurf ist durch die Phasen in der Abbildung 2.2 gekennzeichnet. Der erste Schritt umfasst die genaue Untersuchung aller Anforderungen, dabei werden alle Dinge der realen Welt und die Beziehungen zwischen ihnen beschrieben.

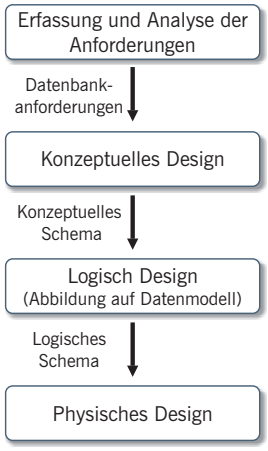


Abbildung 2.2: Phasen eines Datenbankentwurfs

Im Rahmen des konzeptuellen Designs wird häufig das *Entity-Relationship-Modell* (ERM) eingesetzt um Teile der realen Welt zu beschreiben. ER-Diagramme bestehen im Wesentlichen aus den *Entities* (Dinge in der realen Welt) und *Relationships* (Beziehungen) aus der Anforderungsanalyse.

Ziel ist die Entitäten so zu wählen, dass keine funktionalen Abhängigkeiten zwischen ihnen bestehen. Sind zum Beispiel *Student* und *Mitarbeiter* zu erfassen, sollten die Attribute *Name* und *Vorname* nicht in beiden Entitäten gespeichert werden. Dafür ist eine neue Entität *Person* zu wählen, eine Person kann zugleich *Student* und *Mitarbeiter* sein. Diesen Vorgang nennt man *Normalisierung*. [EN04]

2.1.5 Datenintegrität

Die im Datenbankentwurf modellierten Beziehungen werden durch das Datenbanksystem garantiert. Änderungsoperationen, welche die Integrität der Daten verletzten würden, werden zurückgewiesen. Die technische Umsetzung erfolgt in der Regel durch Primär- und Fremdschlüssel. Primärschlüssel identifizieren einen Datensatz innerhalb einer Relation eindeutig. Fremdschlüssel

[1] `http://www.omg.org/`
[2] Computer Aided Design
[3] XML Path Language - `http://www.w3.org/TR/xpath`
[4] XML Query - `http://www.w3.org/XML/Query/`

verweisen auf einen anderen Primärschlüssel und bilden so eine Beziehung nach. Diese *referenzielle Integrität* wird direkt durch das physikalische Datenschema festgelegt und bedarf keiner weiteren, sprachlichen Beschreibung.

Zusätzlich bieten viele Datenbanksysteme weitere Konstrukte zur Einhaltung der Datenintegrität an. Zum Beispiel *Check-Bedingungen* zur Prüfung von Wertebereichen oder *Trigger* als programmiersprachliche Erweiterung für komplexere Integritätsbedingungen. [EN04]

2.1.6 Einbettung von Datenbanksprachen in Wirtssysteme

Der Zugriff auf Datenbanken ist heute in allen gängigen Programmiersprachen möglich. Die bekannteste Methode ist die Verwendung standardisierter Schnittstellen, so genannten *Call level interfaces* (CLI). Implementierungen sind beispielsweise ODBC (Open Database Connectivity), JDBC (Java Database Connectivity) oder Microsoft ADO (ActiveX Data Objects). [EN04]

Für eine ausführlichere Beschreibung der gesamten Thematik sei auf das Werk von Elmasri und Navathe, *Fundamentals of database systems* ([EN04]) verwiesen.

2.2 Informationssysteme

Die Bezeichnung *Informationssystem* ist weit verbreitet. Es existieren viele verschiedene Interpretationen und Definitionen, kurz gesagt handelt es sich um einen sehr dehnbaren Begriff.

Definition

Seibold nähert sich dem Begriff in [Sei03] über eine Reihe von formalen Definitionsansätzen:

Information: (lat. Informatio: das Versehen von etwas mit einer Form) Die Kenntnis über bestimmte Sachverhalte oder Vorgänge.

System: (griech.: Stück aus mehreren Teilen) Eine Menge von Personen, Dingen und/oder Vorgängen und der ganzheitliche Zusammenhang zwischen diesen.

Sozio-informationstechnisches System: Ein System, in dem Menschen und Maschinen nach festgelegten Regeln bestimmte Aufgaben erfüllen sollen. Systeme können sich in Teilsysteme untergliedern.

Informationssystem: (sozio-technisches) Teilsystem eines Unternehmens [einer Organisation, Klinik, Hochschule], das aus den informationsverarbeitenden Aktivitäten und den an ihnen beteiligten menschlichen und maschinellen Handlungsträgern in ihrer informationsverarbeitenden Rolle besteht.

Krcmar liefert folgende, formale Definition:

> "Informationssysteme sind soziotechnische Systeme, die aus Teilsystemen für optimale Bereitstellung von Information und technischer Kommunikation dienen." [Krc00]

Pragmatische Definition

Ein Informationssystem stellt einer Zielgruppe die von ihr benötigten Informationen in einer geeigneten Sicht dar. Es erfüllt damit einen bestimmten Zweck und umfasst alle Informationsquellen (technische und nichttechnische Systeme, Datenbanken, andere (Teil-) Informationssysteme, ...) die hierfür nötig sind. Die Zweckbindung des Systems wird meist durch einen Präfix ausgedrückt, beispielsweise *Betriebswirtschaftliches Informationssystem* oder *Krankenhausinformationssystem*.

2.3 Anwendungsintegration

Die Integration dient der Verknüpfung von verschiedenen Anwendungen. Warum sollte man Integration betreiben? Um diese Frage beantworten zu können, wird im Folgenden die Entwicklung der IT-Struktur in größeren Unternehmen betrachtet und die auftretenden Probleme dargestellt.

Historische Entwicklung

Die rechnergestützte Verarbeitung von Informationen begann mit der Entwicklung und Einführung von problemorientierten Anwendungen zur Unterstützung der Arbeitsabläufe. Eine Zusammenarbeit zwischen den Anwendungen war nicht vorgesehen.

Die ersten Schritte zur Einführung elektronischer Informationssysteme wurden mit den daraus entstehenden, dedizierten *Abteilungssystemen* gemacht. Diese unterstützten aber nur die Prozesse innerhalb der eigenen Abteilung.

Ein Datenaustausch zwischen diesen isolierten Systemen fand aber nur in den wenigsten Fällen über standardisierte Schnittstellen statt. Eine inkonsistente und redundante Datenhaltung war die Folge.

Problematik

Warum nun Integration? Redundanzen zwingen dem Nutzer einen Mehraufwand bei der Erfassung und Verknüpfung von Informationen auf. Er muss Daten mehrfach eingeben, auf Korrektheit und Vollständigkeit überprüfen. Sind mehrere Systeme an der Verarbeitung eines Prozesses beteiligt und können diese nicht miteinander kommunizieren, entstehen Medienbrüche. Diese wiederum müssen vom Nutzer mühsam überbrückt werden. Die Informationsprozesse werden dadurch erschwert, verlangsamt oder in ihrer Qualität gemindert.

Zudem steigt die Zahl der Kommunikationsverbindungen durch reine Kopplung mit der Anzahl der angebundenen Systeme und führt zur $n \cdot (n-1)$ Problematik. Im schlimmsten Fall kommen bei n Systemen $n \cdot (n-1)$ neue Kommunikationsverbindungen hinzu. Damit steigt der Aufwand für die Administration des Gesamtsystems unverhältnismäßig an. Integration dient also der Reduzierung und Standardisierung der Schnittstellen und fördert damit die Leistungsfähigkeit des Gesamtsystems. [Pro01]

Integrationskonzepte

Man unterscheidet folgende Konzepte für die Integration der IT-Struktur [Pro01]:

Monolithische Konzepte bieten möglichst viele Anwendungssysteme aus einer Hand. Sie zeichnen sich durch ein hohes Maß an Konsistenz aus. Allerdings setzen diese Konzepte detaillierte Kenntnisse der Entwickler voraus. Eine mangelnde Spezialisierung stellt der einzelnen Abteilung oft keine optimale Lösung zur Verfügung. Ist ein monolithisches Informationssystem erst etabliert, ist es schwierig dieses System zu wechseln oder zu erneuern.

Heterogen verteilte Konzepte versuchen für jeden Bereich ein optimales Anwendungssystem zu finden (so genannte Best-of-Breed Lösungen) oder zu entwickeln. Der Datenaustausch muss dann über Kommunikations- oder Informationsserver erfolgen. Die gute Unterstützung der einzelnen Abteilungen wird hier durch einen hohen Kommunikationsaufwand erkauft und setzt entsprechende Standardisierung voraus. Um- und Ausbau des Gesamtsystems sind wesentlich einfacher als bei monolithischen Konzepten.

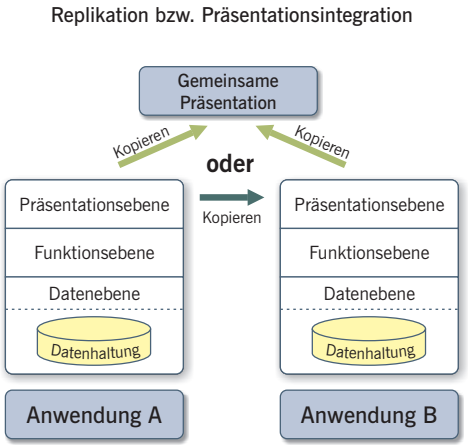
Komponentenbasierte Konzepte liefern die Basisfunktionalitäten (wie Benutzeridentifikation, Personalmanagement, Terminplanung, usw.) als wiederverwendbare Komponenten, welche ihre Funktionalität über eine standardisierte Programmierschnittstelle (API) bereitstellen. Das Zusammenstellen derartiger Komponenten zu einem Informationssystem kann als Integration bezeichnet werden.

Welches Konzept im Einzelfall zu wählen ist, hängt stark von der Problemstellung ab. Für relativ überschaubare und kleine Probleme eignen sich monolithische Konzepte. Ab einer gewissen Komplexität ist eine optimale Zweckerfüllung von einem System alleine aber kaum zu gewährleisten. Deshalb kommen verteilte Konzepte zum Einsatz, der Markt bietet inzwischen ein breites Spektrum an Kommunikationslösungen an. Mischformen sind als Mittelweg ebenfalls möglich. Der Kompliziertheit wird durch Reduktion, der Komplexheit durch Beibehaltung von Systemen begegnet. “So viele verschiedene Systeme wie nötig, so wenige wie möglich.” [Pro01]

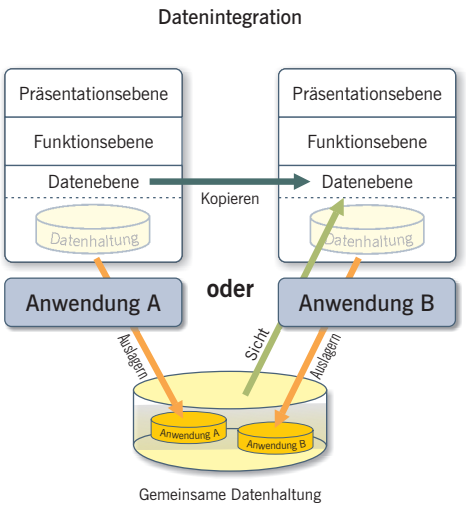
2.3.1 Integrationsformen

Zusätzlich zu den Konzepten unterscheidet man die Integrationsformen. Diese ergeben sich aus Art und Grad der Integration. Es ist möglich bei einem Anwendungssystem mehrere Integrationsformen gleichzeitig zu verwenden, auch müssen die verwendeten Formen nicht bei allen Systemen identisch sein.

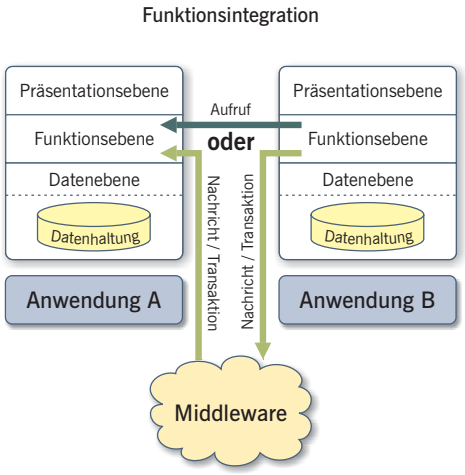
Aus dem *Verbinden* der Systeme folgt nicht automatisch auch ein gemeinsames *Verständnis* der Daten. Deshalb muss zusätzlich noch zwischen *technischer* und *semantischer Integration* unterschieden werden (siehe Abbildungen 2.3 bzw. 2.4). Durch technische Integration werden unterschiedliche Technologien überbrückt, sie ist relativ leicht zu erreichen. Die semantische Integration ist ungleich schwerer, da in allen beteiligten Systemen eine Übereinstimmung im Bezug auf die Bedeutung von Daten und Funktionalitäten herrschen muss. [PKSL06]



2.3.1: Replikation bzw. Präsentationsintegration



2.3.2: Datenintegration



2.3.3: Funktionsintegration

Abbildung 2.3: Technische Integrationsformen nach Grad der Integration

Technische Integration

Es lassen sich drei technische Integrationsformen unterscheiden, sie repräsentieren einen gewissen Integrationsgrad und inkludieren sich aufsteigend.

Durch *Präsentationsintegration* (auch Screen-Scraping, vgl. Abb. 2.3.1) werden die Nutzerschnittstellen der beteiligten Anwendungen einheitlich visualisiert. Dies kann im Kontext einer bereits bestehenden Anwendung sein oder durch eine Zusatzschicht (z.B. Webserver, Java-Applet). Bei diesem Vorgehen handelt es sich um eine reine Replikation der Daten. Ein Zugriff auf die Funktions- oder Datenebene ist nicht möglich.

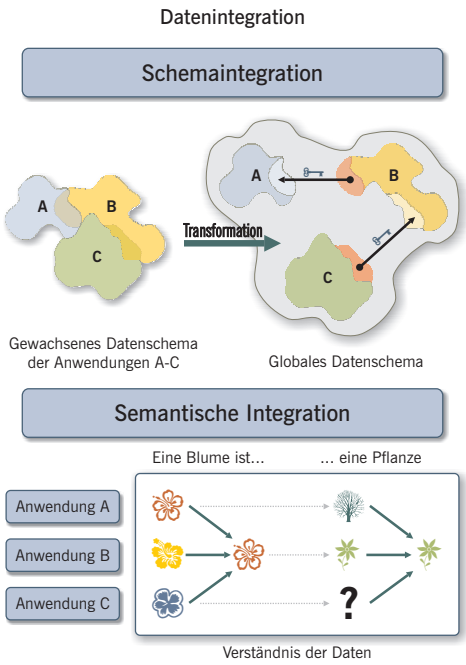
Bei der *Datenintegration* (vgl. Abb. 2.3.2) erfolgt der Austausch der Datenbestände direkt zwischen den Datenebenen der Anwendungen. Oft wird in diesem Zuge die Datenhaltung der Anwendungen standardisiert in einer gemeinsamen Datenbank (Unternehmensspeicher) gebündelt oder durch eine Database Access Middleware verbunden. Die Replikation (Data briding) kann dann direkt abgewickelt, oder der Anwendung als Sicht (Data sharing) vorgegaukelt werden. Eine Verwendung der Anwendungslogik ist aber weiterhin nicht möglich. Greift eine Anwendung auf fremde Daten zurück, muss sie Zustandsänderungen selbst durchführen. Unter Umständen kann dies zu Inkonsistenzen führen, wenn entsprechende Bedingungen ausschließlich in der Anwendungslogik stecken. Ein weiteres Problem ist die Abbildung (Data Mapping) der Datenstrukturen, falls diese zwischen den Anwendungen nicht disjunkt sind.

Die *Funktionsintegration* (vgl. Abb. 2.3.3) ermöglicht es schließlich direkt auf Funktionalitäten zugreifen zu können. Im einfachsten Fall sind dies Methoden und Funktionen einer Anwendung, der Zugriff erfolgt dann beispielsweise über Fernaufrufe (Remote Procedure Calls). Problematisch ist dabei die Verstrickung der Anwendungen im Quellcode und die damit verbundene Komplexität. Eine deutliche Verbesserung dieser Problematik bieten Middleware-Lösungen (Message Oriented Middleware, Kommunikationsserver). Durch sie werden die konkreten Prozeduraufrufe in Nachrichten, Transaktionen, oder ähnliches verpackt, welche eine Funktionalität beschreiben. Die Funktionsintegration entfaltet ihr Potential deshalb erst, wenn ein gewisses Maß an semantischer Integration erreicht wurde.

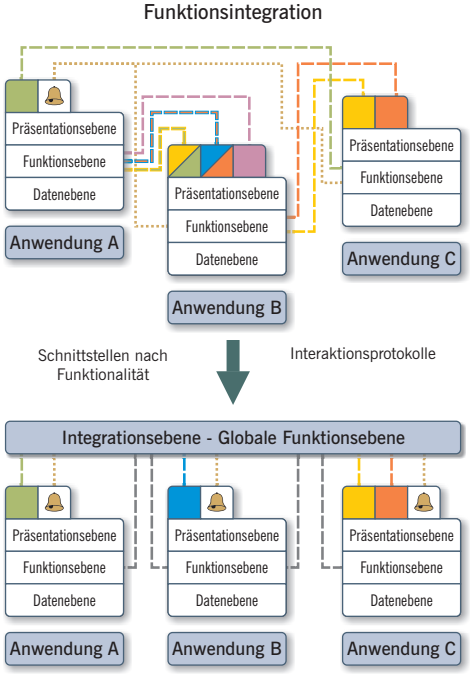
Semantische Integration

Durch *semantische Integration* wird schließlich ein gemeinsames Verständnis geschaffen. Wie bei der technischen Integration, können die Bemühungen auf unterschiedlichen Ebenen erfolgen.

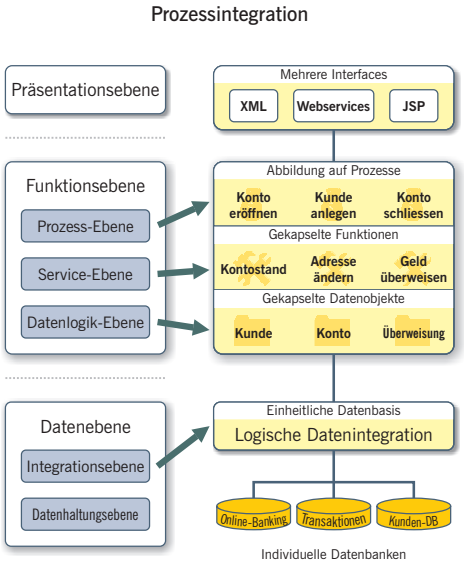
Die *Datenintegration* (vgl. Abb. 2.4.1) findet in zwei Bereichen statt, im Datenschema und in der Ontologie. Zuerst wird das Datenschema der einzelnen Anwendungssysteme zu einem globalen Datenschema transformiert. Dadurch werden Redundanzen vermieden und die Daten erhalten eine kontrollierte Terminologie. Die Schemaintegration kann virtuell (Schema Mapping) oder materialisiert (Schema Integration) erfolgen. Bei einem virtuellen Schema bleiben die Datenquellen unberührt und der Zugriff erfolgt ausschließlich über Sichten, es wird deshalb oft auch als föderierte Schemaintegration bezeichnet. Anderenfalls werden die Daten in ein globales Schema verschoben, meist wird dieses Vorgehen bei vorhandenen Unternehmensspeichern



2.4.1: Datenintegration



2.4.2: Funktionsintegration



2.4.3: Prozessintegration [Amb06]

Abbildung 2.4: Semantische Integrationsformen

angewendet. Im nächsten Schritt wird eine gemeinsame Ontologie geschaffen, damit die Interpretation der Daten einheitlich ist. Im Allgemeinen werden dazu Vokabulare eingesetzt, die eine allgemein akzeptierte Begriffsdefinition der gewünschten Domäne enthalten. Zum Beispiel der Duden für die Domäne der deutschen Rechtschreibung.

Ein globales Verständnis ist auch bei der *Funktionsintegration* (vgl. Abb. 2.4.2) das Ziel, jedoch auf der Ebene der Funktionen. Zunächst werden die gewünschten Funktionalitäten definiert und die verfügbaren Funktionen danach gegliedert. Jede Funktionalität steht dabei nur an einer Schnittstelle zur Verfügung. Zusammengesetzte Funktionalitäten können durch Trigger an alle beteiligten Anwendungen weitergereicht werden. Diese Beziehungen zwischen den Anwendungen werden in Interaktionsprotokollen spezifiziert. Das Verbinden der Anwendungen und die Ausführung der Interaktionsprotokolle übernimmt in der Regel die eingesetzte Middleware. Dabei unterscheidet man zwei Formen, die globale und die föderierte Funktionsintegration. Auch hier bedeutet die föderierte Funktionsintegration, dass die individuellen Funktionsebenen bestehen bleiben und die Integrationsebene nur eine Sicht bietet. Dies ist bei der globalen Strategie nicht der Fall.

Vor allem in der betriebswirtschaftlichen Steuerung der Arbeitsabläufe wird zusätzlich eine weitere Form der Integration eingesetzt, die *Prozessintegration* (vgl. Abb. 2.4.3). Hierbei wird die Integration an und mit der Prozesskette des Unternehmens vollzogen. Diese Form wird im weiteren Sinne oft auch als *Enterprise Application Integration* bezeichnet und baut auf den zuvor genannten Formen auf. Es sei hierzu auf die Literatur verwiesen (vgl. [Amb06], [Wik06b], [Eic02]).

2.3.2 Webservices

Im Internet ist es heute möglich die unterschiedlichsten Informationen abzurufen. Von Aktienkursen über aktuelle Nachrichten bis zum Wetterbericht stehen dem menschlichen Nutzer viele Dienste zur Verfügung. Im Prinzip bieten *Webservices* Informationsdienste in einer elektronisch nutzbaren Form - eine Art WWW für Rechnersysteme. [Wik06a]

Webservices kommen aus dem Bereich des eCommerce und der Business-to-Business Anwendungen[5]. Dort versucht man individuell vereinbarte Schnittstellen und Nachrichtenformate zu ersetzen oder zu automatisieren. [Bet]

> "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." [BHM+04]

Eine einheitliche Terminologie existiert bisher jedoch nicht, in den Standardisierungsgremien wie dem World Wide Web Consortium (W3C)[6] werden Webservices aber übereinstimmend wie

[5] Anwendungsschnittstellen zwischen zwei Geschäftspartnern, z.B. zwischen dem Hersteller und seinen Lieferanten

[6] `http://www.w3c.org/`

folgt charakterisiert:

Programmierbar: Webservices sind über programmierbare Schnittstellen erreichbar, sind in erster Linie zur Anwendungskommunikation geschaffen und haben keine graphische Benutzeroberfläche.

Selbstbeschreibend: Ein Webservice wird begleitet von Metadaten [7], die während der Laufzeit von weiteren Webservices ausgewertet werden können.

Kapselung: Ein Web-Service ist eine unabhängige, in sich abgeschlossene bzw. gekapselte Anwendung, die eine genau definierte Aufgabe erfüllt.

Lose gekoppelt: Die Kommunikation erfolgt über Nachrichtenaustausch. Webservice-Konsumenten und -Anbietern bleiben Implementierungsdetails verborgen.

Ortstransparenz: Webservices sind ortsunabhängig und können jederzeit und von jedem Ort aus aktiviert werden.

Protokolltransparenz: Ein Webservice basiert auf der Internet-Protokollsuite. Operationen und Nachrichten können mehrere Protokolle unterstützen.

Komposition: Webservices können aus mehreren (Basis-) Webservices zusammengestellt, zusammengesetzte Webservices bis zu ihren Basiswebservices zerlegt werden.

Ein *Anbieter* beschreibt seine Dienstleistung und die Schnittstelle mit Hilfe der Web Service Description Language (WSDL) und registriert den Webservice in einem Verzeichnis.

Der *Verzeichnisdienst*, auch Universal Description, Discovery and Integration (UDDI), bietet Gelbe Seiten für die Dienstfindung an. Diese bestehen aus White-Pages mit Anbieterdaten, Yellow-Pages mit Dienstleistungen und Green-Pages mit den technischen Details.

Der *Konsument* erhält die gewünschten Informationen über SOAP (Simple Object Access Protocol).

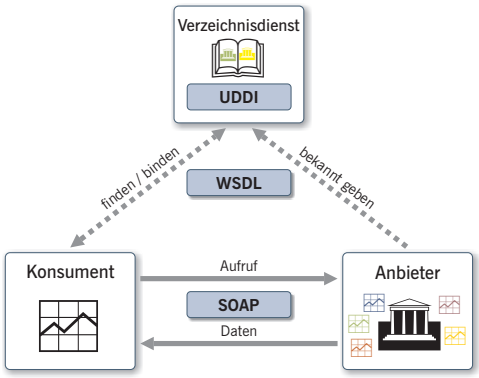


Abbildung 2.5: Webservice Architektur

Es gibt in einem Webservice damit drei unterschiedliche Rollen: Anbieter, Verzeichnis und Konsument (siehe Abbildung 2.5).

Webservices eignen sich für die lose Kopplung von Systemen und sind damit ein guter Baustein im Bereich der Enterprise Application Integration und Business-to-Business Integration. Sie integrieren aber nicht per se. Entsprechende Integrationskonzepte und Maßnahmen sind trotzdem notwendig.

[7] Metadaten bezeichnen im Allgemeinen Daten, die Informationen über andere Daten enthalten.

2.4 Integration am Beispiel eines Universitätsklinikums

Das Universitätsklinikum ist ein anschauliches Beispiel für die in Kapitel 2.3 genannten Probleme und Lösungen.

Ein Klinikum besitzt eine heterogen gewachsene IT-Landschaft mit vielen kleinen Anwendungssystemen und einigen größeren Abteilungs- bzw. Teilinformationssystemen (Radiologieinformationssystem, Laborinformationssystem, ...). In ihrer Gesamtheit bilden sie das *Klinikinformationssystem* (KIS).

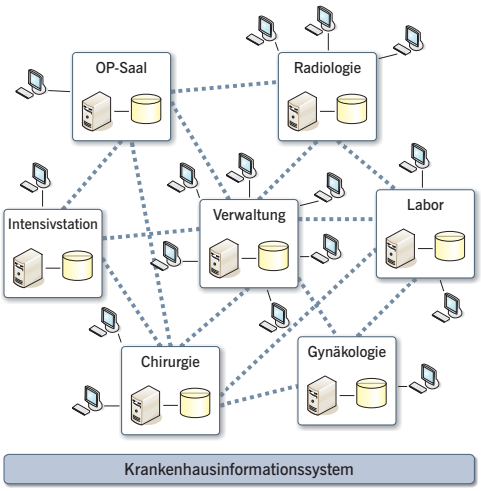


Abbildung 2.6: Schematisch: Krankenhausinformationssystem (nach [Pro03])

Häufige, alltägliche Arbeitsabläufe, wie die Verlegung eines Patienten in eine andere Abteilung, stellen hohe Anforderungen an die IT. Die Akte des Patienten muss mitverlegt werden. Existiert sie in elektronischer Form, benötigt die weiterbehandelnde Abteilung Zugriff auf bereits gemachte Eintragungen. Der Anwender wiederum will seine Aufgaben optimal unterstützt haben und Daten nicht unnötig doppelt erfassen.

Zudem benötigen viele Prozesse eine Datenverarbeitung in Echtzeit, Informationen müssen auf Abruf bereit stehen. Integration ist deshalb schon lange ein Thema und verschiedene Wege wurden beschritten.

2.4.1 Entwicklung am Universitätsklinikum Erlangen

Zur Kommunikation und Integration wird seit über 10 Jahren die *Erlanger Kommunikationsdrehscheibe* (EKDS, vgl. Abb. 2.7) eingesetzt. Die EKDS gliedert sich in zwei Hauptkomponenten, die *Kommunikationsdatenbank* (KDB, vgl. Abb. 2.7.2) als datenbankbasierte Integrationsplattform und den *Kommunikationsserver* (KS, vgl. Abb. 2.7.3) für nachrichtenbasierte Kommunikation und Integration.

Historische Entwicklung Anfang der neunziger Jahre wurden am Universitätsklinikum Erlangen erste Ansätze von datenbankbasierter Kommunikation auf der Basis des hierarchischen Datenbanksystems ADABAS C getestet. Mit der Einführung des SAP IS-H[8] Systems ging, auf Basis des relationalen Datenbank-Management-System ADABAS D, die Kommunikationsdatenbank als erster Baustein der EKDS in den Produktiveinsatz. Später wurde der Kommunikationsserver e*Gate Integrator als zweiter Baustein ergänzt. [WKKP05]

[8] Branchenlösung von SAP für Krankenhäuser zum Patientenmanagement

Aktueller Stand

Wie in Abbildung 2.7.1 dargestellt, sind derzeit über 50 Schnittstellen zu verschiedenen IT-Systemen im Universitätsklinikum Erlangen im Betrieb. Davon sind über ein Dutzend Subsysteme direkt an der KDB angeschlossen (siehe Abbildung 2.7.2). Künftiges Ziel ist es diese Systeme von der proprietären Kommunikation via SQL-Abfragen auf standardisierte Kommunikation zu migrieren. Trotzdem wird die KDB weiterhin benötigt, sie dient als Zwischenspeicher für Inhalte aus nicht standardkonformen Subsystemen und zur Realisierung von Schnittstellen, die bei einigen Subsystemen nicht vorhanden sind.

Die KDB weist einen Füllstand von über 50 GB (Stand 2004) auf, davon über 28 Millionen Laborbefunde und 1,6 Millionen Patientenfälle. Das Datenschema ist in circa 30 Bereiche verschiedener Anwendungen aufgeteilt und enthält ungefähr 600 Tabellen und 400 Nutzerkennungen. Die KDB dient nicht nur als Datenlieferant, sondern auch als Unternehmensspeicher für verschiedene Projekte wie zum Beispiel das Tumorzentrum oder die Pharmakologie. [PKSL06]

2.4.2 Standards im Gesundheitswesen

Die Erlanger Kommunikationsdrehscheibe entwickelt sich von der Kommunikationsdatenbank zunehmend zum Kommunikationsserver. Die Vorteile einer Integration auf funktionaler Anwendungsebene wurden bereits in Kapitel 2.3.1 aufgezeigt. Ermöglicht wird diese Form der Integration aber erst durch entsprechende Integrationskonzepte und Standards.

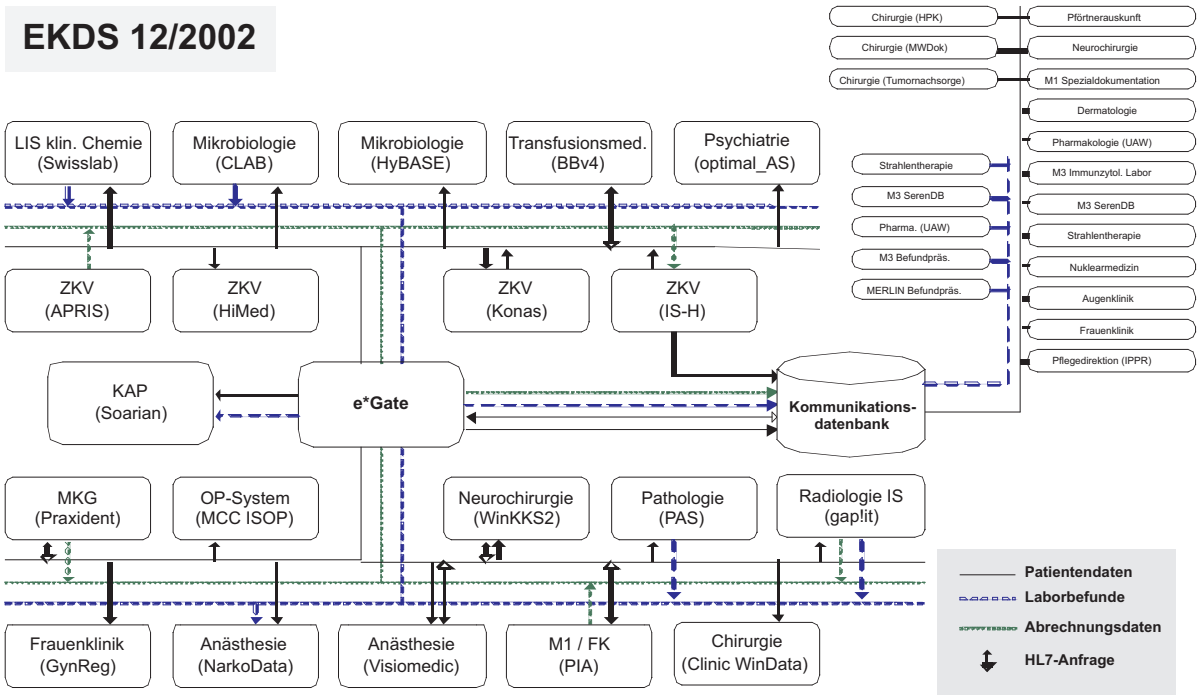
In erster Linie gehören hierzu sicherlich die etablierten Kommunikationsmodelle und -standards wie *HL7* (Health Level 7). Dabei handelt es sich um einen ANSI akkreditierten Kommunikationsstandard in der Medizin. Dieser wird von der gleich lautenden Organisation[9] zur Standardisierung im Gesundheitswesen betreut.

Das HL7-Protokoll standardisiert mit der Nachrichtenstruktur, Nachrichtendarstellung, und den nachrichtenauslösenden Ereignissen die Inhalte und die Schnittstellen zum Datenaustausch zwischen Systemen. Nachrichten werden dabei in Bereiche gegliedert, beispielsweise ADT (Patientendaten), ORU (Befunde), DFT (Leistungsdaten), ORM (Untersuchungs-Anforderungen). Beteiligte Systeme müssen also gewisse Funktionalitäten bereitstellen und auch verstehen. So werden in der EKDS Patientendaten zwar als Broadcast durch das erfassende System (IS-H) gesendet, können aber auch durch eine HL7 A19 Anfrage abgerufen werden. Beherrscht ein Anwendungssystem diese Anfrage, erspart es sich die Speicherung des gesamten Verlaufs.

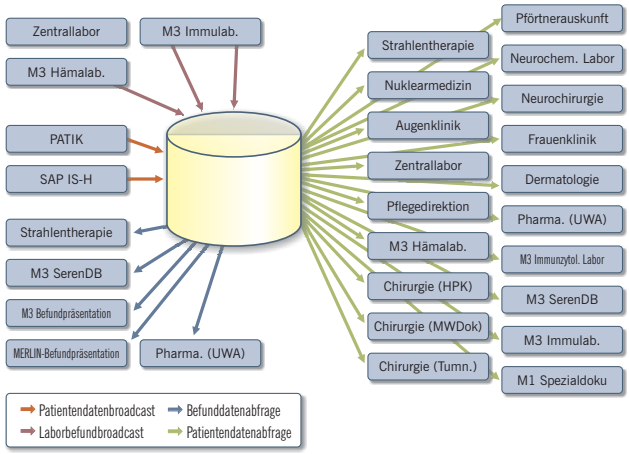
Weitere Standards wie DICOM (Digital Imaging and Communications in Medicine) oder HL7 CDA (Clinical Document Architecture) decken andere Bereiche wie bildgebende Systeme und Dokumentation ab. In der Medizin stehen traditionell Kassifikationssysteme und Kataloge zur Verfügung. Diese können zur semantischen Integration herangezogen werden.

Die Vorraussetzungen für ein funktionierendes Integrationskonzept sind also vorhanden.

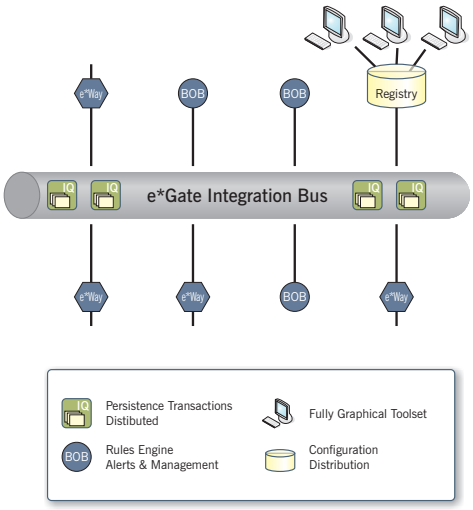
[9] HL7 Benutzergruppe in Deutschland e.V. (`http://www.hl7.de/`)



2.7.1: Erlanger Kommunikationsdrehscheibe mit Kommunikationsdatenbank und Kommunikationsserver (aus [PKSL06])



2.7.2: Detailansicht der Kommunikationsdatenbank, Stand 2002 (nach [PKSL06])



2.7.3: Detailansicht des Kommunikationsservers e*Gate Integrator (nach [PKSL06])

Abbildung 2.7: Die Erlanger Kommunikationsdrehscheibe

Daten	Quelle	Senke	Medium
Stammdaten	Student	HIS SOS	Papier
Stammdaten, Hochschulberechtigung	Bewerber	HIS ZUL	Papier
Stammdaten	ZVS	HIS ZUL	Datei per Mail
Studentische Stammdaten, Benutzerkennung	HIS SOS	RRZE Eingang	Datei
Benutzerkennung	RRZE BV	Sunray Pool	Datei
Studentische Stammdaten, Benutzerkennung	RRZE BV	NDS	LDAP
Benutzerkennung	Service Theke	RRZE BV	Manuelle Eingabe
Personaldaten	Person	UnivIS	Manuelle Eingabe
Personaldaten, eMail-Adresse	UnivIS	RRZE BV	Datei
Personaldaten	Personalabteilung	DIAPERS	Papier

Tabelle 2.1: Auszug: Informationsflüsse der Verwaltung an der FAU [CB05]

2.5 Situation der Hochschulverwaltung

Wie das Universitätsklinikum sind Hochschulen vielschichtig strukturierte Organisationen. Zur Koordination und Dokumentation der internen und externen Prozesse sind umfangreiche Verwaltungsmaßnahmen nötig. Sie erbringt dafür Dienstleistungen für alle Einrichtungen und Angehörige der Hochschule. Dazu gehören beispielsweise die Verwaltung von Personal und Studenten, führen von Bibliothekskatalogen, die Herausgabe von Vorlesungsverzeichnissen, die Koordination von Prüfungsanmeldungen und Dokumentation der Studienleistungen. Die Hochschulverwaltung erfüllt damit ihren gesetzlich festgelegten Auftrag, Hochschulaufgaben wie Forschung, Lehre und Technologietransfer zu unterstützen.

Der Ausbau der virtuellen Hochschule wird inzwischen nicht mehr primär durch die voranschreitende Informationstechnologie getrieben, Hochschulen befinden sich heute in einem zunehmenden wettbewerbsorientierten Umfeld. Reformen wie der Bologna-Prozess, die Einführung von E-Learning, Benchmarking, Evaluation und Dokumentation stellen eine Herausforderung dar und benötigen eine immer umfassendere Erhebung und Vernetzung von Daten. Eine effektive und wirtschaftliche Organisation ist deshalb von großer Bedeutung.

Situation an der Friedrich-Alexander Universität

Die Hochschulverwaltung der FAU besitzt ein breites Spektrum an Abteilungs- und Anwendungssystemen (siehe Abb. 3.2).

Informations- und Datenfluss

Die meisten Anwender wünschen sich eine schnelle und unkomplizierte Nutzung von Dienstleistungen der Verwaltung. Dieses Bestreben wird meist mittels moderneren Kommunikationstechniken realisiert.

Um den bisherigen Informationsfluss und den aktuellen Stand der Kommunikation zu veranschaulichen, ist in Tabelle 2.1 ein Ausschnitt der aktuellen Situation an der FAU dargestellt. Informationen werden umfassend erhoben und kommuniziert. Allerdings sind die Prozesse noch wenig koordiniert, wodurch Medienbrüche entstehen. Eine Lösung, um diese Brüche in den Prozessketten zu vermeiden, ist die Integration in ein Gesamtsystem.

Das Informationssystem UnivIS

Für Teilprobleme können monolithische Systeme (siehe 2.3.1) eine Lösung bieten. Nach dem Aufkommen des Internets wurde an einigen Hochschulen mit der Einführung von größeren Informationssystemen begonnen, vornehmlich für die Gruppe der Studenten. Sie ermöglichen über das Internet einen zentralen Zugriff auf Informationen und unterstützen Abläufe mit uniformer aber dezentraler Erfassung, die keiner einzelnen Abteilung zugeordnet sind. Beispiele sind das Lehrveranstaltungsangebot oder der Veranstaltungskalender. An der FAU wurde dieser Schritt mit dem WWW-basierten Informationssystem *UnivIS* vollzogen.

UnivIS ist ein vielseitig einsetzbares, hochschulspezifisches System zur integrierten Lehrabwicklung. Das System ermöglicht eine dezentrale Dateneingabe und bietet neben der Publikation der Inhalte im Internet, verschiedene Auswertungsmöglichkeiten und Berichte. Das System erlaubt die Erfassung von Informationen aus Forschung und Lehre (Vorlesungen, Personen, Raumdaten, Veranstaltungen, Forschungsprojekte, Publikationen, ...) dezentral per Browser über das WWW. [Conb]

UnivIS ist seit 1997 an der FAU im Produktivbetrieb und wird inzwischen an über einem Dutzend weiterer Hochschulen eingesetzt. Der Vertrieb und Support wird seit 1999 von dem 1995 gegründeten Spinoff *Config eG*[10] geleistet.

In einigen Arbeitsgruppen wird dieser Typ von Informationssystem auch als *Campusinformationssystem* bezeichnet. (vgl. [LHI04])

2.6 Hochschulinformationssysteme

Was ist nun ein Hochschulinformationssystem? Recherchiert man im Internet nach dem Begriff, finden sich eine Fülle von Ansichten, aber keine eindeutige Definition. Legt man die in Kapitel 2.2 gemachten Annahmen zugrunde, so gewinnt man die Vorstellung eines umfassenden Terminus für die gesamte Domäne der Hochschule. Die Definition sollte also ebenso umfassend sein. Einige Firmen bewerben Ihre Produkte bereits als Hochschulinformationssystem, obwohl deren Lösungen nur Teilbereiche der Hochschulverwaltung abdecken. Ein Hochschulinformationssystem existiert immer nur im konkreten Zusammenhang mit der Hochschule selbst. Eine mögliche (umfassende) Definition, in Anlehnung an die eines Krankenhausinformationssystems oder eines betrieblichen Informationssystems, könnte demnach wie folgt aussehen:

[10] Config Informationstechnik eG (`http://www.config.de/`)

Begriffsdefinition

Ein Hochschulinformationssystem ist das Teilsystem einer Hochschule, welches alle informationsverarbeitenden Prozesse und die an ihnen beteiligten menschlichen und maschinellen Handlungsträger in ihrer informationsverarbeitenden Rolle umfasst.

Analog der Definition eines KIS nach [Win97].

2.6.1 Integrationskonzepte

Um die vorhandenen heterogenen Strukturen in ein Hochschulinformationssystem zu überführen sind entsprechende Integrationskonzepte Vorraussetzung. Im Gegensatz zum Gesundheitswesen existieren dafür in der Hochschulverwaltung bisher keine allgemein anerkannten Konzepte. Und so beschränken sich die Ansätze auf einzelne Hochschulen oder Bundesländer (siehe Tabelle 2.2).

Ebene	Institutionen	Projektname	Homepage
Bundesland	Fachhochschule Neubrandenburg, Fachhochschule Stralsund, Hochschule für Musik und Theater Rostock (HMT), Hochschule Wismar, Universität Greifswald, Universität Rostock	Campus Online	[LHI04], `http://www.campusmv.de/`
Hochschule	Technische Universität München	IntegraTUM	`http://portal.mytum.de/cio/projekte/integratum/`
Hochschule	Universität Oldenburg	i³-sic!	`http://www.uni-oldenburg.de/projekti3sic/`
Hochschule	Universität Münster	Miro	`http://www.uni-muenster.de/IKM/miro/`

Tabelle 2.2: Projekte zur Entwicklung von Integrationskonzepten in Deutschland

Standards

Im Bereich der Hochschulverwaltung existieren kaum Standards. Vor allem die semantische Integration bleibt unberücksichtigt. Technische Lösungen müssen zwangsläufig auf jede Hochschule angepasst werden, in Teilbereichen gibt es aber bereits Ansätze. So beschäftigt sich das Deutsche Institut für Normung (DIN e.V.) seit einiger Zeit mit Spezifikationen und Referenzmodellen für die Bewertung von e-Learning-Systemen (PAS 1032, bzw. der DIN EN 1032 [Arb04]). Darin finden sich beispielsweise auch Prozessdefinitionen und Anforderungen an die Soft- und Hardware. Auch für den Datenaustausch gibt es Bemühungen, unter anderem das Learning Technology Standards Committee (LTSC)[11] des IEEE.

[11] `http://ltsc.ieee.org`

2.7 Resümee

Die IT-Struktur großer Organisationen ist meist sehr unübersichtlich. Die Heterogenität der eingesetzten Systeme macht eine umfassende Prozessunterstützung der Anwender durch steigenden Administrations- und Kommunikationsaufwand zunehmend schwieriger. Anwendungsintegration kann helfen diese Komplexität zu mindern und die Abläufe innerhalb einer Organisation zu verbessern.

Durch Veränderungen in den Rahmenbedingungen, die Ansprüche der Anwender und die Weiterentwicklung der Informationstechnologie muss sich auch die Hochschulverwaltung neuen Herausforderungen stellen. Allgemein steigen, durch den Wettbewerb zwischen den Hochschulen, die Qualitätsanforderungen. Die Informationsprozesse dürfen in ihrem Ablauf den Nutzern keinen Mehraufwand auferlegen. Sie müssen vollständig, zeitnah und in höchster Qualität zur Verfügung stehen. Die vorhandene IT-Infrastruktur in der Hochschule kann diesen Ansprüchen in vielen Punkten nicht gerecht werden, sie ist meist wenig integriert und bietet nur unvollständige Unterstützung der Prozesse. Die Defizite müssen oftmals durch den Benutzer ausgeglichen werden.

Exemplarisch für den Nutzen der Integration ist das Universitätsklinikum mit dem Krankenhausinformationssystem. Die gemachten Erfahrungen sollten beispielgebend sein. Die Integration in ein Hochschulinformationssystem kann nicht alleine durch technische Maßnahmen erreicht werden. Auch kann man ein Hochschulinformationssystem nicht von der Stange kaufen. Für eine erfolgreiche Integration müssen vielmehr schlüssige Konzepte und entsprechende Standards auf technischer wie insbesondere auf semantischer Ebene verfügbar sein.

3 Das Informationssystem UnivIS - Aufbau und Betrachtung im Verbund

Im vorigen Kapitel wurden die grundsätzlichen Strukturen und Probleme der Informationsverarbeitung in der Hochschulverwaltung dargestellt. Die Weiterentwicklung erfordert eine Integration der beteiligten Systeme. In diesem Zusammenhang ist eine Bestandsaufnahme der Funktionen und Fähigkeiten der einzelnen Systeme ein erster Schritt. Das bereits vorgestellte Informationssystem *UnivIS* stellt ein größeres Teilinformationssystem dar. Es deckt die Bereiche Forschung und Lehre ab und bildet eine der zentralen Datenbasen der Hochschulverwaltung. In den folgenden Kapiteln soll deshalb UnivIS genauer auf seine Eigenschaften im Verbund der Systeme untersucht werden.

3.1 Aufbau & Struktur

Das System wird in Perl[1], einer freien, plattformunabhängigen und interpretierten Programmiersprache, entwickelt. Die Implementierung ist programmiersprachlich und konzeptionell in Module gegliedert (eine Beschreibung aller Module siehe [Cona]).

Grundsätzlich ist UnivIS ein System zur elektronischen Datenverwaltung. Es gliedert sich dabei in drei Schwerpunkte: Die *Datenerfassung und -präsentation* erfolgt über ein einheitliches Frontend, in der Regel ist dieses internetbasiert. Es kann über einen Internetbrowser verwendet werden. Die *Datenverarbeitung* erfolgt in speziell entworfenen Modulen. Diese stellen Funktionalitäten für die Datenmanipulation, eine Datenlogik und Schnittstellen nach außen bereit. Und die *Datenspeicherung*, deren Datenmodell und Datenbank ebenfalls eine Eigenentwicklung sind.

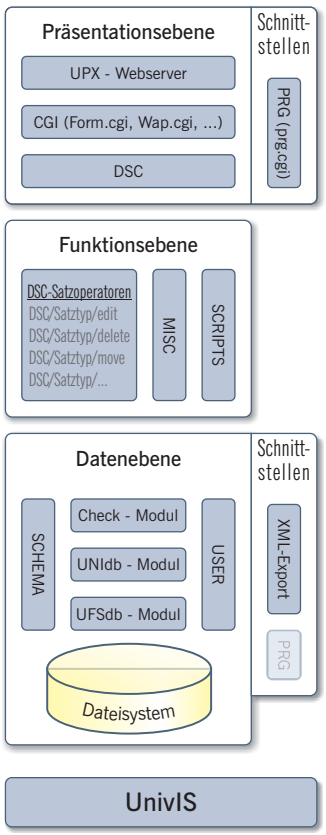
3.1.1 Schematischer Aufbau

Um UnivIS mit dem in Kapitel 2.3 eingeführten Modellen vergleichen zu können, soll hier zunächst der strukturelle Aufbau des Systems entsprechend den drei Ebenen *Präsentations-, Funktions-* und *Datenebene* dargestellt werden. Abbildung 3.1.1 zeigt den Aufbau schematisch.

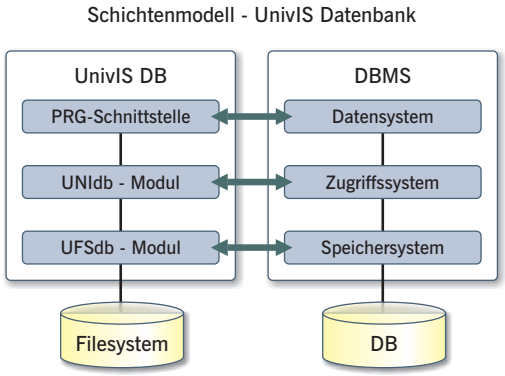
Präsentationsebene

Die Präsentationsebene bietet unter anderem das Frontend für den Benutzer, es dient sowohl zu Datenausgabe als auch zur Datenerfassung. Im Fall des UnivIS besteht diese Ebene im

[1] `http://www.perl.org/`



3.1.1: Aufbau nach dem drei-Ebenen Modell



3.1.2: UnivIS Datenebene im Schichten-modell

Abbildung 3.1: Aufbau & Struktur des UnivIS

Wesentlichen aus:

UPX: Dem UPX (UniPlex) Webserver, der den zentralen UnivIS Server repräsentiert. Der UPX nimmt alle eingehenden Verbindungen an und verteilt sie entsprechend auf die CGI-Scripte.

CGI: Die CGI-Scripte übernehmen die Bearbeitung der Anforderung und bereiten die Ausgabe entsprechend dem gewünschten Format auf (HTML, WML, ...).

DSC: Den Beschreibungsdateien welche als Vorlage (Template) für den Aufbau der Ausgabe dienen.

Die PRG-Schnittstelle kann logisch auf der Präsentationsebene gesehen werden, ist gleichzeitig aber auch Teil der Datenebene.

Funktionsebene

Diese Ebene enthält die Funktionen und Module für die Datenverarbeitung und Administration.

DSC-Satzoperatoren: Im speziellen beinhalten die Beschreibungsdateien auch Funktionalitäten für die Erfassung, Manipulation und Suche der Datenstämme. Der Satztyp bezeichnet dabei die Entität, beispielsweise `DSC/person/*` für die Personendaten (vergleiche Abb. 3.1.1).

MISC & SCRIPTS: Bieten neben den klassischen Programmbibliotheken Funktionen für den internen wie administrativen Gebrauch.

Datenebene

Die Datenebene enthält alle nötigen Module für die Speicherung, Verwaltung und Archivierung der Daten. Der XML-Dump als Schnittstelle ist hier ebenfalls anzusiedeln.

Check-Modul: Dient zur Überprüfung der Anfragen und enthält die Datenlogik.

UNIdb: Stellt die interne Satzschnittstelle für den Zugriff auf die einzelnen Datensätze.

UFSdb: Dient als Dateischnittstelle für den Zugriff auf das Speichersystem.

Dateisystem: Das Dateisystem des Betriebssystems übernimmt die Aufgabe als eigentliche Datenbank.

SCHEMA: Die SCHEMA Beschreibungsdateien enthalten das Datenschema und Metainformationen.

USER: Die Benutzerdatenbank für die Zugangskontrolle.

3.1.2 Schnittstellen

Um die Wiederverwendung der bereits eingepflegten Informationen in anderen Systemen (vornehmlich in anderen Webseiten) zu ermöglichen, bietet UnivIS entsprechende Schnittstellen für den Zugriff von Außen an.

Programmierschnittstelle:

Die *PRG Schnittstelle* (Programmierschnittstelle) bietet eine deskriptive Anfragesprache und erlaubt das Einbetten von UnivIS Inhalten in die HTML-Seite eines Aufrufers.

Dabei wird die Anfrage im HTML-Dokument durch den Tag (Auszeichner) `<UNIVIS></UNIVIS>` gekennzeichnet. Die darin enthaltene Anfrage, entsprechend dem Muster `search DATENBANK {FELD="WERT"}+ show {FORMAT}*`, wird vom UnivIS Server durch Daten ersetzt. Über das Format können verschiedene Vorlagen (Templates) zur Darstellung gewählt werden. Zum Beispiel

eine Sammlung von Lehrveranstaltungen als grafischer Stundenplan in HTML (`show plan`) oder als PDF[2] (`show pdfplan`).

Auch die Ausgabe als semistrukturiertes Dokument im XML Format ist möglich (`show xml`), für eine vollständige Referenz der Schnittstelle siehe [Con05] (S. 153ff).

XML Dump:

UnivIS kann PRG Anfragen zwar auch im XML Format beantworten, nicht öffentlich zugängliche Daten können auf diesem Weg aber nicht erreicht werden. Deshalb stehen dem Administrator auf dem UnivIS-Server zusätzliche Scripte zur Verfügung, die ganze Teile der Datenbank als *XML Dump* exportieren können. Siehe auch hierzu die Referenz im UnivIS Benutzerhandbuch [Con05] (S.164ff).

Problemorientierte Adapter:

Werden Daten in anderen Formaten benötigt, so kann dies durch spezielle Adapter bedient werden. Beispielsweise den Exporter-Scripten.

3.2 UnivIS Datenbank

Die UnivIS Datenbank ist eine Eigenentwicklung. Das verwendete Datenmodell ist satz- und zugriffspfadorientiert (vgl. 2.1.3). Die Einordnung in das Schichtenmodell (vgl. Abb. 2.1.2) ist in Abbildung 3.1.2 dargestellt.

Die PRG-Schnittstelle ermöglicht Anfragen an das System. Da sie nach außen die einzige Anwendungsschnittstelle darstellt, kann sie als *Datensystem* bezeichnet werden. Das Modul UNIdb ist eine satzorientierte Datenbankschnittstelle und bietet das *Zugriffssystem*. Dem Datenmodell nach, ist auch eine Zuordnung zwischen Daten- und Zugriffssystem möglich. Die UFSdb bildet schließlich mit der internen Satz- und Dateischnittstelle das *Speichersystem*.

```
# fieldname => longname/format/must?/requires/
      message/exportlevel/xml-alias

$scheme = {
  'lastname' =>  'Nachname||yes|||1',
  'firstname' => 'Vorname||no|||1',
  'id'  => 'ID||no|||1',
};
```

Programmausdruck 3.1: Beispiel: UnivIS Schemadefinition für eine Person

```
# UFSdb record written at 00:00:00 2004/01/01


$record = {
  'firstname' => 'Hans',
  'lastname' => 'Mustermann',
  'id' => '12345678'
};
```

Programmausdruck 3.2: UnivIS Datensatz für eine Person

[2] Adobe Portable Document Format - `http://www.adobe.de`

Die Schemadefinition erfolgt in Form von Perl-Variablenstrukturen. Der Programmausdruck 3.1 zeigt ein kleines Beispiel mit den Attributen `lastname`, `firstname` und `id`, ihnen zugeordnet sind die Bedingungen welche für das Attribut gelten müssen. Jeder Satztyp wird in einer Datei abgelegt, deren Name zugleich der Bezeichner für den Satztyp ist. Alle Schemadateien zusammen ergeben das logische Datenschema der Datenbank.

Datensätze werden ebenfalls als Perl-Variablenstruktur, einzeln als Dateien abgelegt (siehe Programmausdruck 3.2). Die Verzeichnisstruktur dient dabei als Zugriffspfad.

Die Zugriffskontrolle erfolgt über einen eigenen Satztyp `USER` innerhalb der Datenbank. Die Typüberprüfung und die Integritätsbedingungen werden vom Check-Modul erbracht, bevor die Daten an das UNIdb-Modul weitergereicht werden.

3.3 UnivIS im Verbund

In Abbildung 3.2 wird die aktuelle Struktur der Verwaltungssysteme an der Friedrich-Alexander Universität dargestellt. Die Verbindungen bezeichnen die Datenflüsse zwischen den Systemen, wie sie auch in Tabelle 2.1 gezeigt sind. Die Kommunikation erfolgt dabei sowohl aktiv (über Schnittstellen) wie auch passiv (Papiergebunden oder über Datenträger).

Auch das UnivIS nimmt an der Kommunikation des Verbundes teil, im Wesentlichen handelt es sich dabei um den Export von Personen- und Lehrveranstaltungsdaten. Zugriffe über die öffentliche PRG-Schnittstelle sind vernachlässigt.

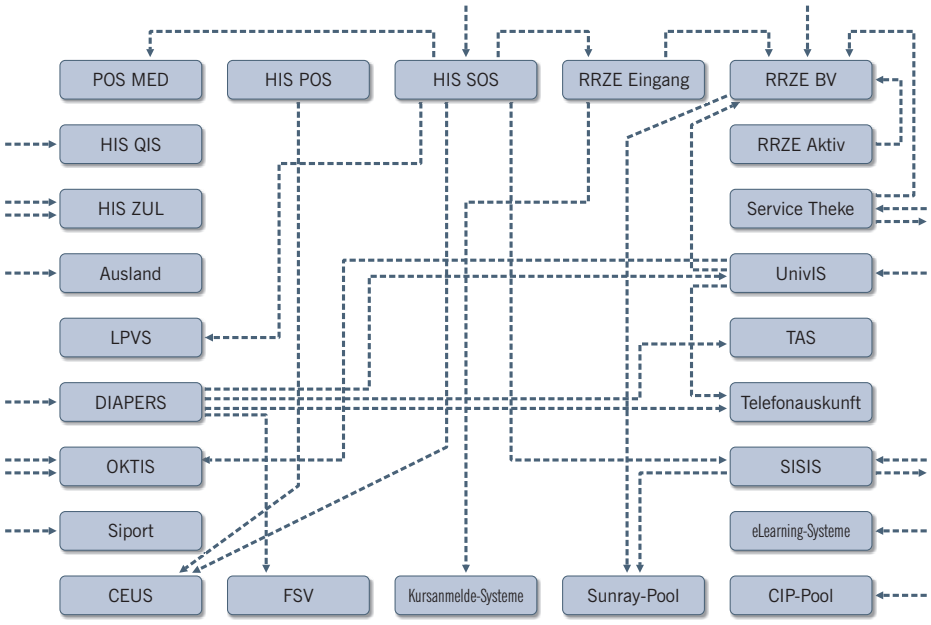


Abbildung 3.2: Schematische Darstellung des Datenflusses zwischen den Systemen der Hochschulverwaltung, Friedrich-Alexander Universität (nach [CB05])

3.3.1 Anforderungen der Anwender an das System

In Zukunft wird sich die Anzahl der Datenflüsse, besonders der aktiven Kommunikation, erhöhen. Für die weitere Analyse ist es deshalb notwendig eine Abschätzung über die künftigen Anforderungen seitens der Anwender zu treffen. In Kapitel 2.5 wurden bereits einige Prozesse die zu Änderungen führen werden (Bologna, eLearning, ...) thematisiert.

Die folgenden Annahmen basieren zum Teil auf der Anforderungsanalyse der Arbeitsgruppe Landes-Hochschul-Informationssystem [LHI04] und denen des Vergleichssystems aus 2.4 (siehe [Pro03]).

Kurzfristig

- Verwendung fremder Präsentationselemente in der eigenen Benutzerschnittstelle. Beispiel: Anzeige von Anmeldemaske und Belegung einer Übungsgruppe aus der Kursverwaltung, direkt in der entsprechenden Seite aus dem Vorlesungsverzeichnis des UnivIS.
- In Gegenrichtung auch Verwendung von UnivIS Elementen in anderen Anwendungen. Beispiel: Maske für die Erfassung von Vorlesungen.
- Übergabe von Daten in andere Systeme, falls diese vom System selbst verarbeitet werden oder die Präsentationsebenen strukturfremd sind. Beispiel: Übernahme der Personendaten in Office Anwendungen oder EMail-Systeme.
- Übernahme von Informationen aus anderen Systemen. Beispiel: Änderungen von Zugangsdaten oder EMail-Adressen aus einem Identitätsmanagementsystem.
- Umsetzung der gesetzlichen Vorgaben und den damit verbundenen Maßnahmen wie Dokumentation und Evaluation. Beispiel: Übernahme von Daten in ein elektronisches Studienbuch.

Mittelfristig

- Eine Abgrenzung der (Informations-) Domänen und ihrer Abteilungssysteme von einander. Die Zentralisierung der IT. Damit verbunden eine Vermeidung von Redundanzen und Transformation in ein hochschulweites Datenschema.
  Beispiel: Einführung eines Unternehmensspeichers und/oder Middlewarelösungen.
- Erfüllung von Sicherheitsanforderungen im Umgang mit vernetzten Daten.
  Beispiel: Das System muss Änderungen an Lehrinhalten dokumentieren (persistente und dauerhafte Speicherung).
- Einhaltung von Kommunikationsstandards für den elektronischen Austausch studienbegleitender Daten auf deutscher, europäischer und internationaler Ebene.
  Beispiel: Das elektronische Studienbuch wird vom Studenten im Auslandssemester weiter geführt.

Langfristig

- Umsetzung von Integrationskonzepten auf semantischer und funktionaler Ebene, sowie der verfügbaren Standards für Datenmodell und Kommunikation. Beispiel: Teilnahme an einer Kommunikationsdrehscheibe.
- Integration in ein umfassendes Hochschulinformationssystem.

3.3.2 Möglichkeiten der Integration

Ebenso sind Annahmen über die Möglichkeiten bei der Integration der Systeme zu untersuchen, da konkrete Integrationskonzepte noch nicht verfügbar sind.

Kurzfristig

Die vorhandenen Möglichkeiten der PRG-Schnittstelle und des XML-Exports können für die *Präsentationsintegration* (siehe Abb. 2.3.1) erweitert werden. Eine sinnvolle Nutzung ergibt sich dabei erst, wenn auch die Eingabemasken verfügbar gemacht werden. Eine Umsetzung kann die Einführung einer Middleware-Lösung und eines hochschulweiten Webportals sein, um die Strukturunterschiede der Präsentationsebenen auszugleichen.

Ist die Präsentationsintegration nicht ausreichend oder gewünscht, muss der Informationsaustausch durch *Kommunikation* erfolgen. Bisher werden Informationen zwischen den Systemen weitgehend durch Replikation ausgetauscht. Ist der Datenfluss bidirektional wirft dieses Vorgehen das Problem der Rückintegration auf, sobald sich die Informationen semantisch überschneiden. Um Replikation zu vermeiden, müssen die Anfragemöglichkeiten verbessert werden, um feingranularen Datenaustausch zu ermöglichen.

Mittelfristig

Die *technische Datenintegration* (siehe Abb. 2.3.2) kann über eine Database Access Middleware oder einen Unternehmensspeicher realisiert werden. Um den in Kapitel 2.3.1 genannten Problemen mit Inkonsistenzen vorzubeugen, ist die gleichzeitige semantische Integration der Datenschemata anzuwenden. Dazu muss das UnivIS Datenschema gegenüber anderen in Absprachen über den Umfang und die Schlüssel (Identifier) der Entitäten abgegrenzt werden (siehe Abb. 2.4.1, Schemaintegration).

Langfristig

Funktionsintegration, nach Absprache der Funktionalitäten und Interaktionen und Standardisierung der Kommunikation.

Prozessintegration von betriebswirtschaftlicher Seite, siehe dazu [Amb06].

3.4 Resümee

Das Informationssystem *UnivIS*, für die Administration im Bereich der Forschung und Lehre, ist eine wesentliche Datenquelle in der Hochschulverwaltung der FAU. UnivIS ist eine vollständige Eigenentwicklung in der Sprache Perl. Die zentralen Komponenten des Systems sind der UnivIS-Webserver (UPX), die Module für die Präsentation und Verarbeitung der Daten (CGI und DSC), die UnivIS-Datenbank (UNIdb) sowie die Programmierschnittstelle (PRG).

Anwender können durch deskriptive Anfragen an die PRG-Schnittstelle Daten aus dem UnivIS auf eigenen Webseiten einbinden. Dadurch sind praktisch alle öffentlichen Informationen zugänglich.

Die UnivIS Datenbank verwendet ein satz- und zugriffspfadorientiertes Datenmodell, sie setzt direkt auf dem Dateisystem des Betriebssystems auf. Die Definition des Datenschemas erfolgt über Beschreibungsdateien, jeder Satztyp ist in einer Perl-Variablenstruktur abgelegt. Auch die Speicherung der Sätze erfolgt in Variablenstrukturen, als Datei.

Die Verwaltungssysteme der FAU kommunizieren durch eine Vielzahl von Datenflüssen, auch das UnivIS ist daran beteiligt. Momentan erfolgt die Kommunikation mit dem UnivIS vorrangig über die PRG-Schnittstelle oder passiv durch Austausch von Datenträgern oder Dateien. Änderungen im Umfeld der Hochschulverwaltung werden zukünftig höhere Anforderungen an die Kommunikation stellen. Den Bedürfnissen der Anwender kann durch eine Steigerung der Integrationsfähigkeit des Systems begegnet werden. Dabei stehen in Zukunft verschiedene Möglichkeiten der Integration offen. Von der Verbesserung und Erweiterung der bestehenden Schnittstellen bis zur Anwendungsintegration.

4 Integration des UnivIS in einen Verbund - Aktueller Stand

Im letzten Kapitel wurde auf die grundlegende Struktur des UnivIS eingegangen. Im Besonderen auf den Aufbau der UnivIS-Datenbank und deren Datenmodell. Ebenso wurden künftige Anforderungen an das System erörtert. Diese werden in Zukunft eine stärkere Integration des UnivIS erforderlich machen. In diesem Kapitel wird nun der aktuelle Stand im Hinblick auf die Integrationsfähigkeit untersucht.

4.1 Voraussetzungen für die Integrationsfähigkeit

UnivIS befindet sich an mehreren Hochschulen im Einsatz (siehe Kapitel [Conb]). Jede Hochschule besitzt dabei eine individuelle Struktur ihrer IT und der darin enthaltenden Datenflüsse. Wie in Kapitel 2.6.1 aufgezeigt, gibt es oft auch eigene Konzepte der Integration. Trotzdem können zumindest technische, in begrenztem Umfang auch semantische Vorraussetzungen für eine verbesserte Integrationsfähigkeit geschaffen werden.

4.1.1 Kurz- und Mittelfristig

Die Replikation von Daten zwischen den Systemen wird bereits angewendet. Eines der ersten Ziele ist es deshalb, die *Kommunikationsfähigkeit* des UnivIS auszubauen. Hierzu sind gängige Schnittstellen und Formate anzubieten. Auch muss ein gegenseitiger Datenaustausch möglich sein. Daten müssen von anderen System hinzugefügt und verändert werden können. Ein weiterer Schritt ist es, die *Datenintegrationsfähigkeit* zu verbessern.

4.1.2 Langfristig

Die kurz- und mittelfristige Planung sollte die langfristige Entwicklung immer im Auge behalten. Eine gute Kommunikations- und Datenintegrationsfähigkeit schafft die Basis für das weitere Vorgehen. Aber die semantische Integration der Systeme muss im konkreten Kontext der Hochschule oder basierend auf allgemeinen Integrationskonzepten und Standards für die Hochschulverwaltung erfolgen.

4.2 Aktueller Stand - Kommunikationsfähigkeit

Die *Kommunikationsfähigkeit* soll in diesem Zusammenhang keine Integrationsstufe bezeichnen. Sondern lediglich die Eignung des Systems mit anderen Systemen auf elektronischem Wege zu kommunizieren.

4.2.1 PRG-Schnittstelle

Die PRG-Schnittstelle ist ausgezeichnet geeignet für die schnelle und unkomplizierte Präsentation von UnivIS Informationen in Webseiten. Sie ist derzeit auch die einzige Möglichkeit für Ad-Hoc-Anfragen und bietet sich deshalb für die Kommunikation mit anderen Systemen an.

Folgende Punkte sind für die Integration problematisch:

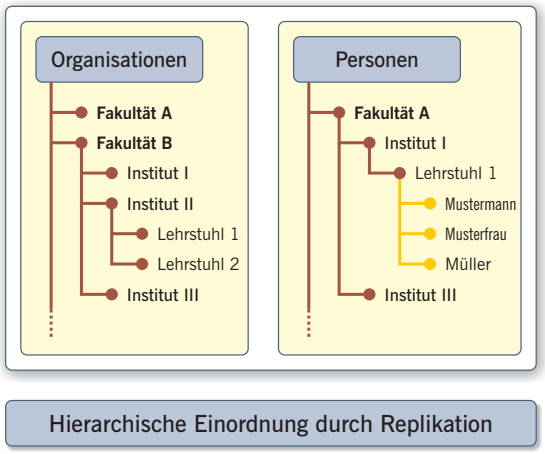
- Die Anfragesprache ist proprietär und muss bei Schemaänderungen angepasst werden.
- Ein schreibender Zugriff ist nicht möglich, Informationen können nur gelesen werden.
- Mechanismen für die Nutzer- und Zugangskontrolle fehlen. Deshalb sind nur öffentliche Informationen verfügbar.
- Die Rückgabe erfolgt nach festgelegten Templates. Werden Daten zur internen Weiterverarbeitung oder in einer anderen Darstellung benötigt, müssen sie im XML-Format abgerufen und aufbereitet werden.
- Anfragen beschränken sich auf Möglichkeiten, welche durch die Anfragesprache vorgesehen sind. Anfragen entgegen der vorgegebenen Suchparameter sind schwierig zu formulieren und benötigen Kenntnisse des internen Datenaufbaus.
- Die Schnittstelle wurde nicht für den Einsatz zwischen Systemen entwickelt. Damit ist sie auch in keine Entwicklungsumgebung eingebettet. Anfragen und Ergebnisse müssen stets übersetzt und an die eigenen programmiersprachlichen Konstrukte gebunden werden.

4.2.2 XML-Dump

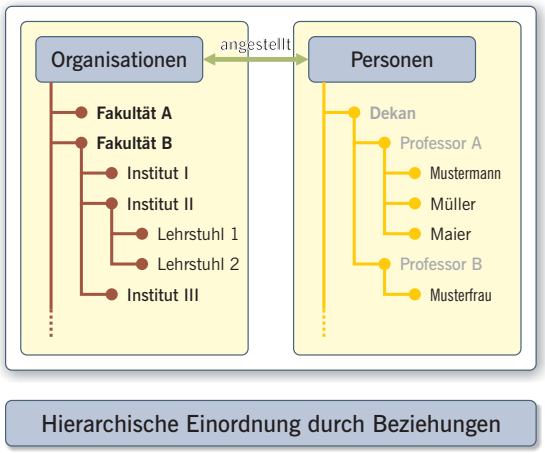
Der XML-Dump kann alle gespeicherten Informationen liefern. Die Suche und Verarbeitung ist aber vollständig dem Aufrufer überlassen. Diese Methode ist deshalb für den Versand, weniger für die Echtzeitverarbeitung geeignet.

4.2.3 Adapter

Adapter sind problemorientiert, für jeden neuartigen Datenfluss muss ein eigener Adapter entwickelt, oder zumindest die Ausgabe angepasst werden. Der Aufwand für Wartung und Pflege steigt mit der Anzahl der Adapter. Änderungen an Fremdsystemen müssen von UnivIS berücksichtigt werden.



4.1.1: Hierarchische Ordnung der Daten durch Replikation im UnivIS.



4.1.2: Hierarchische Ordnung der Daten durch Beziehungen. Jede Entität kann eine eigene Hierarchie besitzen.

Abbildung 4.1: Hierarchische Gliederung von Daten

4.3 Aktueller Stand - Datenintegrationsfähigkeit

Die Integrationsfähigkeit auf Datenebene unterteilt sich in technische und semantische Integration (siehe Kapitel 2.3.1).

4.3.1 Technische Integrationsfähigkeit

Für die technische Datenintegration (vgl. Abb. 2.3.2) sind die unterschiedlichen Datenmodelle der eingesetzten Systeme zu berücksichtigen und gegebenenfalls aufeinander abzustimmen. Das Datenmodell der UnivIS Datenbank basiert auf einer Satzschnittstelle (siehe 2.1.3). Daraus ergeben sich folgende Probleme:

- Greifen andere Systeme direkt auf die Datenebene zu, benötigen sie ein umfangreiches Wissen der internen Datenlogik der UnivIS-Datenbank. Zum Beispiel über die hierarchische Gliederung der Datensätze nach Organisationen.
- Satzschlüssel (entsprechen dem Zugriffspfad) können durch Änderungen des Satzes ungültig werden. Werden Daten repliziert, können Inkonsistenzen entstehen. Sätze können dann nicht mehr gefunden, oder zurückgeschrieben werden.
- Eine Database Access Middleware ist auf gängige Datenbanksysteme ausgelegt und bietet keine Treiber um direkt mit der UnivIS Datenebene zu kommunizieren.
- Die Eingliederung in einen gemeinsamen Datenspeicher und der Aufbau eines globalen Datenschemas (vgl. 2.3.1) ist durch die konzeptionellen Unterschiede vergleichsweise schwierig. Die UnivIS Datenebene müsste vollständig auf die Schnittstellen des Unternehmens-

speichers umgesetzt werden.

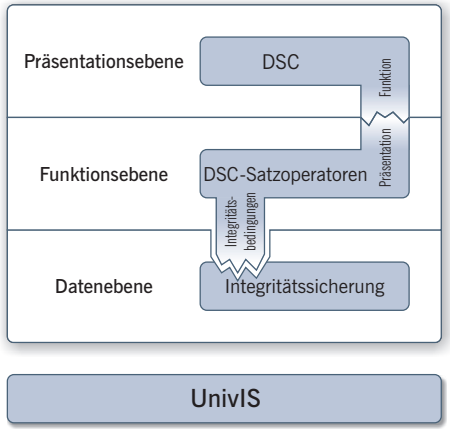


Abbildung 4.2: UnivIS Integritätsprüfung, Bruch zwischen den Ebenen.

Eine weitere Besonderheit der UnivIS Datenbank ist die Organisation der Daten. Abbildung 4.1.1 zeigt diese am Beispiel der Entitäten *Organisation* und *Person*. Die Organisationen sind entsprechend der realen Gegebenheiten hierarchisch geordnet, Personen sind einer bestimmten Organisation zugeordnet. Diese Zuordnung wird ebenfalls durch eine Hierarchie ausgedrückt, welche von den Organisationen übernommen wurde. Die Replikation der Hierarchie wiederholt sich bei allen Entitäten, die in Beziehung mit einer Organisation stehen.

Im Gegensatz dazu zeigt Abbildung 4.1.2 die Darstellung dieser hierarchischen Zuordnung durch eine Beziehung zwischen den Entitäten. Die Personen können dabei eine eigene hierarchische Ordnung aufweisen. Dieses Vorgehen entspricht der in Kapitel 2.1.4 gezeigten Datenmodellierung mit dem ER-Modell.

Da die hierarchische Einordnung in UnivIS nicht durch Beziehungen beschrieben wird, ist sie auch nicht im Datenschema verankert. Sie entsteht vielmehr durch die Position im Verzeichnisbaum. Für die Integration in ein globales Datenschema müssen diese Besonderheiten berücksichtigt und in Form von zusätzlichen Metadaten nachgebildet werden. Können die Eigenarten nicht durch die eingesetzte Middleware ausgeglichen werden, schwinden die Vorteile eines gemeinsamen Datenschemas.

4.3.2 Semantische Integrationsfähigkeit

Für die semantische Datenintegration (Abb. 2.4.1) bedarf es grundsätzlich einer Absprache zwischen den beteiligten Systemen. Es ist im Vorfeld aber möglich dafür zu sorgen, dass spätere Änderungen kleiner ausfallen, oder sogar unnötig werden.

Das UnivIS Datenschema ist an manchen Stellen schlecht normalisiert, “Funktionen” einer Person werden beispielsweise in den “Organisationen” gespeichert. Der Einsatz von Zugriffspfaden anstatt Schlüsseln und die hierarchische Ordnung ohne Verankerung im Datenschema entsprechen nicht dem gängigen Vorgehen. Dadurch steigt die Gefahr, dass im Verbund ein anderes semantisches Verständnis der Daten vorherrscht.

4.4 Aktueller Stand - Schreibender Zugriff

Die Fähigkeit Daten von anderen Systemen zu übernehmen ist eine grundlegende Vorraussetzung für die Integrationsfähigkeit. Dies gilt sowohl für die Kommunikation wie auch für die Datenintegration. Ein schreibender Zugriff auf UnivIS Daten ist derzeit nur über das Webfrontend möglich.

Dies liegt nicht nur an der fehlenden Authentifizierung in den Schnittstellen. Daten müssen durch das System überprüft werden. Ein Nutzer könnte sonst Inkonsistenzen hervorrufen, selbst unbeabsichtigt. Vor jeder schreibenden Operation muss deshalb eine Integritätsprüfung durchgeführt werden. Diese Prüfung wird in der Regel auf der Datenebene vorgenommen. So können alle Teile des Systems (Präsentation, Funktion, Schnittstellen) durch ein zentrales Element überwacht werden.

Die Integritätsprüfung der Daten erfolgt im UnivIS in Teilen bereits bei der Erfassung im Frontend. Programmausdruck 4.1 zeigt die Vermischung von Präsentationselementen, Integritätsprüfung und Datenverarbeitung wie sie in einer `DSC` Beschreibungsdatei, in diesem Fall `DSC/person/edit`, vorkommen. Die Integritätsbedingungen sind nicht vollständig zentral erfasst, wie zum Beispiel die Schemadefinitionen.

Abbildung 4.2 zeigt diesen Bruch zwischen den Ebenen schematisch. So ziehen sich die DSC-Scripte für die Datenmanipulation logisch durch die Funktionsebene bis zur Datenebene. Die Datenerfassung und Manipulation ist damit auf den Weg über die Eingabemasken beschränkt. Ein schreibender Zugriff durch Schnittstellen oder Adapter setzt das Duplizieren der Integritätsbedingungen voraus.

```
# Darstellung der Eingabemaske
<H2>Personeneintrag [[$per->{'firstname'}]] [[$per->{'lastname'}]] editieren</H2>
<table>
  <tr><td>Anrede:<td>
...
# Verhindern, dass eine Person durch umbenennen einer bereits bestehenden angelegt wird.
IF !$cgi{'pcreate'}
  $origperk = $cgi{'per'}
  $origper = $persdb{$origperk}
  ASSERT defined($origper)
  IF $per->{'lastname'} ne $origper->{'lastname'}
    push @errmsg
  ENDIF
ENDIF
...
# Einfuegen in die Datenbank
IF ! tied(%persdb)->STORE($perk, $per)
```

Programmausdruck 4.1: Beispiel: UnivIS Integritätsbedingungen für eine Person

4.5 Resümee

Nach aktuellem Stand ist eine Integrationsfähigkeit des UnivIS nur eingeschränkt vorhanden. Ein schreibender Zugriff für den bidirektionalen Datenaustausch ist nicht vorgesehen und muss nachgerüstet werden. Dafür notwendig ist die Authentifizierung der Nutzer in den Schnittstellen und die Zentralisierung der Integritätsprüfung in einem eigenen Modul.

Die Kommunikationsfähigkeit beschränkt sich auf die PRG-Schnittstelle. Sie besitzt eine eigene Anfragesprache. Diese muss bei sich ändernden Anforderungen mit angepasst werden. Die Rückgabe erfolgt formatiert als HTML oder XML. Die Verarbeitung obliegt dem aufrufenden System.

Die Fähigkeit sich in globale Datenschemata zu integrieren, ist ebenfalls durch das verwendete Datenmodell beschränkt. Die explizite Verwendung von Zugriffspfaden und die Art der hierarchischen Ordnung ist in modernen Datenbanksystemen nicht üblich und muss zusätzlich als Metainformation verwaltet werden.

Die Teilnahme an einer Kommunikationsdrehscheibe, wie sie im Universitätsklinikum eingesetzt wird, ist nur mit hohem Aufwand zu erreichen. Die hierzu notwendigen Adapter, beziehungsweise Treiber, für die UnivIS Datenbank sind nicht vorhanden.

5 Analyse und Bewertung von Änderungsmöglichkeiten

Die Integrationsfähigkeit des UnivIS ist an einigen Stellen eingeschränkt. Die Schwächen liegen in den zur Verfügung stehenden Schnittstellen, dem fehlenden schreibenden Zugriff und den Besonderheiten der UnivIS-Datenbank.

Kurz- bis mittelfristig ist die Verbesserung der Kommunikations- und Datenintegrationsfähigkeit vielversprechend. Deshalb werden im Folgenden Möglichkeiten aufgezeigt und bewertet entsprechende Verbesserungen umzusetzen.

5.1 Schreibender Zugriff

Änderungen im Bereich der Integritätsprüfung sind für einen schreibenden Zugriff generell notwendig, gleichgültig über welche Schnittstelle. Für das weitere Vorgehen müssen alle Integritätsbedingungen zentral in einem Prüfmodul zusammengefasst werden, Abbildung 5.1 zeigt die schematische Einordnung. Die Aufrufparameter der Integritätsprüfung sollten der Attributmenge einer Schemadefinition entsprechen. Das Prüfmodul selbst ist nach Satztypen gegliedert, eine Unterteilung der Prüffunktionen für das *Einfügen*, *Ändern* und *Löschen* von Sätzen ist zu empfehlen.

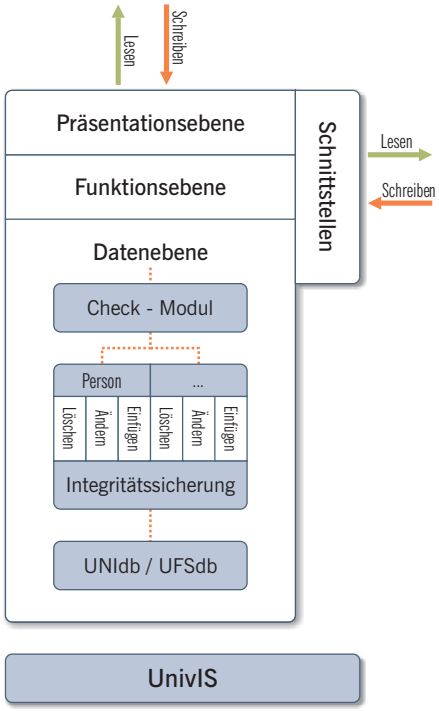


Abbildung 5.1: UnivIS mit zentraler Integritätsprüfung

Als Rückgabewert sollte bei Verletzung einer Integritätsbedingung eine Fehlermeldung im Klartext, sowie ein Fehlercode zurückgegeben werden. So kann die Fehlermeldung im Frontend direkt angezeigt werden. Alternativ kann mit dem Fehlercode eine Fehlerbehandlung durch den Aufrufer erfolgen.

Dieses Vorgehen hat zwei Vorteile. Zum einen kann die Integritätsprüfung als Funktion an die Schnittstellen weitergereicht werden. Zum anderen entspricht es der gängigen Aufteilung in Datenbanksystemen, was eine spätere Portierung erleichtert.

Der Bruch welcher zwischen den Ebenen besteht, wurde in Abbildung 4.2 gezeigt. Im Zuge der Abtrennung der Integritätsbedingungen sollten auch die

Satzoperatoren (edit, delete, insert, ...) von Präsentationselementen befreit werden. Die Einbringung in die Funktionsebene ermöglicht deren Wiederverwendung in anderen Schnittstellen.

5.2 Kommunikationsfähigkeit

5.2.1 Erweiterung der PRG-Schnittstelle

Erweiterung

Nutzer müssen sich authentifizieren, um Daten zu schreiben oder nicht öffentliche Informationen lesen zu können. Es ist möglich die PRG-Schnittstelle um eine Authentifizierungsmöglichkeit zu erweitern. Zudem benötigt die Anfragesprache eine Ergänzung um entsprechende deskriptive Konstrukte (`insert`, `update`, `delete`).

Bewertung

Die Erweiterung der PRG-Schnittstelle ist mühsam, da die Anfragesprache bei jeder Änderung des Datenschemas gepflegt werden muss. Nimmt man die Operatoren für den schreibenden Zugriff hinzu, ergibt sich sogar der vierfache Aufwand. Die Anfragesprache bleibt zudem proprietär und der Aufrufer muss sich weiterhin um das ver- und entpacken der Anfragen selbst kümmern.

5.2.2 Webservice Schnittstellen

Eine weitere Möglichkeit der Kommunikation zwischen Systemen sind die in Kapitel 2.3.2 eingeführten Webservices. Jeder Webservice bietet dabei eine bestimmte Dienstleistung. Dieser Ansatz wurde von Beyaz bereits umgesetzt. In seiner Arbeit beschreibt er den Einsatz von Webservices als Anwendungsschnittstelle (vgl. [Bey05]), jedoch nicht im Zusammenhang mit der Integration in ein Hochschulinformationssystem.

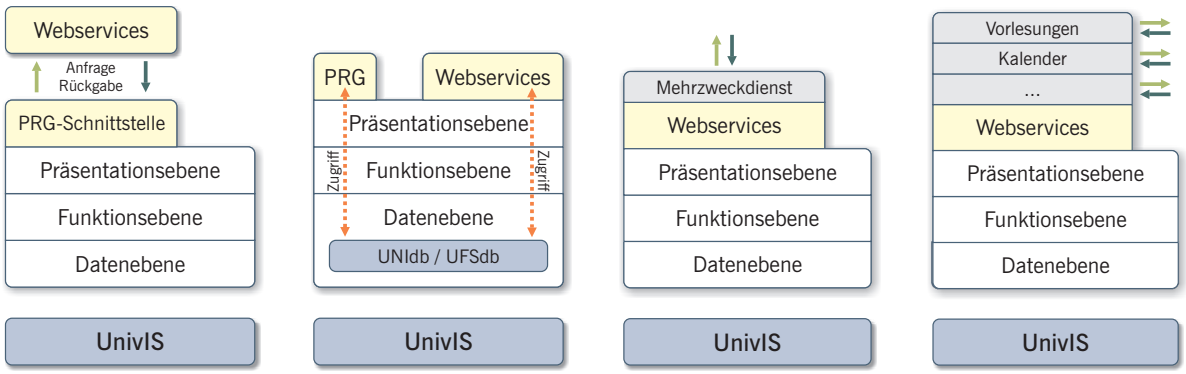
Erweiterung

UnivIS erhält mit dem Einsatz von Webservices eine weitere Schnittstelle. Datenformat und Schnittstelle des Webservice werden durch WSDL beschrieben. Die Implementierung kann deshalb weitgehend unabhängig von UnivIS erfolgen. Die von einem Webservices angebotene Dienstleistung muss schließlich an UnivIS Funktionalitäten gebunden werden.

Anbindung an UnivIS

Abbildung 5.2.1 zeigt die Anbindung über die PRG-Schnittstelle. Der Webservice fragt die benötigten Informationen mittels einer PRG-Anfrage ab, die Rückgabe erfolgt als XML.

Eine andere Möglichkeit zeigt Abbildung 5.2.2. Der Webservice ist als eigenständige Schnittstelle in das UnivIS integriert, er kann direkt auf interne Funktionen zurückgreifen.



5.2.1: Anbindung über die Programmierschnittstelle

5.2.2: Anbindung an die UnivIS Datenbank

5.2.3: Ein Webservice als vollständige Anfrageschnittstelle

5.2.4: Für jeden Anfragetyp ein Webservice

Abbildung 5.2: Webservices als weitere Schnittstelle des UnivIS

Bewertung

Webservices werden von Integrationswerkzeugen und in Entwicklungsumgebungen gut unterstützt. Sie sind einfach umzusetzen und bieten Dienste über standardisierte Schnittstellen an.

Aber es ergibt sich das Problem der Dienstspezifikation. Welche Dienste sollen angeboten werden? Zwei Extreme sind vorstellbar: Es gibt *einen Webservice* der *alle Datenoperationen* ermöglicht (vgl. Abb. 5.2.3), der damit quasi die PRG-Schnittstelle ersetzt. Oder es gibt für *jede Datenoperation* einen *speziellen Webservice*, also einen Webservice für jede Art Anfrage (vgl. Abb. 5.2.4). Der Umfang und die Anzahl der Webservices richten sich dabei nach den Nutzern der Dienste. Eine sinnvolle Nutzung kann deshalb nur bei genau definierten Dienstleistungen erfolgen.

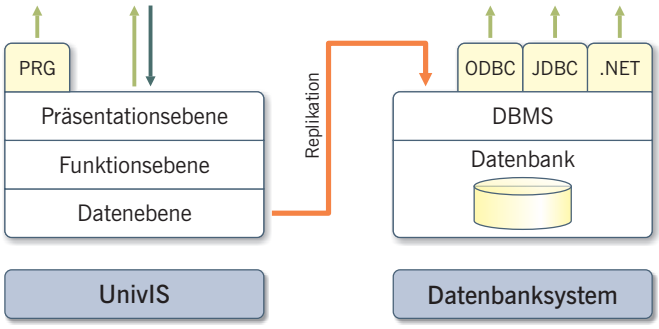
Hinzu kommt die Frage der Anbindung. Eine Bereitstellung der Zugriffsfunktionen aus den Datenbankmodulen ist problematisch, da Zugriffsrechte und Integritätsbedingungen nicht überprüft werden können (vgl. [Bey05] S.41ff). Die direkte Anbindung an die Datenebene erfordert deshalb eine Kapselung, vergleichbar der PRG-Schnittstelle. Das Aufsetzen auf die PRG-Schnittstelle (vgl. [Bey05] S.38ff) erübrigt zwar die Kapselung, erlaubt aber auch keine zusätzlichen Fähigkeiten.

Der Einsatz von Webservices alleine, kann die Integrationsfähigkeit des Systems nicht verbessern. Sie eignen sich deshalb im momentanen Umfeld nicht als generelle Erweiterung der UnivIS Systemschnittstellen.

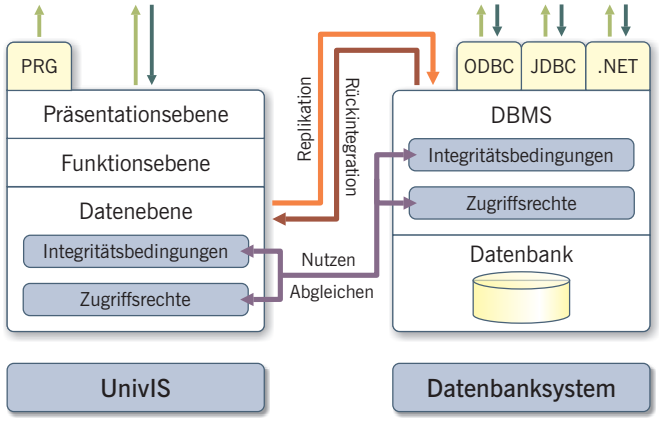
5.2.3 Datenreplikation

Wie bereits erwähnt, existiert für UnivIS keine Einbettung in Wirtssysteme. Systeme, die UnivIS Anfragen verarbeiten, müssen entsprechende Funktionalitäten selbst implementieren. Im

Gegensatz dazu bieten gängige Datenbanksysteme standardisierte Schnittstellen und Treiber an (vgl. Kapitel 2.1.6).



5.3.1: Replikation für lesenden Zugriff



5.3.2: Replikation für lesenden und schreibenden Zugriff

Abbildung 5.3: Replikation der UnivIS Datenbank in ein Datenbanksystem mit standardisierten Schnittstellen und Treibern

Erweiterung

Eine zügige Erweiterung um standardisierte Schnittstellen kann durch die Replikation der UnivIS-Datenbank in ein gängiges (relationales) Datenbanksystem, die *Replikationsdatenbank*, ermöglicht werden. Ein schrittweiser Umstieg ist ausgehend von einer unidirektionale Replikation der Daten (Abbildung 5.3.1) möglich. Auf die Replikationsdatenbank kann dabei nur lesend zugegriffen werden. Diese Variante eignet sich als Einstieg und zur Erprobung des Datenaustausches.

Der schreibende Zugriff macht eine Rückintegration der Daten nötig, die Replikation erfolgt dann bidirektional (vgl. Abb. 5.3.2). Sowohl die Integritätsbedingungen, als auch die Zugangsrechte müssen in beiden Datenbanken (UnivIS-DB und Replikations-DB) zur Verfügung stehen.

Nach der vollständigen Trennung der Präsentations- und Funktionsebene von der Datenebene,

ist es in einem letzten Schritt möglich die UnivIS Datenbank durch die Replikationsdatenbank zu ersetzen.

Bewertung

Die Replikation in ein bewährtes, relationales Datenbanksystem ermöglicht dem Nutzer den Zugriff über standardisierte Schnittstellen. Zusätzlich erhält das System mit SQL eine weitere, mächtige Anfragesprache. Problematisch ist die Extraktion des Datenschemas. Die Unterschiede zwischen dem satzorientierten und dem relationalen Datenmodell müssen berücksichtigt werden. Die hierarchische Struktur der Daten ist durch geeignete Beziehungen im Datenschema nachzubilden.

Die unidirektionale Replikation kann periodisch erfolgen. Als Austauschformat kann beispielsweise der XML Dump genutzt werden.

Die bidirektionale Replikation erfordert eine Rückintegration der Daten in Echtzeit. Da die Replikationsdatenbank standardmäßig über keine Funktionen für die Berechnung der Zugriffspfade verfügt, müssen diese implementiert oder zur Verfügung gestellt werden.

Für den schreibenden Zugriff benötigen beide Systeme die Zugriffsrechte und Integritätsbedingungen. Diese müssen entweder dupliziert oder dem Partner als Funktion bereitgestellt werden.

5.3 Datenintegrationsfähigkeit

5.3.1 Technische Integrationsfähigkeit

```
# fieldname => longname/format/must?/unique/requires/message/exportlevel/xml-alias

$scheme = {
  'lastname' =>  'Nachname||yes||||1',
  'firstname' => 'Vorname||no||||1',
  'organisation' => 'Organisation|key(Organisation)|yes||||1',
  'id'  => 'ID||yes|yes|||1',
};
```

Programmausdruck 5.1: Darstellung der Organisationszugehörigkeit durch Beziehungen

Die Änderung des Datenmodells, oder die Implementierung von standardkonformen Treibern für die UnivIS Datenbank ist aufwendig und erscheint nicht sinnvoll.

Zweckmäßig ist die generelle Einführung von Schlüsseln für den logischen Satzzugriff, wie es bei den Personen bereits durchgeführt wurde. Weiterhin die Darstellung der hierarchischen Ordnung durch Beziehungen nach dem Vorbild aus Programmausdruck 5.1. Ein Satzzugriff ohne Kenntnis des Zugriffspfades wird damit ermöglicht.

Erweiterung

Eine Verbesserung der technischen Datenintegrationsfähigkeit bringt die Replikation in ein gängiges Datenbanksystem, wie sie bereits in Kapitel 5.2.3 beschrieben wurde.

Bewertung

Die Vor- und Nachteile entsprechen denen aus Kapitel 5.2.3.

Für die Datenintegration bieten sich die standardisierten Schnittstellen an, beispielsweise für den Einsatz einer Database Access Middleware. Das Datenmodell und das Datenschema sind in der Regel portierbar und damit gut in ein globales Datenschema einzugliedern.

5.3.2 Semantische Integrationsfähigkeit

Generell sollten alle Entitäten normalisiert werden, sie sollten sich also nicht semantisch überschneiden. Dies erleichtert den Austausch einzelner Entitäten, zum Beispiel wenn die Daten in Zukunft von einem anderen System übernommen werden müssen.

Auch alle Entitäten, für die UnivIS der zentraler Datenspeicher ist, sollten als minimaler *Stammdatensatz* zur Verfügung stehen. Systeme die diese Informationen benötigen, sollten keine, oder genau diese Form der Daten enthalten. Für diese Entitäten ist UnivIS die *Masterdatenbank* und damit auch für die Vergabe von eindeutigen Identifikatoren (Identifier) verantwortlich.

5.4 Resümee

Es bestehen verschiedene Möglichkeiten die Integrationsfähigkeit des UnivIS zu verbessern. Dabei sind zwei Gesichtspunkte von entscheidender Relevanz. Zum einen der Mangel an umfassenden und idealerweise standardisierten Zugriffsmöglichkeiten. Und zum anderen die Verstrickung zwischen den Ebenen, was zum Beispiel den schreibenden Zugriff auf UnivIS Daten verhindert (vgl. Abb. 4.2). Eine klare Trennung der Ebenen ist also von zentraler Bedeutung, ein Beispiel ist die Aufspaltung der Beschreibungsdateien (`DSC`). Wie die Integritätsbedingungen auf Datenebene, sollten auch die Satzoperatoren auf Funktionsebene und die Darstellungselemente auf Präsentationsebene zusammengefasst werden.

Für die Kommunikation mit anderen Systemen steht die PRG-Schnittstelle zur Verfügung. Der Ausbau ihrer Anfragesprache ist möglich, aber mühsam. Zudem bleibt die Schnittstelle weiterhin proprietär und es sind keine Hilfsmittel für die Einbettung in andere Systeme verfügbar.

Der Einsatz von Webservices ist ebenfalls problematisch. Es existieren keine Spezifikationen für die benötigten Dienstleistungen und die Kopplung an UnivIS Funktionalitäten ist derzeit schwierig.

Durch die Replikation in ein Datenbanksystem stehen standardisierte Schnittstellen und eine gute Unterstützung für die Einbettung in Programmiersprachen zur Verfügung. Für den schreibenden Zugriff ist eine Rückintegration der Daten in die UnivIS Datenbank erforderlich. Langfristig kann diese durch die Replikationsdatenbank abgelöst und dadurch eine saubere Trennung der Datenebene erreicht werden.

Aufgrund der Eigenheiten der UnivIS-Datenbank läuft das System Gefahr im Prozess der Datenintegration ausgegrenzt zu werden. Umfangreiche Änderungen wären notwenig, was die weitere Verwendung in Frage stellt. Auch hier bietet sich der Einsatz einer Replikationsdatenbank an. Ein Zugriff über das UnivIS Datenmodell und Datenschema ist nicht mehr notwendig. Zudem werden zum Beispiel relationale Datenbanksysteme von praktisch jedem Integrationswerkzeug unterstützt. Die Eingliederung in einen Unternehmensspeicher oder der Einsatz einer Database Access Middleware werden dadurch erheblich erleichtert.

6 Lösungsvorschläge

In Kapitel 5 wurden verschiedene Möglichkeiten aufgezeigt das UnivIS auf eine Integration vorzubereiten. Die Replikation der UnivIS Datenbank in ein modernes Datenbanksystem zeigt sich dabei am vielseitigsten. Dieser Ansatz ermöglicht eine deutliche Verbesserung der Kommunikation-, wie auch der Datenintegrationsfähigkeit des Systems. Es ergeben sich jedoch auch eine Reihe von Hindernissen, wie die Abbildung des Datenschemas oder der Integritätsbedingungen.

Dieses Kapitel beschäftigt sich mit der praktischen Implementierung einer Replikationsdatenbank für das Informationssystem UnivIS. Es werden Vorschläge für die Überwindung der genannten Probleme gegeben und Implementierungsalternativen einander gegenübergestellt.

6.1 Auswahl der Replikationsdatenbank

Die Entscheidung für ein spezielles Datenbanksystem hängt von vielen Faktoren ab. Das UnivIS Datenschema ist überschaubar und erfordert keine spezielle Rücksicht auf das Datenmodell. Ein gewisser Aufwand für die Abbildung der UnivIS Eigenheiten, wie zum Beispiel der hierarchischen Ordnung (vgl. Kapitel 4.3), ist fast immer erforderlich. Abgesehen von der zu ersetzenden Satzorientierung eignen sich daher praktisch alle Datenmodelle, wie sie in Kapitel 2.1.3 bereits kurz eingeführt wurden.

6.1.1 Auswahl eines Datenmodells

Die Wahl eines Datenmodells ist trotzdem von großer Tragweite, da ein späterer Wechsel aufgrund der unterschiedlichen Standards schwierig ist.

Objektorientiert: Objektorientierte Datenbanken sind verbreitet und gut durch das ODMG[1]-Modell standardisiert. Den Datenbanksystemen fehlte es zu Anfang bei großen Datenmengen an Stabilität und Geschwindigkeit. Zudem hat sich die Objektorientierung für Standardanwendungen als zu *sperrig* erwiesen. Nach einer anfänglichen Euphorie, werden diese Systeme inzwischen hauptsächlich für den Umgang mit komplexen Datenobjekten eingesetzt.

XML: Eine neuere Entwicklung ist der Einsatz von XML als Datenmodell, es bietet sich vor allem für den Bereich der semistrukturierten Daten an. Deshalb erhofft man sich Vorteile im Umgang mit Dokumenten und Inhalten. Datenbanksysteme, die XML direkt für die

[1] Object Data Managemt Group, Untergruppe der OMG

Speicher- und Zugriffsstrukturen verwenden, befinden sich noch in weiten Teilen im Entwicklungsstadium. Ob sich diese Systeme in Zukunft auf breiter Front etablieren können ist ungewiss.

(Objekt-) Relational: Sieht man von dem Bereich der eingebetteten Systeme ab, so sind relationale Datenbanksysteme (RDBMS) die verbreitetsten. Das Modell basiert auf soliden und erforschten Konzepten. Die Schnittstellen der Systeme sind durch SQL gut standardisiert und es gibt eine breite Unterstützung in den Entwicklungsumgebungen. Zudem haben es die Hersteller verstanden sich einer aufkommenden Konkurrenz anzupassen. So wurden die grundlegenden und für den Kunden interessanten Konzepte der Objektorientierung durch Erweiterungen der Standards SQL 99 und 2003 in die relationale Welt übernommen. Ähnlich verhält es sich mit den semistrukturierten Daten, für deren praxisgerechte Verwaltung große Datenbankhersteller wie Oracle bereits seit längerem Erweiterungen anbieten.

Relationale Datenbanksysteme bieten als Replikationsdatenbank für das UnivIS eine gute und bewährte Basis. Sie eignen sich für die Integration, da sie praktisch von jedem Integrationswerkzeug unterstützt werden. Auch im Hinblick auf die UnivIS Kunden ist die Bekanntheit und die weite Verbreitung ein entscheidender Vorteil. Schließlich empfiehlt sich der Einsatz eines relationalen Datenbanksystems aus wirtschaftlichen Aspekten, da es eine Reihe frei verfügbarer und qualitativ hochwertiger Systeme gibt.

6.1.2 Auswahl eines Datenbanksystems

Von geringerer Bedeutung ist die Auswahl eines konkreten Datenbanksystems, da eine Portierung der Daten durch die vorhandenen Standards relativ unkompliziert ist. Deshalb sollte man bei der Umsetzung des Datenschemas, von der logischen auf die physische Ebene, auf eine standardkonforme Implementierung achten.

Trotzdem gibt es zwischen den Systemen Unterschiede. Zum einen sind vor allem die neueren SQL Versionen 99 und 2003 oft noch nicht vollständig umgesetzt, durch welche speziell die objektorientierten Konzepte Einzug gefunden haben. Zum anderen bauen die Hersteller proprietäre Erweiterungen in ihre Systeme ein, um sich von den Konkurrenten abzusetzen. Diese können im Einzellfall, auf Kosten der Portabilität, sehr hilfreich sein und einiges an Arbeit ersparen (Programmausdruck 6.5 zeigt dies beispielhaft). Für den konkreten Fall bietet sich eine Open Source Lösungen aus Kostengründen an.

Ein fundierter Vergleich würde den Rahmen dieser Arbeit sprengen. Zwei objektive Vergleichstests finden sich unter anderem von Schüler in [Sch05] und von Horstmann in [Hor06], aus dem auch Tabelle 6.1 auszugsweise entnommen ist. Sie zeigt die wichtigsten Funktionalitäten für den Einsatz als Replikationsdatenbank, die Funktionen sind nach Themen gruppiert und jeweils absteigend gewichtet.

Nach Punkten setzten sich *PostgreSQL*, *Firebird* und *MySQL* ab, wobei im Fall von MySQL die Eigenart der unterschiedlichen Tabellentypen berücksichtigt werden sollte. Eine Auswahl zwischen diesen Systemen wird in der Regel subjektiv, nach den Präferenzen der Entwickler und den zur Verfügung stehenden Werkzeugen zu treffen sein. Für die folgenden Beispiele wurde das

	Firebird[1]	Ingres	MaxDB[2]	MySQL	PostgreSQL
Version	1.5	3.0	7.6	5.0	8.1
Datenintegrität					
Fremdschlüssel	●	●	●	●[3]	●
Trigger	Before/After	After	After	Before/After	Before/After
ACID-Transaktionen	●	●	●	●[3]	●
CHECK-Bedingung	●	○	●	○	●
2-phasiges Commit	●	●	○	●[3]	●
Datenbankobjekte					
View	●	●	●	●	●
Stored Procedures	●	●	●	●	●
Updatable View	●	○	●	●	●
Materialized View	○	○	○	○	○
Volltext-Index	○	○	○	●[4]	●[5]
SQL, Datentypen					
SQL-Standard	92, 99	92, 99	92	92, 99	92, 99, 03
Sub-Select	●	●	●	●	●
Full Outer Join	●	●	●	○	●
Nutzerdef. Funktionen	●	●	●	●	●
Nutzerdef. Typen	○	●	○	○	●

[1]Open Source Zweig von InterBase 6.0 (Borland), seit InterBase 7.1 wieder getrennte Entwicklung
[2]vormals DDB/4 (Nixdorf) → Adabas D (Software AG) → SAP DB (SAP AG) → MaxDB (SAP AG)
[3]Nur mit Tabellentyp *InnoDB*, Hersteller Innobase Ende 2005 von Oracle aufgekauft
[4]Nur mit Tabellentyp *MyISAM* [5]Durch Add-in *Tsearch2*

Tabelle 6.1: Leistungsumfang bekannter OpenSource Datenbanksysteme (Auszug aus [Hor06])

Datenbanksystem Firebird verwendet.

6.2 Replikation

Wie bereits in Kapitel 5.2.3 erwähnt, gibt es mehrere Möglichkeiten um Daten aus UnivIS in ein Datenbanksystem zu replizieren. Für den unidirektionalen, nicht schreibenden Austausch kann der XML-Dump verwendet werden. Hierzu bieten viele Datenbanksysteme Werkzeuge (Loader) für das Importieren strukturierter Daten an. Ein schreibender Zugriff erfordert eine Replikation in Echtzeit, deshalb eignet sich XML hierfür nicht, die Replikation muss satzweise über Schnittstellen erfolgen.

Für den Zugriff auf eine Datenbank in Perl, kann das PerlDBI-Modul verwendet werden. Es bietet einen abstakten Zugriff auf das Call Level Interface der verschiedenen Datenbanksysteme an (siehe Programmausdruck 6.1).

```
use DBI;
use strict;

# Verbindung zur Datenbank aufbauen
$dbh = DBI->connect($database, $user, $password,{
      RaiseError => 1, AutoCommit => 0 });

# Datenbankanweisung vorbereiten
$sth = $dbh->prepare("INSERT INTO table(foo,bar,
      baz) VALUES (?,?,?)");

# Anweisung ausfuehren
$sth->execute( $foo, $bar, $baz );
```

Programmausdruck 6.1: Beispiel: Datenbankzugriff mit dem PerlDBI-Modul

```
/* Definition der UDF 'insertUnivIS' */

DECLARE EXTERNAL FUNCTION insertUnivIS (
    foo INTEGER,
    bar VARCHAR(10),
    baz INTEGER )
  RETURNS (returnValue INTEGER)

  /* Methodenname */
  ENTRY_POINT 'UnivISWrapper'

  /* Bibliothek (UnivISLibrary.so) */
  MODULE_NAME 'UnivISLibrary';
```

Programmausdruck 6.2: Beispiel: Einbinden einer externen Methode als *User Defined Function* in Firebird

Umgekehrt können in Datenbanksystemen, die nutzerdefinierte Funktionen (User Defined Functions - UDF) unterstützen, externe Bibliotheken genutzt werden, um die Rückintegration zu verwirklichen (siehe Programmausdruck 6.2). In einigen Systemen können, neben der internen Sprache, auch andere Programmiersprachen verwendet werden, zum Beispiel in Firebird C/C++ oder in PostgreSQL unter anderem C/C++, Java und Perl.

Es sind zwei unterschiedliche Ansatzpunkte für eine kontinuierliche Replikation möglich:

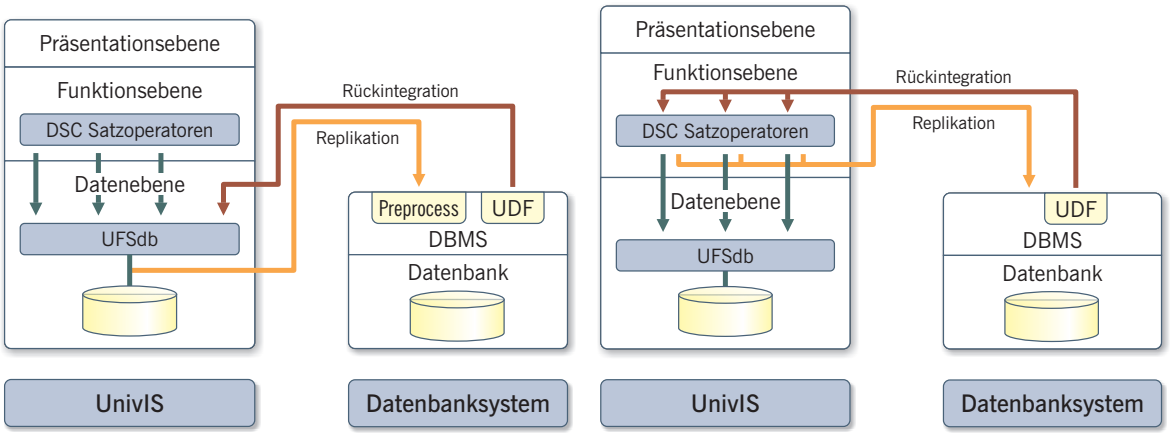
Datenebene

Dieser Ansatz ist in Abbildung 6.1.1 dargestellt. Der Datenaustausch erfolgt auf der Datenebene direkt im Speichersystem, dem UnivIS UFSdb-Modul. In diesem liegen die Daten bereits in ihrer endgültigen Form vor und werden direkt in das Dateisystem geschrieben. Die Änderungen am UnivIS beschränken sich damit auf das Weiterreichen an die Replikationsdatenbank. Diese muss die Daten im UnivIS Format annehmen, mittels Trigger oder Stored Procedures vorverarbeiten und die fehlenden Datenfelder durch berechnen oder durch parsen füllen. Falls das Datenschema normalisiert ist und zusätzliche Tabellen für einen Datensatz vorsieht, muss die Operation entsprechend aufgeteilt und in der richtigen Reihenfolge ausgeführt werden. Für die Rückintegration muss die Replikationsdatenbank den Datensatz in die UnivIS Form zurücktransformieren und an eine UDF übergeben, welche diesen dann an das UFSdb-Modul weiterreicht.

Vorteile: Die Änderungen am UnivIS sind überschaubar und beschränken sich auf eine zentrale Stelle. Im Prinzip kann auch der lesende Zugriff an die Replikationsdatenbank weitergeleitet und damit das Dateisystem als Datenbank vollständig abgelöst werden.

Nachteile: Durch die tiefe, zentrale Lage des UFSdb-Moduls wird das UnivIS Datenformat quasi zu einer Art Austauschformat zwischen den Datenbanken. Dadurch verlagert sich ein Großteil des Aufwandes in die Replikationsdatenbank, da praktisch jeder UnivIS Satzzugriff einer SQL Anweisung entspricht. Deshalb können Anfragen an die Replikationsdatenbank nicht optimiert werden und die Auswertung und Aggregation der Daten erfolgt

weiterhin im UnivIS, obwohl dies durch das Datenbanksystem geleistet werden könnte. Zudem durchläuft jede Operation die Datenebenen beider Systeme, selbst bei vollständiger Umstellung der Datenhaltung bleibt die alte Datenebene des UnivIS erhalten.



6.1.1: Replikation durch das UFSdb-Modul auf der UnivIS Datenebene, für alle Satztypen gemeinsam.

6.1.2: Replikation durch die DSC-Satzoperatoren auf der UnivIS Funktionsebene, für jeden Satztyp einzeln.

Abbildung 6.1: Ansatzpunkte für eine Replikation von UnivIS Daten

Funktionsebene

Der Datenaustausch findet, wie in Abbildung 6.1.2 dargestellt, auf der Funktionsebene statt. Die Replikation erfolgt für jeden Satztyp individuell in seinem Satzoperator, wodurch eine bessere Anpassung an die Replikationsdatenbank möglich wird. Die Rückintegration der Daten erfolgt wiederum durch UDFs, die auf den entsprechenden Satzoperator im UnivIS abbilden.

Vorteile: Durch die dezentrale Einführung können die Anfragen auf das Problem zugeschnitten werden. Die Suche, Auswertung und Aggregation der Daten kann durch die Replikationsdatenbank erfolgen. Die Datenoperationen laufen nicht durch zwei Datenebenen, sondern werden nacheinander auf beiden Systemen ausgeführt. Vollständig umgestellte Teile des UnivIS verwenden nur noch die Replikationsdatenbank als Datenebene.

Nachteile: Fast der gesamte Implementierungsaufwand liegt auf Seiten des UnivIS. Die Änderungen betreffen jedes Modul und jede Funktion, welche Daten mit der Replikationsdatenbank austauscht. Eine vollständige Umstellung der Datenhaltung kann daher erst erfolgen, wenn alle Teile des UnivIS angepasst sind.

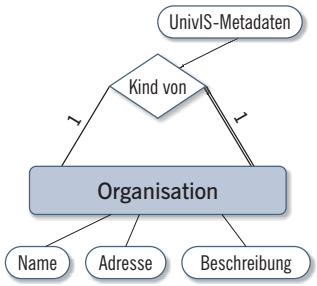


Abbildung 6.2: ER-Diagramm: Konzeptioneller Entwurf der Organisationshierarchie

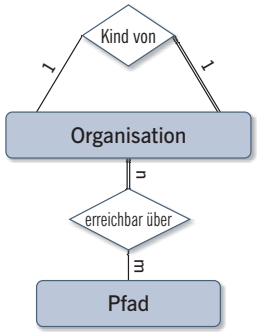


Abbildung 6.3: ER-Diagramm: Konzeptioneller Entwurf der materialisierten Pfade

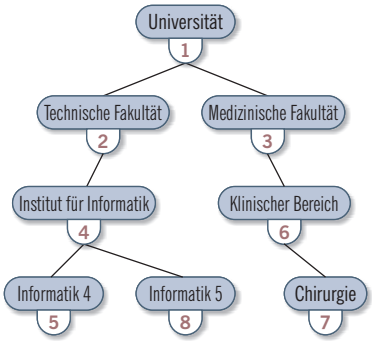


Abbildung 6.4: Beispieldaten: Darstellung der Organisationshierarchie als Baum

id	Vorgaenger	Name	Meta_Pfad
1	1	Universität	root
2	1	Technische Fakultät	tech
3	1	Medizinische Fakultät	med
4	2	Institut für Informatik	inf
5	4	Lehrstuhl für Informatik 4	inf4
6	3	Klinischer Bereich	klin
7	6	Chirurgie	chir
8	4	Lehrstuhl für Informatik 5	inf5

Tabelle 6.2: Tabelle Organisation

id	Ahne
2	1
4	1
4	2
5	1
5	2
5	4
8	1
8	2

id	Ahne
8	4
3	1
6	1
6	3
7	1
7	3
7	6

Tabelle 6.3: Tabelle Pfad, Materialisierung aller Pfade.

6.3 Datenschema extrahieren

Nach den eher konzeptionellen Entscheidungen für ein Datenbanksystem und die Replikationsebene, ist die Extraktion des Datenschemas der erste praktische Schritt. Soll die Replikation auf der Datenebene angesetzt werden, sind grundsätzlich wesentlich weniger Änderungen des Schemas nötig. Die Replikationsdatenbank muss die Datensätze ohnehin in der *alten* UnivIS Form liefern. Hingegen lohnen sich Schemaanpassungen bei einer Replikation auf Funktionseben. In diesem Fall können die UnivIS Satzoperatoren ein optimiertes Schema der Replikationsdatenbank leicht berücksichtigen, da sie in jedem Fall überarbeitet werden müssen.

In Kapitel 3 und 4 wurden bereits einige Besonderheiten des UnivIS Datenschemas und Datenmodells aufgezeigt. Im Folgenden werden diese beispielhaft und ausgehend von einer Replikation auf Funktionsebene auf relationale Strukturen abgebildet.

```
CREATE TABLE Vorlesung (
  id Integer NOT NULL,
  Vorgaenger Integer NOT NULL,
  Name varchar(100) NOT NULL,
  Adresse varchar(100) NOT NULL,
  Meta_Pfad varchar(100) NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (Vorgaenger) REFERENCES (id)
)
```

Programmausdruck 6.3: SQL Schemadefinition für die Entität Organisation

```
CREATE TABLE Pfad (
  id Integer NOT NULL,
  Ahne Integer NOT NULL,
  PRIMARY KEY (id, Ahne),
  FOREIGN KEY (id) REFERENCES Organisation(id),
  FOREIGN KEY (Ahne) Organisation(id)
)
```

Programmausdruck 6.4: SQL Schemadefinition für die Entität Pfad

6.3.1 Hierarchie

Obwohl relationale Datenbanken enorme Fähigkeiten bei der Speicherung und Auswertung von Daten besitzen, sind sie überraschend unvollständig in der Modellierung von transitiven Abhängigkeiten.

Wie bereits in Abbildung 4.1.1 gezeigt, speichert UnivIS seine Daten hierarchisch geordnet. Durch die Vorgaben des Dateisystems als Datenspeicher, handelt es sich dabei um einen Baum. Ein Baum ist ein azyklischer, zusammenhängender Graph mit einem ausgezeichneten Knoten als Wurzel. Es gibt verschiedene Möglichkeiten diese Struktur in einer relationalen Datenbank abzubilden. Ein anschauliches und auch umfassendes Werk zu diesem Thema ist "Joe Celko's Trees and Hierarchies in SQL for Smarties" ([Cel04]), aus dem im Folgenden auch einige Varianten entnommen sind.

Grundsätzlich wird die Ordnung eines Graphen durch Beziehungen dargestellt. Abbildung 6.2 zeigt ein mögliches ER-Diagramm des logischen Schemaentwurfs, die Entität Organisation besitzt eine binären (1:1) Beziehung Kind von auf sich selbst.

Eine mögliche Umsetzung in SQL ist in Programmausdruck 6.3 dargestellt. Das Attribut id dient als Ersatzschlüssel (Surrogate Key) und als eindeutiger Identifikator, da der Umgang mit numerischen Werten wesentlich einfacher ist. Um den Informationsgehalt der Daten zu erhalten, werden von der Beziehung Kind von zusätzlich noch Metainformationen mitgeführt. Diese werden der Entität Organisation als Attribut zugeschlagen, in diesem Beispiel der Verzeichnis- beziehungsweise Dateiname innerhalb des UnivIS Verzeichnisbaumes als Meta_Pfad.

Adjazenzliste

Der nächstliegende Ansatz ist die direkte Abbildung dieser Beziehung durch ein Tupel (Knoten, Vorgänger) als Adjazenzliste (vgl. Tabelle 6.2 und Abbildung 6.4). Das Attribut Vorgaenger fungiert als Zeiger auf den Elternknoten. Die referenzielle Integrität wird durch die Deklaration von id als Primärschlüssel und Vorgaenger als Fremdschlüssel aufgebaut. Diese Konstruktion ermöglicht das Speichern von zusammenhängenden, gerichteten Graphen, die Zyklenfreiheit muss allerdings separat durch einen Trigger sichergestellt werden.

```
/* Rekursive SQL Anfrage nach SQL-99 */
WITH Rekursion (id, Vorgaenger, Name) AS (
  SELECT org.id, org.Vorgaenger, org.Name
    FROM Organisation org
    WHERE org.Name = 'Technische Fakultaet' )
  UNION ALL
  SELECT kind.id, kind.Vorgaenger, kind.Name
    FROM Rekursion vater, Organisation kind
    WHERE vater.id = kind.Vorgaenger )

SELECT DISTINCT Name
  FROM Rekursion



/* Zusaetzliche Loesung von Oracle */
SELECT Name
  FROM Organisation
  START WITH Name = 'Technische Fakultaet'
  CONNECT BY id = Vorgaenger;
```

Programmausdruck 6.5: Rekursive Anfragen nach SQL/99 und Oracle spezifisch.

```
/* Hilfsfunktion fuer rekursive Anfragen in Firebird */
CREATE PROCEDURE Org_Hierarchie (root_id Integer,
      root_level Integer)
RETURNS (org_id Integer, org_level Integer)
AS
BEGIN
...
  FOR
  SELECT org_id, org_level FROM Org_Hierarchie(:
        root_id, :root_level+1) INTO :org_id, :org_level
  DO
  SUSPEND;
END

/* Abfrage */
SELECT org.Name, org.Adresse
  FROM Organisation root
  LEFT JOIN Org_Hierarchie(root.ID, 1) baum On (1=1)
  LEFT JOIN Organisation org On (org.ID = baum.Org_ID)
  WHERE root.name='Technische Fakultaet';
```

Programmausdruck 6.6: Rekursive Anfragen mit Hilfe einer Stored Procedure in Firebird.

```
/* Anfrage mit materialisierten Pfaden */
SELECT id
  FROM Pfad
  WHERE Ahne = (SELECT id
          FROM Organisation
          WHERE Name='Technische Fakultaet')
```

Programmausdruck 6.7: Anfragen über materialisierte Pfade in Firebird.

```
/* Anfrage im Nested-Set Modell */
SELECT a.id, b.Name
  FROM Organisation a, Organisation b
  WHERE a.Name='Technische Fakultaet' AND
    b.Links BETWEEN a.Links AND a.Rechts
```

Programmausdruck 6.8: Anfragen im Nested-Set Modell in Firebird.

Eine Adjazenzliste besticht durch ihren einfachen und intuitiven Aufbau. Auch ist sie robust gegen falsche Daten, ein falscher Vater-Zeiger führt *nur* zu einem falsch angehängten Ast. Auch das Umhängen von einzelnen Ästen ist relativ einfach. Problematischer ist die Ausgabe eines Teilbaumes, die Bestimmung der maximalen Tiefe, oder die Darstellung aller Knoten einer Ebene. Eine entsprechende Anfrage ist rekursiv und wird von SQL erst seit der Version SQL/99 vorgesehen. Programmausdruck 6.5 zeigt exemplarisch die Abfrage eines Teilbaumes mit der Technischen Fakultät als Wurzel durch einen SQL/99 Ausdruck und eine proprietäre Variante von Oracle.

Bedauerlicherweise wird bei allen genannten Open Source Datenbanksystemen noch an der vollständigen Umsetzung des SQL Standards gearbeitet, weshalb bislang keines rekursive Anfragen unterstützt. Eine Anfrage in SQL/92 würde aber im schlimmsten Fall aus $k = Baumhöhe - 1$ verschachtelten `SELECT...UNION` bestehen. Dies wäre einerseits recht unübersichtlich und andererseits nur bei bekanntem k brauchbar. Eine Stored Procedure kann durch ihren prozeduralen Aufbau die fehlende Funktionalität ausgleichen. Exemplarisch ist in Programmausdruck 6.6 die

Funktion Org_Hierarchie auszugsweise dargestellt. Sie liefert zu einem Startknoten den vollständigen Ast. Da die Ausgabe als Menge unsortiert ist, markiert sie die Knoten zusätzlich mit ihrer relativen Ebene. Die zugehörige Anfrage ist wieder überschaubar.

Materialisierung der Pfade

Der Einsatz von Stored Procedures oder vergleichbarer Konstrukte haben einen Nachteil: Sie berechnen bei jeder Abfrage alle möglichen Pfade zu den gesuchten Tupel neu. Eine sinnvolle Erweiterung der Adjazenzliste ist deshalb die Speicherung (Materialisierung) der Pfade.

Der einfachste Weg dies zu erreichen, ist der Einsatz einer *Materialized View*. Sie wird automatisch aktualisiert, wenn sich Daten ändern. Leider verfügen auch hier noch nicht alle Open Source Datenbanksysteme über diese Funktionalität.

Das Modell lässt sich aber auch direkt im Datenschema verankern (vgl. [Jon01]). In Abbildung 6.3 ist ein entsprechend erweitertes ER-Diagramm dargestellt. Das Datenschema wird um eine weitere Tabelle Pfad ergänzt (vgl. Programmausdruck 6.4). Sie speichert die Pfade als Tupel (id, Ahne), also alle Möglichkeiten (Ahnen) von denen aus ein Knoten (id) erreicht werden kann (siehe Tabelle 6.3).

Die Abfrage des Beispielteilbaumes kommt nun ohne die Hilfe einer Stored Procedure aus (siehe Programmausdruck 6.7). Der Aufwand verlagert sich allerdings in die Einfüge-, Lösch- und Änderungsoperationen, wo nun Trigger die Konsistenz der Tabelle Pfad sicherstellen müssen.

Die Berechnung der Pfade erfolgt nicht mehr bei jeder Anfrage erneut, sondern lediglich bei Änderungen in der Hierarchie. Insgesamt ist eine Materialisierung der Pfade mit geringem Mehraufwand bei der Implementierung verbunden, bei überwiegend lesendem Zugriff kann aber ein erheblicher Performancegewinn erreicht werden.

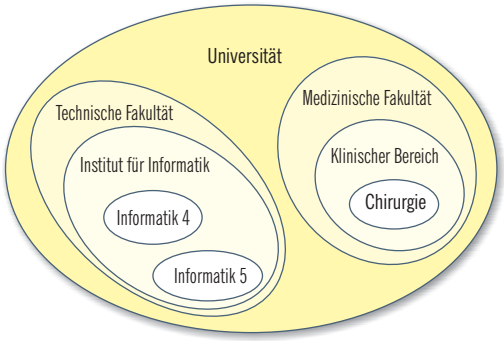


Abbildung 6.5: Mengenorientierte Darstellung der Organisationshierarchie im Nested-Set Modell.

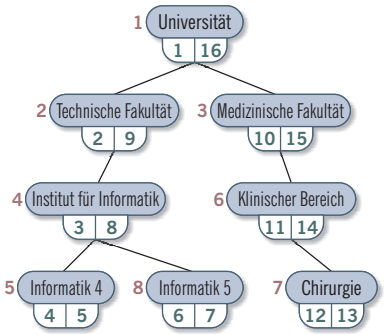


Abbildung 6.6: Beispieldaten: Umsetzung des Nested-Set Modells durch Erweiterung um linken und rechten Nachbarn.

Das Nested-Set Modell

Bisher wurde die Hierarchie als eine Folge von Knoten und Kanten betrachtet. Einen völlig anderen Ansatz verfolgt das so genannte Nested-Set Modell. Hierbei werden die Konten als eine Schachtelung von Mengen betrachtet, für die Beispieldaten veranschaulicht Abbildung 6.5 dieses Konzept. Die Ordnung wird nun nicht mehr durch explizite Beziehungen, sondern durch die Zugehörigkeit zu einer Menge definiert.

Die Umsetzung erfolgt durch Speicherung des linken und des rechten Nachfolgers. In der Tabelle Organisation wird also das Attribut Vorgänger durch die Attribute Links und Rechts ersetzt. Die Nummerierung erfolgt von links absteigend bis zu den Blattknoten und aufsteigend nach rechts für jede Ebene (oft als Preorder tree traversal bezeichnet). In Abbildung 6.6 ist die Nummerierung für die Beispieldaten angegeben. [Hil]

Die Anfragen profitieren von der mengenorientierten Darstellung. Die Beispielanfrage in Programmausdruck 6.8 zeigt wie der Teilbaum als Menge zwischen dem Attribut Links und Rechts abgefragt werden kann, auch die Auswertung eines Pfades oder einer Baumebene ist vergleichsweise einfach.

Das Nested-Set Modell kommt mit seinem Konzept relationalen Datenbanken entgegen. Auf der einen Seite lassen sich Anfragen einfach stellen und gut optimieren, im Gegenzug verliert sich der intuitive Umgang mit der Hierarchie. Die Konsistenz wird nicht mehr explizit durch die referenzielle Integrität unterstützt, da eine Beziehung zwischen den Konten nicht zwingend erforderlich ist. Deshalb müssen umfangreiche Trigger für die Einfüge-, Lösch- und Änderungsoperationen zum Einsatz kommen. Einen guten Überblick und eine Beispiel-Implementierung für MySQL findet sich von Mike Hillyer unter [Hil].

6.3.2 Semester- & Versionsverwaltung

Zusätzlich zu der Organisationshierarchie, gliedert UnivIS alle Datensätze nach Semestern. Die Datenhaltung realisiert dies durch eine weitere Verzeichnisebene oberhalb der Organisationen. Bei einem Semesterwechsel werden deshalb die gesamte Organisationshierarchie sowie alle weiterhin gültigen Datensätze dupliziert.

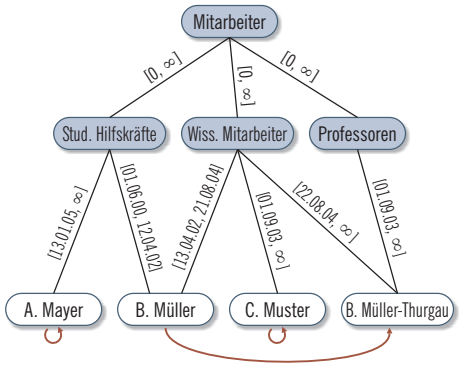


Abbildung 6.7: Zeitstempel für die Versionierung und Semestereinteilung

In Kapitel 6.3.1 wurde bereits beschrieben wie das Duplizieren der Organisationshierarchie vermieden werden kann. Um diesem Vorgehen weiter zu folgen, ist es deshalb nicht sinnvoll die Unterteilung der Semester in der Tabelle Organisation zu verankern. Die feste Zuordnung von Datensätzen zu Semestern muss mit Blick auf künftige Entwicklungen ebenfalls überprüft werden.

Wiederum gibt es verschiedene Implementierungsalternativen. Eine gängige Lösung sind flexible Gültigkeitsbereiche (Von, Bis), wie sie zum Beispiel in der Dokumentation verwendet werden.

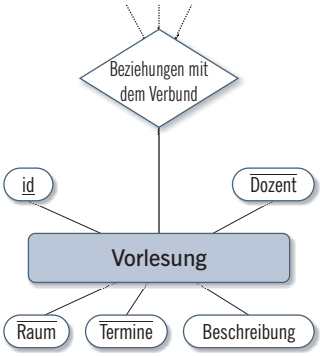
Der Einsatz von Zeitstempeln ist in Abbildung 6.7 dargestellt. Um die Historie eines Datensatzes auch bei Änderungen an auszeichnenden Daten (Schlüsselattribute) zu gewährleisten, werden diese durch eine Beziehung verknüpft (siehe Heirat von Frau Müller mit Herrn Thurgau in Abb. 6.7).

Für die grobe Einteilung in Semester eignen sich Sichten (Views), die Datensätze nach einem Zeitfenster selektieren. Für die Tabelle Organisation sind zum Beispiel Sichten der Form Organisation_[SS|WS]Jahr denkbar.

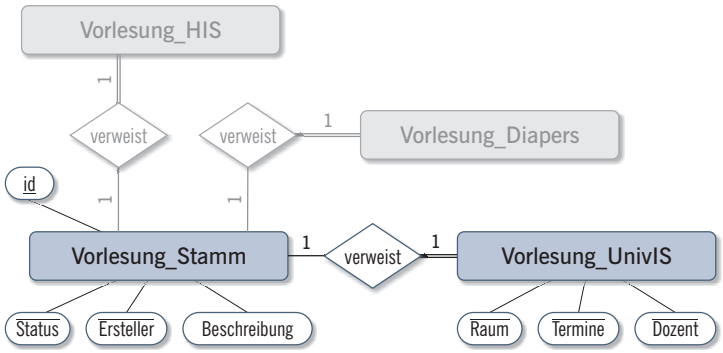
Die Versionsverwaltung und das Protokollieren von Nutzerzugriffen erledigen die meisten Datenbanksysteme auf Wunsch selbst, sodass hier kein zusätzlicher Aufwand entsteht.

6.3.3 Normalisierung

Wie bereits in Kapitel 2.1.4 eingeführt, werden Datenmodelle während der Entwurfsphase in der Regel normalisiert um die spätere Auswertung der Daten zu vereinfachen und zu bescheunigen. Dieser Vorgang kann formal und unabhängig von der eigentlichen Bedeutung der Daten erfolgen. Für die konkrete Vorgehensweise sei ebenfalls auf *Fundamentals of database systems* ([EN04]) verwiesen.



6.8.1: Innerhalb des UnivIS



6.8.2: Für den Einsatz im Verbund: Aufteilung in Stammdaten und systemspezifische Daten.

Abbildung 6.8: Normalisierung der Entität Vorlesung

Aufwändiger ist die Normalisierung und damit die Vermeidung funktionaler Abhängigkeiten zwischen den Systemen, wie sie bei der Schemaintegration (siehe Abbildung 2.4.1) auftreten können. Der Umfang und die Form gemeinsamer Datenstrukturen muss zwischen den Systemen abgestimmt werden. Die Trennung der Datenstrukturen im Vorfeld, in einen verbindlichen Stammdatensatz und optionale Attribute kann dabei erheblichen Aufwand einsparen. Für die Stammdaten kann nur ein System verantwortlich sein, es stellt die Master-Datenbank und ist für die Vergabe und Verwaltung von eindeutigen Identifikatoren verantwortlich.

Da jedes System im Verbund seine Datendomäne auf diese Weise abstecken muss, ist eine Auswahl geeigneter Entitäten ohne konkreten Kontext schwierig. Für die Entität Vorlesung könnte eine Aufteilung wie in Abbildung 6.8 aussehen. Aus der Entität Vorlesung, die nur innerhalb

des UnivIS-Datenschemas normalisiert ist, werden die Entitäten Vorlesung_Stamm und Vorlesung_UnivIS. Dadurch können systemspezifische Daten gekapselt bleiben, Änderungen an anderen Systemen sind transparent und müssen nicht in das UnivIS Datenschema eingepflegt werden.

6.4 Schreibender Zugriff

Für einen schreibenden Zugriff wurden in Kapitel 4 unter anderem die Integritätsprüfung der Daten und die Authentifizierung der Nutzer als zentrale Probleme identifiziert. Diese bestehen ebenso für die Replikationsdatenbank, falls ein direkter Zugriff gewährt werden soll (vgl. Abbildung 5.3.2). Die Vorraussetzungen aus Kapitel 5.1, wie die Zentralisierung der Integritätsprüfung und die Trennung der Ebenen, gelten weiterhin.

Wie schon bei der Extraktion des Datenschemas (Kapitel 6.3) ist der Aufwand bei einer Replikation auf Datenebene geringer. Im Wesentlichen lässt sich die Prüfung der Datenintegrität und der Zugriffsrechte mittels UDFs auf UnivIS Funktionen abbilden. Auch hier eignet sich die Replikation auf Funktionseben um die Möglichkeiten der Replikationsdatenbank besser auszuschöpfen.

6.4.1 Integritätsbedingungen

Grundsätzlich werden Integritätsbedingungen, die über die referenzielle Integrität hinausgehen in Triggern formuliert. Ein Äquivalent für das UnivIS Beispiel aus Programmausdruck 4.1 ist in Programmausdruck 6.9 dargestellt. Tigger können für eine Tabelle jeweils vor und hinter den Operationen *Ändern*, *Einfügen* und *Löschen* definiert werden.

```
/* Eine Fehlermeldung definieren */
CREATE EXCEPTION PERSON_NEW_FORM_OLD 'Wollen␣Sie␣eine␣Person␣anlegen?␣Nutzen␣Sie␣die␣
    Eingabemaske!';

/* Vor dem Einfuegen ueberpruefen. Der Vorname einer Person aendert sich in der Regel nicht. */
CREATE TRIGGER PERSONEN_BU FOR PERSONEN
ACTIVE BEFORE UPDATE POSITION 0
AS
BEGIN
    IF (NEW.vorname != OLD.vorname) THEN EXCEPTION PERSON_NEW_FORM_OLD;
END
```

Programmausdruck 6.9: Beispiel: Trigger für die Tabelle Person

Anstatt die Integritätsbedingungen in beiden Systemen zu pflegen, ist es auch möglich die Prüfung des jeweils anderen Systems zu benutzen. Einerseits können die bereits genannten UDFs auch innerhalb eines Triggers aufgerufen werden, andererseits kann ein UnivIS Satzoperator seine Aktion vom dem Gelingen der Operation auf der Replikationsdatenbank abhängig machen. Besonders sinnvoll wird dies wenn die Schemata der beiden Systeme stärker voneinander abweichen. Operationen können dann mehrere Schritte umfassen und sollten in Transaktionen zusammengefasst werden.

6.4.2 Zugriffsrechte

Die Zugriffsrechte werden als eigenständiger Datensatz im UnivIS verwaltet. Sie beziehen sich auf Organisationen, welche der Nutzer bearbeiten darf. Diese Informationen müssen in die Replikationsdatenbank übernommen werden.

Benutzerverwaltung durch die Anwendung

Die gängigste Variante ist die Nutzerverwaltung durch die Anwendung, wie es auch bei UnivIS der Fall ist. Die Authentifizierung und die Beschränkung der Nutzer werden durch das Frontend erfüllt. Eine Umsetzung in die Replikationsdatenbank ist relativ einfach. Wie alle anderen Daten, werden auch die Nutzer repliziert. Die Nutzerverwaltung des Datenbanksystems bleibt ungenutzt, weshalb sich alle zugreifenden Systeme wiederum selbst um die Zugriffsrechte kümmern müssen.

Der große Vorteil ist die unabhängige und freie Implementierung. Dagegen ist ein direkter Zugriff auf die Datenbank praktisch unmöglich, da ihr gegenüber keine Authentifizierung erfolgt.

Benutzerverwaltung durch die Datenbank

Der größte Nachteil einer Benutzerverwaltung durch die Anwendung ist der unbeschränkte Zugriff auf die Datenbanktabellen. Für eine transparente Zugriffskontrolle wird die Benutzerverwaltung des Datenbanksystems benötigt, nur sie kann den Zugriff bereits während einer Anfrage einschränken.

Leider besitzen die meisten Datenbanksysteme nur sehr rudimentäre Möglichkeiten der Benutzerverwaltung. In der Regel können nur ganze Tabellen freigegeben werden, eine Kontrolle auf Satzebene wie im UnivIS ist nicht vorgesehen.

Mischform

Um dieses Problem zu lösen, bietet sich eine Kombination beider Varianten an. Hierzu werden alle UnivIS-Nutzer auch in der Benutzerverwaltung der Replikationsdatenbank angelegt. Zusätzlich werden diese Konten mit den UnivIS-Zugriffsrechten verknüpft. Für deren Durchsetzung bis auf Satzebene, muss der direkte Zugang zu den Daten beschränkt werden. Stattdessen erfolgt der Zugriff über Sichten, die Anfragen vor der Rückgabe beschränken.

Diese Lösung kombiniert die Sicherheit und Transparenz einer Authentifizierung gegenüber der Datenbank selbst, mit den Gestaltungsmöglichkeiten einer Benutzerverwaltung durch die Anwendung.

6.5 Resümee

Die Umsetzung einer Replikationsdatenbank bringt viele Alternativen mit sich. Mit Blick auf die Kosten und die UnivIS Nutzer, bieten sich die relationalen Datenbanksysteme Aufgrund ihrer Verfügbarkeit und Verbreitung an.

Auch die Replikation zwischen UnivIS und Datenbanksystem kann auf unterschiedlichen Ebenen ablaufen. Die Replikation auf Datenebene ist dabei zwar einfacher umzusetzen, bietet aber wenig Raum für Verbesserungen und erstickt das Potential der Replikationsdatenbank. Im Gegensatz dazu ist die Replikation auf der Funktionsebene schwieriger und aufwendiger in der Umsetzung, dafür kann das Datenschema besser an die Replikationsdatenbank angepasst werden.

Im Zuge der Portierung des UnivIS-Datenschemas müssen einige der UnivIS Eigenheiten konzeptionell überarbeitet werden. Beispielsweise der Umgang mit der hierarchischen Struktur, welcher in relationalen Datenbanken so nicht vorgesehen ist. Das Konzept der materialisierten Pfade ist dabei ein guter Mittelweg zwischen intuitivem Umgang und schneller Verarbeitung. Gleiches gilt auch für die Zuordnung zu den Semestern, die durch Zeitstempel ersetzt werden kann.

Große Bedeutung hat auch die Normalisierung der Entitäten, speziell im Hinblick auf ein globales Datenschema im Verbund der Systeme. Die Trennung relevanter Datensätze in Stammdaten und systemspezifische Informationen ermöglicht es im Bedarfsfall schnell die Rolle der Master-Datenbank zu übernehmen und damit die Bedeutsamkeit des UnivIS zu untermauern.

Schließlich muss auch die Nutzerverwaltung der Replikationsdatenbank mit den UnivIS Zugriffsrechten verknüpft werden um den Nutzern einen direkten Zugriff auf die Daten zu ermöglichen.

7 Fazit

Die IT-Infrastruktur an Hochschulen besteht derzeit aus vielen unterschiedlichen Anwendungssystemen, die für einzelne Abteilungen spezialisierte Leistungen erbringen. Das UnivIS der Friedrich-Alexander Universität Erlangen-Nürnberg stellt ein größeres Informationssystem innerhalb dieser Gruppe dar. Es dient zur dezentralen Erfassung von Informationen aus Forschung und Lehre und zur integrierten Lehrabwicklung. Durch die historische Entstehung haben sich diese Systeme unabhängig und nebeneinander entwickelt. Ein Datenaustausch war ursprünglich nicht vorgesehen, weshalb Daten heute oft mehrfach und in unterschiedlichen Formaten erfasst werden müssen.

Die Entwicklung der Anwendungsdomänen mit immer höheren Erwartungen und die daraus resultierenden Probleme in den Arbeitsabläufen erfordern eine immer umfangreichere Datenerfassung und einen Austausch zwischen den verschiedenen Systemen. Diese Problematik stellt Forderungen an die Weiterentwicklung der Systeme, hin zu einem größeren Verbund, mit dem Ziel ein umfassendes Hochschulinformationssystem zu schaffen. Durch diese neuen Zielsetzungen entstehen Herausforderungen für die bestehenden Systeme und stellen sie in Konkurrenz zu anderen Produkten. Im Wesentlichen bleiben der Hochschulleitung zwei grundlegende Optionen. Entweder die Beschaffung und Einführung eines großen monolithischen Gesamtsystems, welches alle Aufgaben erfüllen kann, oder die Integration der bereits vorhandenen, heterogenen Systemlandschaft. Obwohl es von Anbietern immer wieder versprochen wird, kann man ein so komplexes Informationssystem nicht von der Stange kaufen. Anschauliches Beispiel hierfür sind die Universitätskliniken mit ihren Krankenhausinformationssystemen. Auch im Bereich der Hochschulen scheint der Einsatz eines monolithischen Systems auf Grund der hohen Komplexität eher problematisch. Für eine Integration der etablierten Systeme in einen Verbund sprechen deren Prozesskompetenz und die Vertrautheit der Anwender mit den Teilsystemen. Leider ist die Umsetzung der Integration nicht ohne Schwierigkeiten.

Im Gegensatz zum Gesundheitswesen existieren noch keine anerkannten Standards, die für diesen Zweck herangezogen werden könnten. Damit ist eine prospektive Umsetzung eines konkreten Integrationsverfahrens derzeit nicht möglich. Im Rahmen dieser Arbeit wird die generelle Integrationsfähigkeit des UnivIS untersucht und Problemstellen aufgezeigt. Diese finden sich sowohl im technischen und vor allem auch im semantischen Bereich.

UnivIS ist eine Eigenentwicklung mit eigener Datenbank und eigenem Datenmodell. Es sind keine standardisierten Schnittstellen für den Datenaustausch vorhanden und damit eine transparente Kopplung mit anderen Systemen oder der Einsatz einer Middleware praktisch nicht möglich. Aufgrund dieser individuellen Strukturen ist es schwierig UnivIS in einen Integrationsprozess einzubinden und es besteht die Gefahr dass es ausgegrenzt oder sogar ersetzt wird. Um UnivIS weiter nutzen zu können, ist es deshalb notwendig die Kommunikations- und Integrati-

onsfähigkeit zu verbessern.

Dabei zeigen sich folgende Probleme: Die bestehende PRG-Schnittstelle ist nicht für den bidirektionalen Datenaustausch ausgelegt und nur mühsam nachzurüsten. Auch eine Erweiterung des Systems um neue Schnittstellen ist Aufgrund des gewachsenen Designs problematisch. Es fehlt an einer klaren Trennung der Programmschichten und zentralen Modulen für die Wahrung der Zugriffsrechte und der Datenintegrität. Dadurch ist eine direkte Nutzung von UnivIS Funktionalitäten nicht möglich und ein Umweg über die PRG-Schnittstelle bringt keinen Mehrwert.

Im Gegensatz dazu bieten moderne Datenbanksysteme ein breites Spektrum an Schnittstellen und Integrationsmöglichkeiten. Aufgrund ihrer Anwendungs- und Datenneutralität werden sie in vielen Systemen für die Datenhaltung eingesetzt. Die Datenebene des UnivIS kann auf ein Datenbanksystem umgestellt werden, was mit umfangreichen Arbeiten verbunden ist. Eine schrittweise Migration kann diesen Aufwand in verträgliche Stücke aufteilen. Hierzu wird die Datenhaltung durch Replikation in ein parallel zum UnivIS arbeitendes Datenbanksystem übernommen. Diese Replikationsdatenbank kann dann stufenweise Teile der UnivIS Datenebene ersetzen und deren Aufgaben übernehmen. Durch die Replikation in ein Datenbanksystem profitiert das UnivIS von den bereits genannten Vorteilen und es werden standardisierte Schnittstellen, sowie eine gute Unterstützung für die Einbettung in Programmiersprachen zur Verfügung gestellt.

Dem UnivIS Kunden erlauben die hinzugewonnenen Möglichkeiten eine effektivere Weiterverwendung von UnivIS-Daten in anderen Systemen oder der Einsatz eines kundenspezifischen Integrationsverfahrens. Er ist nicht mehr auf den Einsatz eines monolithischen Informationssystems angewiesen und kann sich zugunsten der etablierten Lösungen für einen Integrationsprozess entscheiden.

Für die Implementierung empfiehlt sich ein relationales Datenbanksystem. Diese sind ausgereift, frei verfügbar und weit verbreitet. Die Replikation muss mit der UnivIS-Datenhaltung abgestimmt werden, dabei sind Änderungen am Datenschema nötig um es auf relationale Strukturen umsetzen zu können. Im Zuge dieser Arbeiten können bereits einige Hindernisse die einer technischen und semantischen Integration im Wege stehen ausgeräumt werden. Hierzu zählen beispielsweise die hierarchische Struktur der UnivIS Daten und die fehlende Normalisierung der Datensätze.

Die Config eG als Anbieter von UnivIS wird die beim Umbau gewonnenen Erkenntnisse für eine kundenspezifischere Betreuung nutzen können. Die Umstellung macht UnivIS auch flexibler und offener um künftige Standards ohne größere Probleme in das System zu integrieren und als Mehrwert dem Kunden zur Verfügung zu stellen.

Abkürzungsverzeichnis

ACID	Atomicity, Consistency, Isolation, Durability, Seite 6, 7
ADO	ActiveX Data Objects, Seite 8
API	Schnittstelle zur Anwendungsprogrammierung, [engl.] application programming interface, Seite 10
CAD	Computer Aided Design, Seite 7
CGI	Common Gateway Interface, Seite 24
CLI	Call level interface, Seite 8
DB	Datenbank, Seite 5
DBMS	Datenbankmanagementsystem, Seite 5
DBS	Datenbanksystem, Seite 5
DICOM	Digital Imaging and Communications in Medicine, Seite 17
DIN	Deutsches Institut für Normung e.V., Seite 21
DSC	Description, Seite 24
EKDS	Erlanger Kommunikationsdrehscheibe, Seite 16
FAU	Friedrich Alexander Universität Erlangen Nürnberg, Seite 1
HL7	Health Level 7, Seite 17
HL7 ADT	HL7 Nachrichtentyp: Admission Discharge and Transfer, Seite 17
HL7 DFT	HL7 Nachrichtentyp: Detailed Financial Transaction, Seite 17
HL7 ORM	HL7 Nachrichtentyp: Order Messages, Seite 17
HL7 ORU	HL7 Nachrichtentyp: Observation Report Unsolicited, Seite 17
HTML	Hypertext Markup Language, Seite 24
IEEE	Institute of Electrical and Electronics Engineers, Inc., Seite 21
IS	Informationssystem, Seite 8
IT	Informationstechnik, [engl.] Information Technology, Seite 22

JDBC	Java Database Connectivity, Seite 8
KDB	Kommunikationsdatenbank, Seite 16
KIS	Krankenhausinformationssystem, Seite 16
KS	Kommunikationsserver, Seite 16
MISC	Miscellaneous, Seite 25
ODBC	Open Database Connectivity, Seite 8
OMG	Object Management Group, Seite 7
PDF	Portable Document Format, Seite 26
PRG	Programmierschnittstelle, Seite 26
RDBMS	Relationales Datenbankmanagmentsystem, Seite 40
SOAP	Simple Object Access Protocol, Seite 14
SQL	[ugs.] Structured Query Language, Seite 17
UDDI	Universal Description, Discovery and Integration, Seite 15
UDF	User Defined Functions, Seite 48
UnivIS	Universitäts-Informationssystem, Seite 20
UPX	UniPlex, Seite 24
WAP	Wireless Application Protocol, Seite 24
WML	Wireless Markup Language, Seite 24
WSDL	Web Services Description Language, Seite 14
WWW	World Wide Web, Seite 14
XML	Extensible Markup Language, Seite 7

Literaturverzeichnis

[Amb06] Amberg, Prof. M.: Integrationstechnologien. In: *Vorlesung: Business Information Technology* Lehrstuhl für Wirtschaftsinformatik III der Friedrich-Alexander Universität Erlangen-Nürnberg, 2006. – `http://www.wi3.uni-erlangen.de/fileadmin/Dateien/Lehre/Business_IT/WS_05-06/BIT-ws05-03integration.pdf`

[Arb04] *Kapitel* Referenzmodell für Qualitätsmanagement und Qualtitätssicherung - Planung, Entwicklung, Durchführung und Evaluation von Bildungsprozessen und Bildungsangeboten. In: Arbeitsgruppe Qualität im e-Learning: *PAS 1032 - Aus- und Weiterbildung unter besonderer Berücksichtigung von e-Learning*. Bd. 1. Referat Entwicklungsbegleitende Normung im DIN Deutsches Institut für Normung e.V., 2004

[Bet] Bettag, Urban: *Web-Services*. `http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/95/`, Abruf: 03.02.2006. GI Informatiklexikon

[Bey05] Beyaz, Serkan: *Evaluierung von Web-Service-Techniken zur Integration von Anwendungsschnittstellen in ein Universitätsverwaltungssystem.*, Lehrstuhl für Informatik 4 - Verteilte Systeme und Betriebssysteme, Diplomarbeit, 2005

[BHM+04] Booth, David ; Haas, Hugo ; McCabe, Francis ; Newcomer, Eric ; Champion, Michael ; Ferris, Chris ; Orchard, David: *Web Services Architecture*. Version: 11.02.2004. `http://www.w3.org/TR/ws-arch/`, Abruf: 17.01.06

[CB05] Cooper, S. ; Bussjäger, J.: *Protokoll zum initialen Workshop zum Identitätsmanagement mit der Friedrich-Alexander-Universität Erlangen-Nürnberg*. Nicht publiziert (siehe beiliegende CD), 03 2005. – Siemens Information and Communication Networks

[Cel04] Celko, Joe: *Trees and Hierarchies in SQL for Smarties*. Morgan Kaufmann, 2004. – ISBN 1–558–60920–2

[Cona] Config Informationstechnik eG: *Interne Dokumentation & Dokumentation des Quellcodes - UnivIS Programmverzeichnis*. nicht Veröfentlicht,

[Conb] Config Informationstechnik eG: *Produktbeschreibung UnivIS*. `http://www.config.de/UnivIS/`, Abruf: 15.02.2006

[Con05] Config Informationstechnik eG: *UnivIS Benutzerhandbuch*. Online. `http://univis.uni-erlangen.de/userman.pdf`. Version: 24. Januar 2005

[ECS93] Engesser, Hermann (Hrsg.) ; Claus, Volker (Hrsg.) ; Schwill, Andreas (Hrsg.):

Duden »Informatik«: ein Sachlexikon für Studium und Praxis. Mannheim : Bibliographisches Institut & F. A. Brockhaus AG, 1993. – ISBN 3–411–05232–5

[Eic02] Eichhorn, Felix: *Evaluation von Webservice-Techniken für den Einsatz zur Business-to-Business Integration (B2BI)*, Lehrstuhl für Informatik 4 Verteilte Systeme und Betriebssysteme, Diplomarbeit, 2002

[EN04] Elmasri, Ramez ; Navathe, Shamkant B.: *Fundamentals of database systems*. Boston : Pearson/Addison Wesley, 2004. – ISBN 0–321–20448–4

[Hä01] Härder, Theo: *Datenbanksysteme - Konzepte und Techniken der Implementierung*. Berlin : Springer, 2001. – ISBN 3–540–42133–5

[Hil] Hillyer, Mike: *Managing Hierarchical Data in MySQL*. `http://dev.mysql.com/tech-resources/articles/hierarchical-data.html`, Abruf: 28.02.2006

[Hor06] Horstmann, Jutta: Freie Datenbanken im Unternehmenseinsatz. Version: 2006. `http://www.opensourcejahrbuch.de/2006/`. In: Lutterbeck, Bernd (Hrsg.) ; Bärwolff, Matthias (Hrsg.) ; Gehring, Robert A. (Hrsg.): *Open Source Jahrbuch 2006*. Lehmanns Media, 2006. – ISBN 3–865–41135–5, 181-201

[Jon01] Jonsson, Lennart: *Representing Trees in a relational DB*. Internet. `http://fungus.teststation.com/~jon/treehandling/TreeHandling.htm`. Version: 2001

[Krc00] Krcmar, Helmut: *Informationsmanagement*. Springer, Berlin, 2000. – ISBN 3–540–66359–2

[LHI04] Landes-Hochschul-Informationssystem, Arbeitsgruppe: *Landes-Hochschul-Informationssystem Campus Online*. Version: 18.05.2004. `http://www.campusmv.de/fileadmin/files/Konzept_lhis.pdf`, Abruf: 16.02.2006

[Mey05] Meyerhöfer, Marcus: XML-Datenbanken - Speicherung von XML in Datenbanken. In: *Vorlesung: Spezielle Kapitel von Datenbanken* Institut für Informatik - Lehrstuhl 6, Datenbanken der Friedrich-Alexander Universität Erlangen Nürnberg, 2005. – `http://www6.informatik.uni-erlangen.de/inf6/teaching/scripts/2005-SS/spezkap/SpezKap_11_XML04.pdf`

[PKSL06] Prokosch, Prof. Dr. Hans-Ulrich ; Kraska, Detlef ; Sojer, Reinhold ; Lentz, Richard: Kommunikationskonzepte und Standards in der Medizin. In: *Vorlesung: Informationssysteme im Gesundheitswesen 1* Lehrstuhl für medizinische Informatik der Friedrich-Alexander Universität Erlangen Nürnberg, 2006. – `http://www.imi.med.uni-erlangen.de/lehre/ws0506/medinfsys1_10.pdf`

[Pro01] Prokosch, Prof. Dr. Hans-Ulrich: KAS, KIS, EKA, EPA, EGA, E-Health - ein Plädoyer gegen die babylonische Begriffsverwirrung in der Medizinischen Informatik. In: *Informatik, Biometrie und Epidemiologie in Medizin und Biologie* 32 (2001), Nr. 4, 371-382. `http://www.imi.med.uni-erlangen.de/team/download/mis_begriffsdefinitionen.pdf`

[Pro03] Prokosch, Prof. Dr. Hans-Ulrich: Eine Einführung in Informationssysteme im Gesundheitswesen. In: *Vorlesung: Informationssysteme im Gesundheitswesen 1* Lehr-

stuhl für medizinische Informatik der Friedrich-Alexander Universität Erlangen Nürnberg, 2003. – `http://www.imi.med.uni-erlangen.de/lehre/ws0304/inf1_folien.pdf`

[Sch05] Schüler, Peter: Laden-Hüter. In: *c't* 20 (2005), S. 146–149

[Sei03] Seibold, Dr. H.: Vorlesungsfolien. In: *Vorlesung: Informationssysteme im Gesundheitswesen 1* Lehrstuhl für medizinische Informatik der Friedrich-Alexander Universität Erlangen Nürnberg, 2003. – `http://www.imi.med.uni-erlangen.de/lehre/ws0304/inf_ges_sei.pdf`

[Wik06a] Wikipedia - Die freie Enzyklopädie: *Webservice*. Version: 11.01.2006. `http://de.wikipedia.org/wiki/Webservice`, Abruf: 02.02.2006

[Wik06b] Wikipedia - Die freie Enzyklopädie: *Enterprise Application Integration*. Version: 19.01.2006. `http://de.wikipedia.org/wiki/Enterprise_Application_Integration`, Abruf: 02.02.2006

[Win97] Winter, Andreas: Krankenhaus-Informationssysteme: Begriffsbildung und Stand der Technik. Version: 1997. `http://www.uni-koblenz.de/~winter/papers/winter1997.pdf`. In: *E. Zwierlein, (Hrsg.): Klinikmanagement, Erfolgsstrategien für die Zukunft*. München : Urban & Schwarzenberg, 1997, 536-547

[WKKP05] Wentz, Bernhard ; Kraska, Detlef ; Knispel, Sabine ; Prokosch, Hans-Ulrich: *10 Jahre Erlanger Kommunikationsdrehscheibe - der Weg zu einer zukunftssicheren Integrationsplattform*. Version: 09.08.2005. `http://www.egms.de/en/meetings/gmds2005/05gmds387.shtml`, Abruf: 08.02.2006. Konferenz Abstract