5 Integration und Migration von Datenbanken

5.1 Zur Nutzung heterogener Datenbestände

Zum Umgang mit Altlasten

Viele Datenbestände sind im Laufe der Zeit strukturell verändert und stückweise erweitert worden. Als Resultat existieren heute in vielen Unternehmen inkonsistente und redundante Datensammlungen, die auf eine neue Basis gestellt werden müssen. Man spricht in diesem Zusammenhang von Altlasten (engl. *legacy systems*), die systematisch erneuert oder abgelöst werden müssen.

Datenbanken im Web

Eine Lösungsoption besteht darin, Teile der *Informationssysteme ins Web einzubetten*. Damit können die Daten und Informationen sowohl den internen Benutzern wie auch wichtigen Kundengruppen auf einfache Art und Weise zur Verfügung gestellt werden. Die Integration von Daten aus heterogenen Quellen im Web stellt eine besondere Herausforderung dar: Einerseits sind die bestehenden Datensammlungen in der Zwischenzeit vorwiegend in relationalen Datenbanken abgelegt, andererseits bieten die relationalen Datenbanksysteme zur Verwaltung von Hyperdokumenten nicht immer optimale Lösungen.

XML im Aufwind

In Abschnitt 5.2 werden Lösungsvorschläge behandelt, wie man sinnvollerweise Datenbanken im Web einbetten kann. Neben Architekturaspekten wird auch kurz auf die Entwicklung von XML (eXtensible Markup Language) und XML-Datenbanken eingegangen, damit diese Technologie mit der relationalen Datenbanktechnologie verglichen und beurteilt werden kann.

homogene Entwicklungsumgebung ist gefordert

Eine weitere Aufgabe ergibt sich, falls die Daten teilweise noch in Dateisystemen oder in herkömmlichen Datenbanken, z.B. hierarchisch oder netzwerkartig, verwaltet werden. Der Einsatz unterschiedlicher Datenbanksysteme verlangt vom Unternehmen, für die jeweiligen Technologien geeignete Datenbankspezialisten und unterschiedlich geschulte Anwendungsentwickler zu beschäftigen. Schon aus diesen Gründen ist man bestrebt, *mit der Zeit die Altlasten zu*

beheben und sämtliche Datensammlungen in relationale oder postrelationale Datenbanken zu überführen.

Komplexität bei heterogenen Systemen reduzieren

Das Nebeneinander von heterogenen Datenbeständen bildet aus wirtschaftlichen Gründen eine Hypothek, da neben personellen Aufwänden auch *Sicherheit und Verfügbarkeit komplexer Systemlandschaften* ihren Preis haben. Zudem müssen sich die Unternehmen fragen, ob sie mit überalterten und teilweise heterogen zusammengesetzten Informationssystemen den Konkurrenzkampf bestehen und die Existenz sichern können.

Abbildungsregeln zwischen Quell- und Zielsystem sind festzulegen

Bevor in diesem Kapitel verschiedene Migrationsvarianten vorgestellt werden, wird auf die *Integration von Datenbankschemas* eingegangen. Dazu werden in Kapitel 5.3 Abbildungsregeln diskutiert, die sowohl der Integration wie Migration von Datenbeständen dienen. Insbesondere muss festgelegt werden, wie Datenbestände eines Quellsystems (*legacy system*) in Datenbestände des Zielsystems überführt werden können. Falls eine Migration verschiedener Informationssysteme vorgesehen wird, kann ein solches Vorhaben mehrere Monate wenn nicht Jahre dauern. Aus diesem Grunde müssen die angesprochenen Abbildungsregeln für die Datenbankschemas eindeutig definiert sein, damit man die veralteten Bestände bei Bedarf noch eine Zeitlang nachführen kann.

Vorteil der Koexistenzvariante

In Kapitel 5.4 werden nicht nur unterschiedliche Migrationsvarianten diskutiert, sondern eine Koexistenzvariante im Detail besprochen. Eine *Koexistenz paralleler Datenbestände auf Zeit* hat den Vorteil, dass man die Anwendungssysteme nicht unter grossem Zeitdruck und entsprechenden Risiken umschreiben muss. Vielmehr kann man sich bei dieser Umstellung nach den Bedürfnissen des Marktes ausrichten und die Ablösung von Altlasten kontinuierlich vorantreiben.

Planung und schrittweise Umsetzung

Sowohl die Integration von Datenbeständen im Web wie die Migration von Anwendungssystemen muss geplant und schrittweise umgesetzt werden. Zur Illustration einer *Integrations- und Migrationsplanung* wird deshalb in Abschnitt 5.5 auf wichtige Grundsätze hingewiesen.

5.2 Datenbanken im Web

Informationen im Internet anbieten

Das World-Wide Web (WWW oder Web) hat als wichtiger Teil der Internettechnologie in den letzten Jahren eine rasante Entwicklungdurchgemacht. Mehr und mehr werden die Informations- und Datenbanksysteme ins Web eingebettet, damit das breite Informationsangebot sowohl von offenen wie geschlossenen Benutzergruppen genutzt werden kann. Insbesondere wird versucht, mit Datenbank- und

Applikationsservern die bestehenden Informationssysteme zu integrieren.

5.2.1 Aufbau eines webbasierten Informationssystems

Webbasierte Systeme unterstützen die Geschäftsprozesse

In einem *webbasierten Informationssystem* (engl. *web-based information system*) werden wichtige Dokumente und Informationen *online* verfügbar gemacht. Solche Systeme dienen nicht nur dem Informationsaustausch, sondern werden für die Beziehungspflege mit den Kunden (engl. *customer relationship management*) und für die Abwicklung elektronischer Geschäfte genutzt. Zudem werden Offertstellungen, Vertragsvereinbarungen, Distribution und Zahlungsverkehr vermehrt online organisiert, vor allem in der Wertschöpfungskette mit den Lieferanten (engl. *supply chain management*).

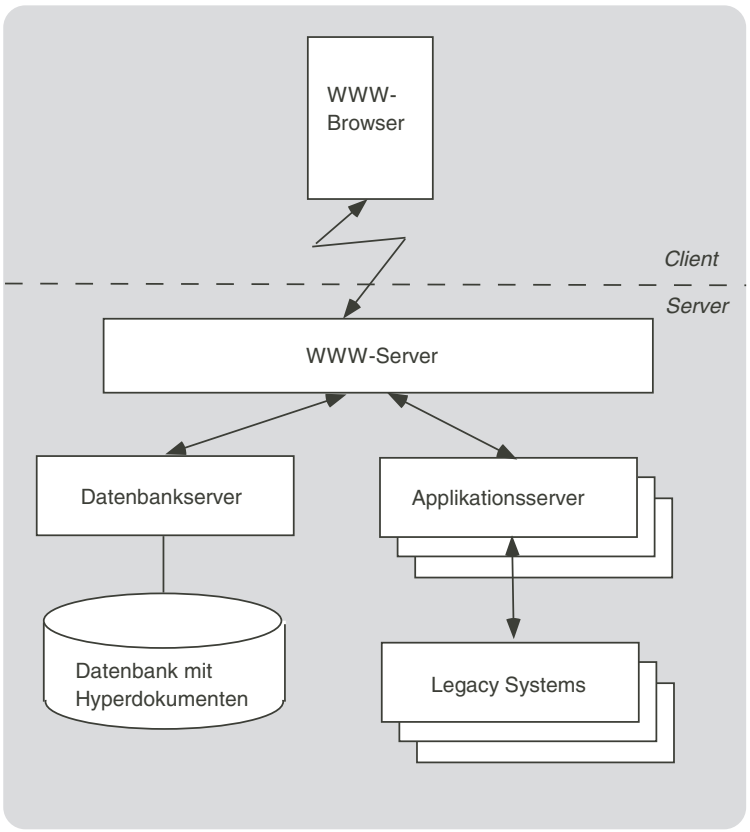


Abb. 5-1 Komponenten eines webbasierten Informationssystems

Zur Grobarchitektur

In Abb. 5-1 ist die Grobarchitektur eines webbasierten Informationssystems schematisch dargestellt. Kernelement bildet der WWW-Server, der über ein Kommunikationsprotokoll (HTTP = HyperText Transfer Protocol) Informationen in Hypertextdokumenten zur Verfügung stellt. Solche Dokumente werden vorwiegend in der Sprache HTML (HyperText Markup Language) oder der weiterführenden XML (eXtensible Markup Language) abgefasst. Auf die Verwaltung semistrukturierter Daten wird in Abschnitt 5.2.2 näher eingegangen.

Vermehrter Einsatz mobiler Endgeräte

Auf die von einem WWW-Server angebotenen Informationen kann im Normalfall rund um die Uhr und von jedem beliebigen Standort aus zugegriffen werden. Voraussetzung dazu ist ein Gerät (engl. *client*) mit einem WWW-Browser. Solche Geräte müssen nicht stationär sein, sondern können mobil eingesetzt werden; Beispiele sind Laptops, Mobiltelefone, Palms, E-Books oder digitale Assistenten.

Arbeitsteilung zwischen Datenbank- und Applikationsserver

Die Hypertextdokumente liegen entweder statisch im Dateisystem des WWW-Servers oder werden dynamisch vom Server beim Zugriff eines Benutzers erzeugt. Für die dynamische Dokumentengenerierung stehen zahlreiche Methoden und Techniken zur Verfügung, die hier nicht näher erläutert werden. In der Regel liegen die für die Dokumentenerzeugung benötigten Informationen auf dem Datenbankserver. Darüber hinaus können Daten aus den bestehenden Informationssystemen (engl. *legacy systems*) mit speziellen Schnittstellen erschlossen werden. Die so genannten Applikationsserver dienen u.a. der Verarbeitung von eingehenden Aufträgen und greifen ebenfalls auf den Datenbankserver oder die bestehenden Informationssysteme zu.

Einsatz von objekt-relationalen und XML-Datenbanken

Die Betreiber eines webbasierten Informationssystems resp. einer Website sehen sich mit einer *umfangreichen Menge von Hypertextdokumenten* konfrontiert. Aus diesem Grunde benutzen sie Datenbankserver, um die Hypertextdokumente längerfristig ablegen zu können. Der Inhalt von dynamisch generierten HTML- Seiten kann aus relationalen Datenbanken stammen. Ganze Dokumente könnten auch in objektrelationalen Datenbanksystemen abgelegt sein (vgl. Abschnitt 6.4). Eine alternative Speicherungsoption bildet die Verwaltung der semistrukturierten Daten in XML-Datenbanken. Dieser Ansatz soll im nächsten Abschnitt näher betrachtet werden.

5.2.2 XML-Dokumente und XML-Schemas

Die Markup Language XML

Die *Auszeichnungssprache* XML (eXtensible Markup Language) wurde vom World-Wide Web Consortium (W3C) entwickelt. Die Inhalte von Hypertextdokumenten werden wie bei HTML durch Tags markiert. Ein XML-Dokument ist selbstbeschreibend, da es neben

den eigentlichen Daten auch Informationen über die Datenstruktur mitführt:

Ausschnitt eines XML-Dokumentes

```
<Adresse>
    <Strasse> Rue Faucigny </Strasse>
    <Nummer> 2 </Nummer>
    <Postleitzahl> 1700 </Postleitzahl>
    <Ort> Fribourg </Ort>
</Adresse>
```

Elemente sind durch Tags markiert

Die Grundbausteine von XML-Dokumenten sind die so genannten Elemente. Diese bestehen aus einem Start-Tag (in spitzen Klammern <Name>) und einem End-Tag (in spitzen Klammern mit Schrägstrich </Name>), dazwischen steht der Inhalt des Elementes. Die Bezeichner des Start- und End-Tags müssen übereinstimmen.

XML-Dokumente können beliebig strukturiert sein

Die Tags liefern Informationen über die Bedeutung der konkreten Werte, somit sagen sie etwas über die Datensemantik aus. Elemente können in XML-Dokumenten beliebig geschachtelt werden. Zur Darstellung solcher *hierarchisch strukturierter Dokumente* wird sinnvollerweise ein Graph verwendet; ein Beispiel ist in Abb. 5-2 gegeben.

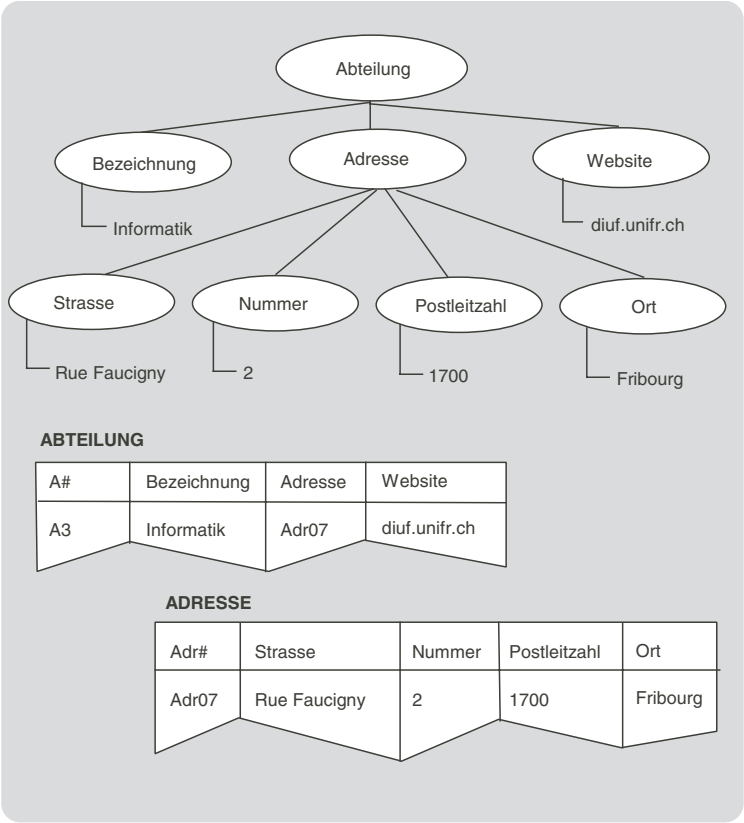
Strukturbeschreibung mit XML

Wie erwähnt, enthalten die XML-Dokumente implizit auch *Informationen über die Struktur* des Dokumentes. Da es für viele Anwendungen wichtig ist, die Struktur der XML-Dokumente zu kennen, sind explizite Darstellungen (DTD = Document Type Definition oder XML-Schema) von W3C vorgeschlagen worden. Mit einem expliziten Schema wird aufgezeigt, welche Tags im XML-Dokument auftreten und wie sie angeordnet sind. Damit lassen sich unter anderem Fehler in XML-Dokumenten lokalisieren und beheben. Im Folgenden soll das XML-Schema illustriert werden, da es für den Einsatz von Datenbanksystemen vorteilhaft ist.

Zur Zuordnung von XML-Dokumenten und relationalen Datenbanken

In Abb. 5-2 zeigen wir die Zuordnung eines XML-Dokumentes zu einem relationalen Datenbankschema. Normalerweise können relationale Datenbankschemas durch eine dreistufige Verschachtelung von Elementen charakterisiert werden, nämlich der Bezeichnung der Datenbank, den Relationennamen sowie den Attributnamen. Der Ausschnitt eines XML-Dokumentes in Abb. 5-2 zeigt die beiden Relationennamen ABTEILUNG und ADRESSE, jeweils mit den zugehörigen Attributnamen resp. Datenwerten. Die Verwendung von Schlüsseln und Fremdschlüsseln ist mit der Hilfe eines XML-Schemas möglich, worauf kurz eingegangen wird.

Abb. 5-2 Darstellung eines XML-Dokumentes in einem relationalen Datenbank-schema



XML- Schemas definieren Datentypen

Das grundlegende Konzept von XML-Schemas ist, Datentypen zu definieren und über Deklarationen Namen zu den Datentypen zuordnen zu können. Dadurch lassen sich beliebige XML-Dokumente explizit darstellen. Zudem besteht die Möglichkeit, Integritätsregeln für die Korrektheit von XML-Dokumenten zu beschreiben.

Spezifische Eigenschaften sind deklariert

Es gibt eine Vielzahl von Standarddatentypen wie String, Boolean, Integer, Date, Time u.a. Daneben können aber benutzerdefinierte Datentypen eingeführt werden. Spezifische Eigenschaften der Datentypen lassen sich durch so genannte Facets deklarieren. So kann die Ordnungseigenschaft eines Datentyps angegeben werden, die Beschränktheit der Werte durch Ober- und Untergrenzen, Längenbeschränkungen oder Aufzählung erlaubter Werte:

Beispiel eines einfachen Datentyps im XML- Schema

```
<xs:simpleType name=«Ort»>
    <xs:restriction base=«xs:string»>
        <xs:length value=«20»/>
    </xs:restriction>
```

```
</xs:simpleType>
```

Zur Darstellung der Ortschaften wird ein einfacher Datentyp vorgeschlagen, der auf dem vordefinierten Datentyp String basiert. Zudem wird verlangt, dass die Ortsnamen nicht mehr als 20 Zeichen umfassen sollen.

Schlüssel- und Referenzbeziehungen

Bei der Verwendung von XML-Schemas ist es möglich, die Schlüssel aus den Datenbanktabellen mittels «key» zu definieren. Mit der Deklaration von «keyref» lassen sich Verweise auf bereits definierte Schlüssel im XML-Schema bewerkstelligen. Mit den beiden Konstrukten können Primärschlüssel wie Fremdschlüssel definiert werden.

Darstellung von XML-Dokumenten resp. XML- Schemas

Es sind verschiedene XML-Editoren entwickelt worden, die eine grafische Darstellung eines XML-Dokumentes resp. eines XML-Schemas erlauben. Diese Editoren können sowohl für die Deklaration der Struktureigenschaften wie für die Erfassung von Dateninhalten verwendet werden. Durch das Ein- und Ausblenden von Teilstrukturen lassen sich umfangreiche XML-Dokumente resp. XML-Schemas übersichtlich anordnen.

5.2.3 Die Abfragesprache XQuery

Wert- und Strukturselektion ist mit XQuery möglich

Es ist wünschenswert, dass XML-Dokumente oder XML-Datenbanken ausgewertet werden können. Im Gegensatz zu relationalen Abfragesprachen werden nicht nur Selektionsbedingungen an Werte geknüpft (Wertselektion), sondern auch an Elementstrukturen (Strukturselektion). Weitere Grundoperationen einer XML-Abfrage betreffen die Extraktion von Subelementen eines XML-Dokumentes resp. das Verändern ausgewählter Subelemente. Das Zusammensetzen von einzelnen Elementen aus unterschiedlichen Quellstrukturen ist ebenfalls gefordert. Damit lassen sich neue Elementstrukturen erzeugen. Last but not least muss eine geeignete Abfragesprache auch mit Hyperlinks resp. Referenzen umgehen können; so genannte Pfadausdrücke sind deshalb unentbehrlich.

FLWR als Grundstruktur von XQuery

XQuery ist von W3C vorgeschlagen worden, beeinflusst durch die Sprachen SQL, unterschiedliche XML-Sprachen (z.B. XPath als Navigationssprache von XML-Dokumenten) sowie objektorientierte Abfragesprachen. Grundelement von XQuery bilden *FOR-LET-WHERE-RETURN-Ausdrücke*: FOR und LET binden eine oder mehrere Variablen an die Ergebnisse der Auswertung von Ausdrücken. Mit der WHERE-Klausel können analog zu SQL weitere Einschränkungen an die Ergebnismenge vorgenommen werden. Das Ergebnis der Abfrage wird durch RETURN angezeigt.

Ein einfaches Beispiel soll das Grundkonzept von XQuery skizzieren. Es soll eine Anfrage an das XML-Dokument «Abteilung» (siehe Abb. 5-2) gestellt werden. Dabei interessieren die Strassennamen sämtlicher Abteilungen:

Einfaches Beispiel eines FLWR-Ausdrucks

```
<StrassenNamen>
    {FOR $Abteilung IN //Abteilung
     RETURN $Abteilung/Adresse/Strasse }[1]
</StrassenNamen>
```

Zur Bindung von Variablen

Die obige Abfrage bindet die Variable $Abteilung im Laufe der Bearbeitung jeweils an die Knoten vom Typ <Abteilung>. Für jede dieser Bindungen wird durch den RETURN-Ausdruck die jeweilige Adresse evaluiert und die Strasse ausgegeben. Die Anfrage in XQuery produziert das folgende Ergebnis:

Das Resultat einer XQuery ist eine Sequenz

```
<StrassenNamen>
    <Strasse> Rue Faucigny </Strasse>
    <Strasse> . . . . . </Strasse>
    <Strasse> . . . . . </Strasse>
</StrassenNamen>
```

Variablenbindung durch FOR und LET

In XQuery werden Variablen mit einem durch das $-Zeichen ergänzten Namen eindeutig gekennzeichnet, um eine Unterscheidung zu den Namen von Elementen zu machen. Im Gegensatz zu einigen Programmiersprachen können Variablen in XQuery keine Werte zugewiesen werden. Vielmehr müssen Ausdrücke ausgewertet und das Ergebnis an die Variablen gebunden werden. Diese Variablenbindung erfolgt bei XQuery mit den FOR- und LET-Ausdrücken.

Ergebnis einer Anfrage wird mit RETURN angezeigt

Im obigen Anfragebeispiel wird auf die Spezifikation des LET-Ausdrucks verzichtet. Mit der WHERE-Klausel liesse sich zudem die Ergebnismenge weiter einschränken. Die Auswertung der RETURN-Klausel erfolgt für jeden Schleifendurchlauf mit FOR, muss aber nicht zwingend ein Resultat liefern. Die einzelnen Resultate hingegen werden aneinander gereiht und bilden das Ergebnis des FOR-LET-WHERE-RETURN-Ausdrucks.

XQuery als Anfragesprache für XML-Dokumente und XML-Datenbanken

XQuery ist eine *mächtige Abfragesprache für Hyperdokumente* und wird sowohl für XML-Datenbanken wie auch für einige postrelationale Datenbanksysteme angeboten. Damit relationale Datenbanksysteme XML-Dokumente speichern können, müssen Erweiterungen in der Speicherungskomponente vorgenommen werden. Beispielsweise erlauben objektrelationale Datenbanksysteme dank ihren hierarchischen Strukturierungskonzepten (siehe 6.4), mit XML zu arbeiten.

[1] Geschweifte Klammern stammen von der Navigationssprache XPath. Entsprechende Ausdrücke werden vom Parser direkt ausgewertet.

5.3 Abbildungsregeln für Integration und Migration

Die Notwendigkeit von Abbildungsregeln

Bei der Integration heterogener Datenbestände im Web oder bei der Migration von Datenbanken müssen eindeutige Abbildungsregeln (engl. *mapping rules*) zwischen dem Datenbankschema des Quellsystems und demjenigen des Zielsystems definiert werden; unter Umständen werden mehrere Quellsysteme in ein Zielsystem überführt. Um diese Abbildungsregeln für unterschiedliche Datenmodelle nicht gesondert führen zu müssen, soll das Quellsystem hier mit einem Entitäten-Beziehungsmodell abstrahiert und das Zielsystem als relationales Datenmodell angenommen werden; analoge Abbildungsregeln liessen sich auch für eine XML-Datenbank formulieren.

Eindeutige Beziehung zwischen Quell- und Zielsystem

Wichtig für die Datenintegration wie -migration ist der Grundsatz, dass die Entitätsmengen des Quellsystems auf eindeutige Weise in Tabellen des Zielsystems abgebildet werden und umgekehrt. Diese Eindeutigkeit ist vor allem dann zwingend, wenn bei der Integration oder Migration noch Anwendungen auf den Quellsystemen nachgeführt werden müssen. Aber auch bei der rechnergestützten Konversion bestehender Anwendungsprogramme in solche für das Zielsystem ist die Eindeutigkeit eine wichtige Voraussetzung für eine fehlerfreie und korrekte Datenmigration.

5.3.1 Abbildungen für einfache Entitätsmengen und Wiederholungsgruppen

Um ein Entitäten-Beziehungsmodell in ein relationales Datenbankschema überführen und eine entsprechende Datenintegration oder -migration vorbereiten zu können, lassen sich verschiedene Abbildungstypen für Entitäts- und Beziehungsmengen angeben.

Abbildungstyp 1a überführt Entitätsmengen in Tabellen

Der erste Abbildungstyp aus Abb. 5-3 bezieht sich ausschliesslich auf einzelne Entitätsmengen. Die *Abbildung vom Typ 1a* definiert aus einer Entitätsmenge A eine eigenständige Tabelle A, wobei als (eventuell künstlicher) Schlüssel der Tabelle eine eindeutige und minimale Merkmalskombination dient. Bei allen Abbildungstypen können *jeweils alle oder nur ein Teil der Merkmale* aus den Entitätsmengen des Quellsystems in die Tabellen des Zielsystems übernommen werden. Es ist also nicht zwingend, beim Abbildungstyp 1a alle Merkmale aus der Entitätsmenge A in entsprechende Attribute der Tabelle A zu übertragen.

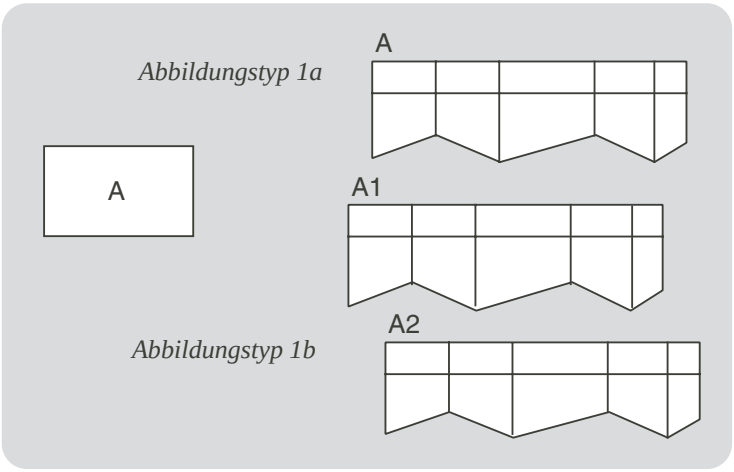


Abb. 5-3 Abbildungsregeln für einfache Entitätsmengen

Abbildungstyp 1b klassifiziert Entitätsmengen

Als zulässige Erweiterung des Abbildungstyps 1a kann eine Entitätsmenge durch ein Selektionsprädikat auf zwei oder auf mehrere Tabellen aufgeteilt werden. So erlaubt die *Abbildung vom Typ 1b* das Überführen der Entitätsmenge A in die beiden Tabellen A1 und A2. Diese Abbildungsregel ist für die Praxis interessant, wenn anhand einer Selektionsbedingung auf der Entitätsmenge A eine Klassenbildung gewünscht wird, um *neue Entitätsmengen A1 und A2 bei einer Integration oder Migration einführen* zu können (Bereinigung von Altlasten).

Beispiel einer Klassifizierung

In Abb. 5-3 werden also die Entitäten der Entitätsmenge A nach einer festen Aufteilungsregel in die beiden Tabellen A1 und A2 übertragen. Würde beispielsweise A der Entitätsmenge MITARBEITER entsprechen, so könnte Tabelle A1 die in Basel wohnhaften und Tabelle A2 die ausserhalb der Stadt wohnenden Mitarbeiter als Teilklassen enthalten. Die beiden Tabellen A1 und A2 würden in diesem Fall die Mitarbeitereinträge in genau zwei Teilklassen unterteilen. Eine Rückspiegelung von den Tabellen A1 und A2 in die herkömmliche Mitarbeiterdatenbank A wäre gewährleistet.

Natürlich ist bei der Abbildung vom Typ 1b nicht zwingend, dass sämtliche Entitäten einen Eintrag in den zugeordneten Teiltabellen aufweisen. Wäre die Tabelle A2 lediglich für Mitarbeiter aus Liestal vorgesehen, so würden die beiden Tabellen A1 und A2 nur die Mitarbeiter aus den Städten Basel und Liestal umfassen, die ausserhalb wohnenden und nicht von der Abbildung erfassten Mitarbeiter wären weiterhin in der herkömmlichen Datenbank enthalten. Dieser unvollständigen Klassenbildung zum Trotz wäre eine Rückspiegelung auf dieser Teilmenge erlaubt und eindeutig.

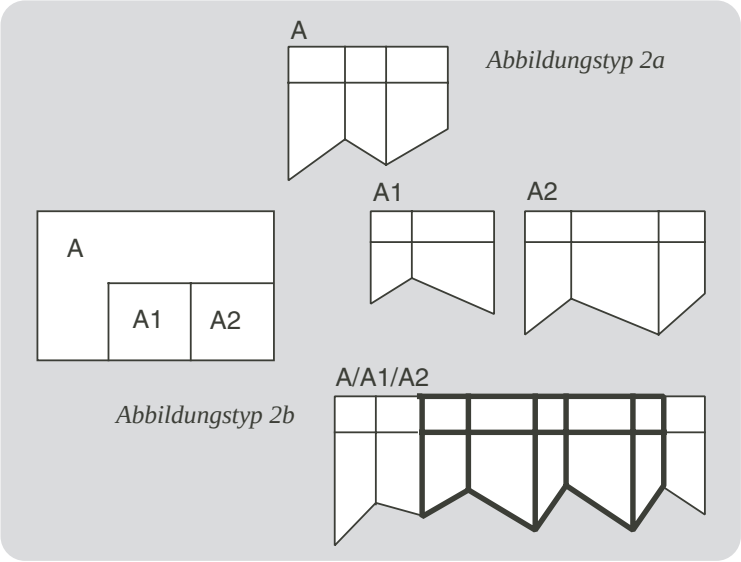


Abb. 5-4 Abbildungsregeln für Wiederholungsgruppen

Abbildungstyp 2a löst Wiederholungsgruppen auf

Wiederholungsgruppen, d.h. Entitätsmengen mit Subentitätsmengen, müssen im Normalfall in einzelne Tabellen aufgelöst werden und umgekehrt. So bringt der *Abbildungstyp 2a* aus Abb. 5-4 die Entitätsmenge A mit den beiden Wiederholungsgruppen A1 und A2 in die drei Tabellen A, A1 und A2. Die Auflösung der Wiederholungsgruppen A1 und A2 in die eigenständigen Tabellen A1 und A2 hängt damit zusammen, weil die erste Normalform nur atomare Merkmalswerte zulässt (vgl. Abschnitt 2.4.2).

Abbildungstyp 2b überführt Wiederholungsgruppen in geschachtelte Tabellen

Neuerdings werden in einem erweiterten Relationenmodell auch Wiederholungsgruppen als Merkmalswerte zugelassen, was zu geschachtelten Tabellen führt. Diese unnormalisierten, d.h. nicht in erster Normalform befindlichen Tabellen werden im Abschnitt 6.4 näher behandelt, hier aber schon zu Abbildungszwecken bei der Integration und Migration verwendet. Der *Abbildungstyp 2b* aus Abb. 5-4 lässt die Entitätsmenge A mit den Wiederholungsgruppen A1 und A2 als eigenständige Tabelle A mit den Untertabellen A1 und A2 bestehen. Solche geschachtelte Tabellen werden zwar nicht von jedem Datenbanksystem unterstützt, setzen sich aber immer mehr durch.

5.3.2 Abbildungen für abhängige Entitätsmengen

In diesem Abschnitt werden Abbildungstypen behandelt, die abhängige Entitätsmengen in abhängige Tabellen überführen und umgekehrt. Eine erste Abbildungsregel ist möglich, wenn zwischen den Entitätsmengen A und B eine (c,1)-Beziehung existiert, d.h. zu jeder Entität in der Entitätsmenge A darf «höchstens eine» Entität in der Entitätsmenge B vorliegen (c=0 oder c=1), umgekehrt gehört zu jeder Entität aus B «genau eine» (1) aus A.

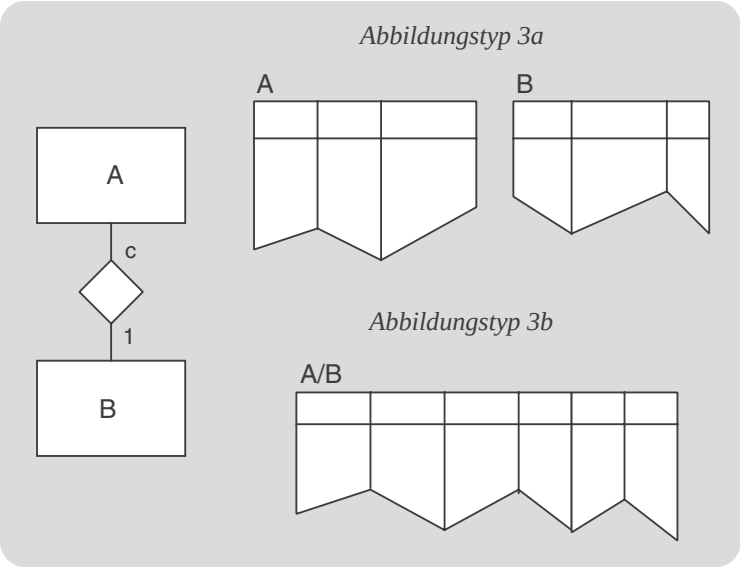
Abbildungstyp 3a generiert aus einer (c,1)-Beziehung eine Tabelle

Hier lassen sich zwei Abbildungstypen unterscheiden: Die *Abbildung vom Typ 3a* aus Abb. 5-5 ist der Normalfall, indem eine (c,1)-Beziehung zwischen den beiden Entitätsmengen A und B durch zwei Tabellen A und B dargestellt wird. Dabei wird in der Tabelle B das Merkmal A# als Fremdschlüssel geführt, um die Referenz auf die abhängige Tabelle A auszudrücken.

Abbildungstyp 3b stellt eine (c,1)-Beziehung in einer Tabelle dar

Als wichtige Ergänzung gilt in Abb. 5-5 die *Abbildung vom Typ 3b*, die die beiden Entitätsmengen in eine einzige Tabelle A/B zusammenfasst. Wie erwähnt, bedeutet der Assoziationstyp c, dass zu jeder Entität in A «höchstens» eine Entität in B vorliegt. Da also nicht jede Entität in A auf eine abhängige Entität in B verweist, können in der Tabelle A/B auch Nullwerte auftreten. Die Abbildung vom Typ 3b ist somit nur sinnvoll, wenn zu den meisten Entitäten in A genau eine Entität in B existiert.

Abb. 5-5 Abbildungsregeln für einfach-abhängige Datensatztypen



Behebung von Altlasten

Typ 3b hilft auch, bestehende Datenbankschemas zu bereinigen. Aus Gründen der Wirtschaftlichkeit hat man nämlich bei Datenbankerweiterungen in herkömmlichen Datenbanken zu einer bestimmten Entitätsmenge trotz Datenmodellierungsgrundsätzen oft ergänzende Entitätsmengen oder neue Zusatzsegmente eingeführt, um die bestehenden Anwendungen bei diesen Schema-Erweiterungen nicht anpassen zu müssen. Mit Hilfe der Abbildung vom Typ 3b lassen sich jedoch zwei (oder auch mehrere) Entitätsmengen bei einer Integration oder Migration entsprechend der Normalformenlehre wieder zusammenfassen (Bereinigung von Altlasten).

Wie werden hierarchische Beziehungen aufgelöst?

Bei der Abbildung vom Typ 4 aus Abb. 5-6 werden zwei Entitätsmengen, die in einer echten hierarchischen (m,1)-Beziehung zueinander stehen, entweder in zwei Tabellen oder in eine Tabelle übertragen. Hier bestehen zu jeder Entität aus der Entitätsmenge C «mehrere» (m=1) Entitäten in der Entitätsmenge D, umgekehrt existiert zu jeder Entität in D «genau eine» (1) Entität in C.

Abbildungstyp 4 überführt eine Hierarchie in eine oder zwei Tabellen

Die *Abbildung vom Typ 4a* überführt die Entitätsmenge C in die Tabelle C und die Entitätsmenge D in die Tabelle D. Die hierarchische Beziehung dazwischen wird durch den Fremdschlüssel C# in der Tabelle D ausgedrückt. Falls das darunterliegende relationale Datenbanksystem auch geschachtelte Tabellen unterstützt, kann die *Abbildung vom Typ 4b* verwendet werden. Solche geschachtelte oder rekursiv definierte Tabellen können als Merkmale wiederum Tabellen aufweisen. Die entsprechende Tabelle C/D ist zwar nicht mehr in erster Normalform, kann jedoch jederzeit durch Projektionsoperatoren in die beim Typ 4a vorliegenden Tabellen C und D zerlegt werden (und umgekehrt).

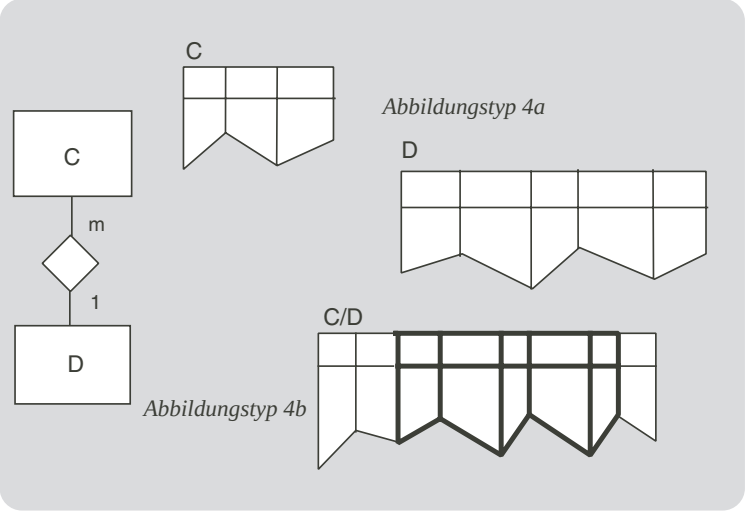


Abb.5-6 Abbildungsregeln für einfach-komplexe Datensatztypen

Abbildungstyp 5 überträgt eine komplex-komplexe Beziehung in drei Tabellen

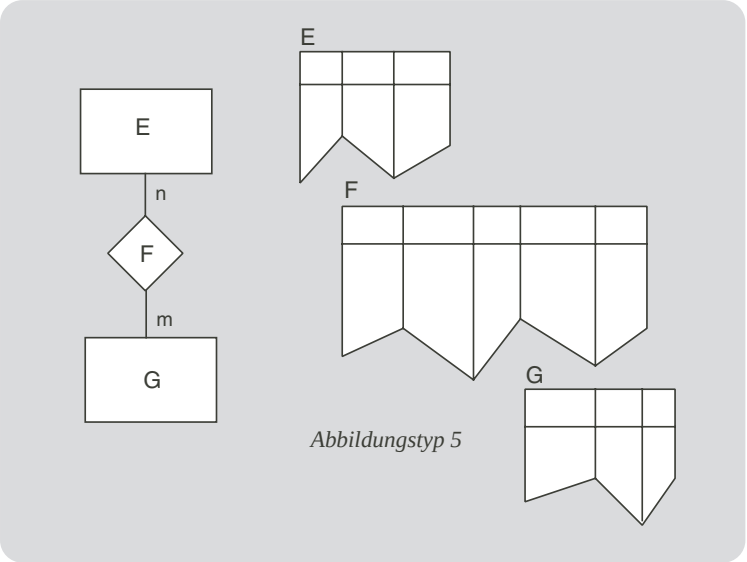
Aufgrund der bekannten Normalformen müssen mehrfach-mehrfache oder (n,m)-Beziehungsmengen (n,m=1) aufgelöst werden, um redundante Tabellen zu vermeiden. Deshalb überführt die *Abbildung vom Typ 5* gemäss Abb. 5-7 die Entitätsmengen E und G in die Tabellen E und G sowie die Beziehungsmenge F in die Tabelle F, möglicherweise ergänzt mit Regeln der referentiellen Integrität.

Neben den Abbildungen vom Typ 1 bis 5 kann man sich weitere vorstellen, die z.B. Generalisationshierarchien oder Aggregationsstrukturen konform in Tabellen überführen. Da die relationalen Datenbanksysteme in der Praxis solche Abstraktionskonzepte noch selten unterstützen, gehen wir auf diese Thematik nicht weiter ein.

Kombinieren von Abbildungstypen ist zugelassen

Die Abbildungen vom Typ 1 bis 5 helfen, auf der logischen Ebene netzwerkartige oder hierarchische Datenstrukturen auf Tabellen abzubilden. Selbstverständlich lassen sich diese Abbildungstypen auch beliebig miteinander kombinieren. Teilweise werden sie durch kommerzielle Reengineering-Tools mit dem Ziel unterstützt, aus herkömmlichen Datenstrukturen relationale herzuleiten. Damit ist im Falle eines Übergangs von nichtrelationalen zu relationalen Datenbanken die Migration noch nicht vollständig durchgeführt, müssen doch auch die dazugehörigen Anwendungsprogramme eventuell neu geschrieben oder zumindest angepasst werden.

Abb. 5-7 Abbildungsregel für komplex-komplexe Datensatztypen



5.3.3 Indirekte Abbildungen für die Datenintegration und -migration

Auf den ersten Blick erscheinen die Abbildungen vom Typ 1 bis 5 zu restriktiv. Weshalb kann bei einer Integration oder Migration nicht einfach eine bestimmte Entitätsmenge auf beliebig viele Tabellen aufgeteilt oder mehrere Entitätsmengen einer herkömmlichen Datenbank in genau eine Tabelle übertragen werden und umgekehrt?

Integration muss Grundsätze der Datenmodellierung respektieren

Der Grund liegt in den Datenmodellierungsgrundsätzen, die in Abschnitt 2.3 ausführlich dargelegt sind. Die dortigen Regeln 1 bis 7 zur Datenmodellierung schreiben vor, wie ein Entitäten-Beziehungsmodell sauber in ein relationales Datenbankschema abgebildet wird. Ähnliche Regeln gelten bei der Integration oder Migration von Datenbeständen. Deshalb ist es beim Abbilden von Datenbanken nicht immer sinnvoll, einzelne Entitätsmengen beliebig auf mehrere Tabellen zu verteilen oder mehrere beliebige Entitätsmengen in genau eine Tabelle abzubilden. Erst beim Auswerten von relationalen Datenbanken steht es frei, auf beliebige Art Informationen zu selektieren oder zu kombinieren.

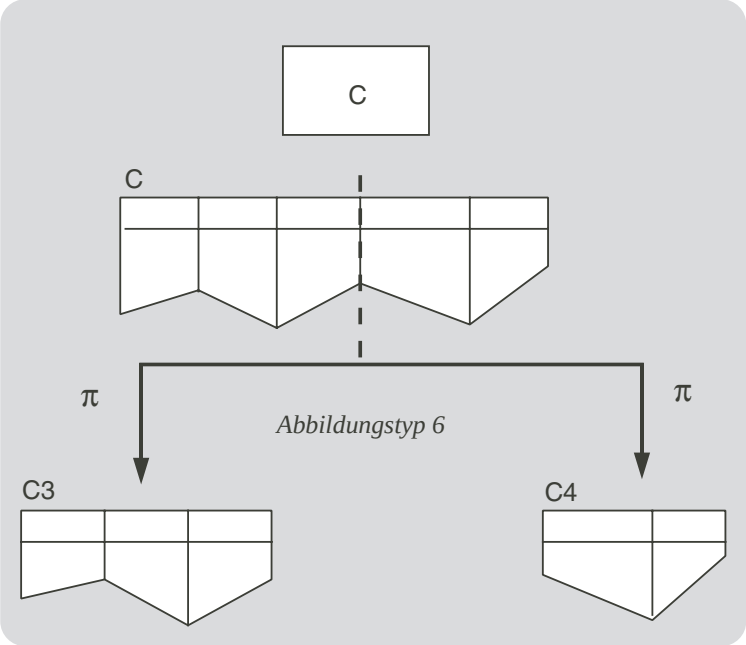
Wege aus dem Datenchaos

Falls die historisch gewachsenen Anwendungssysteme mit ihren heterogenen Datenbanken sauber definiert und implementiert worden sind, lassen sie sich mit den Abbildungen vom Typ 1 bis 5 ohne Schwierigkeiten in relationale Datenbanken überführen. Nun sind aber in der Praxis einzelne Datenbanken oft weniger sorgfältig und isoliert aus Sicht eines einzelnen Anwendungsgebietes heraus aufgebaut worden. So fehlt heute noch vielerorts eine unternehmensweite Datenarchitektur, oder eine solche ist erst im Entstehen begriffen. Wie kann man mit Unzulänglichkeiten umgehen, falls die Abbildungen vom Typ 1 bis 5 nicht genügen?

Abbildungstyp 7 führt unabhängige Entitätsmengen zusammen

Die *Abbildung vom Typ 6* ermöglicht es auf indirekte Art, die Einträge einer bestimmten Entitätsmenge auf verschiedene Tabellen aufzuteilen. Bei diesem in Abb. 5-8 dargestellten Abbildungstyp werden einzelne Merkmale der Entitätsmenge C in die Tabelle C3 und die restlichen in die Tabelle C4 indirekt abgebildet. Bei einer notwendigen Rückspiegelung könnte ein Problem auftauchen: Wenn zu jedem Eintrag in C3 nicht mehr genau ein Eintrag in C4 vorliegt, da beispielsweise in der Zwischenzeit eine Löschoperation gewisse Tupel in C4 entfernt hat, ist eine eindeutige Rückführung der beiden Tabellen C3 und C4 in die Entitätsmenge C nicht mehr gewährleistet.

Abb. 5-8 Indirekte Abbildungsregel mit Projektion



Veränderungsoperation auf Sichten unterliegt Restriktionen

Die Abbildungsregel vom Typ 6 ist *indirekt*, da sie auf indirektem Weg, über eine Zwischenstufe, konfliktfrei realisiert werden kann. Zuerst werden die Entitäten aus C mit Hilfe der Spiegelungsabbildung vom Typ 1a (gegebenenfalls auch vom Typ 1b) in die zugehörige Tabelle C übertragen. Nun steht es dem Administrator einer relationalen Datenbank jederzeit frei, die Tabelle C zum Beispiel mit Projektionsoperatoren auf die Teiltabellen C3 und C4 aufzugliedern, damit dem Benutzer zusätzlich die beiden Sichten C3 und C4 angeboten werden können. Dadurch bleibt die relationale Datenbank konsistent, da Veränderungsoperationen durch die Sichten C3 und C4 gewissen Restriktionen unterworfen sind. Beispielsweise lässt ein relationales Datenbanksystem eine Löschoperation auf der Sicht C4 nicht zu, falls es das Entfernen der zugehörigen Tupel in der Basistabelle C nicht eindeutig nachvollziehen kann.

Abbildungstyp 7 führt unabhängige Entitätsmengen zusammen

Die *Abbildung vom Typ 7* (vgl. Abb. 5-9) ist ebenfalls *indirekt*, da die beiden Entitätsmengen B und T nicht direkt in das Kombinat B/T übertragen werden. Vielmehr werden die beiden Entitätsmengen B und T zunächst mit der Abbildung vom Typ 1a gesondert in die eigenständigen Tabellen B und T übertragen, bevor sie mit einem gemeinsamen Verbundoperator |×| kombiniert werden können. Auch hier gilt, dass ein Verbund zweier Tabellen im Normalfall nur sinnvoll ist, wenn gemeinsame Merkmale sie verbinden. Die gewünschte

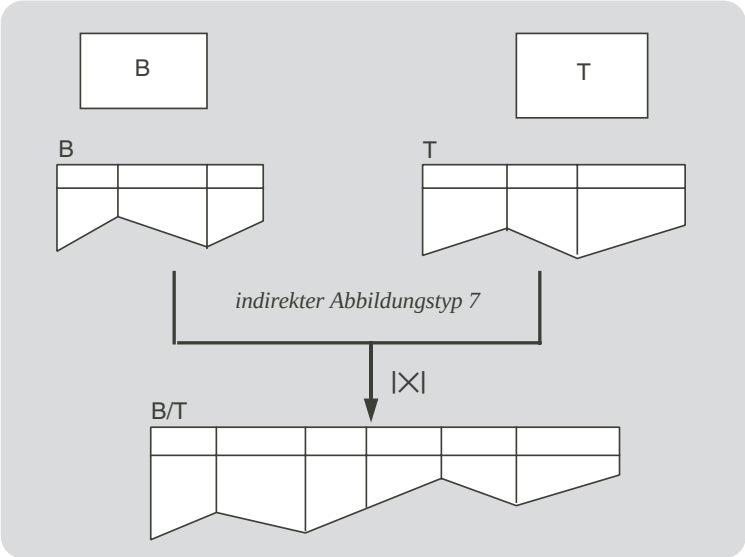


Abb. 5-9 Indirekte Abbildungsregel mit Verbund

Tabelle B/T kann also als Sicht, als Verbund der beiden Tabellen B und T, aufgefasst werden. Die Rückführung der Abbildung vom Typ 7 ermöglicht jederzeit eine Rückführung der Tabellen B und T in die Entitätsmengen B und T.

Vorsicht bei Verbund-abhängigkeit

Bei der Diskussion der Normalformen wurde in Abschnitt 2.4 darauf hingewiesen, dass eine Tabelle mit Projektionsoperatoren nicht in jedem Fall in Teiltabellen zerlegt und durch Verbundoperatoren wieder zurückgewonnen werden kann (vgl. Verbundabhängigkeit). Die fünfte Normalform ist somit ein Grund, für das beliebige Aufteilen und Vereinen von Entitätsmengen indirekte Abbildungsregeln vorauszusetzen, die die direkten ergänzen.

5.4 Migrationsvarianten für heterogene Datenbanken

Wie lassen sich Investitionen schützen?

Obwohl heute viele Unternehmen herkömmliche und relationale Datenbanken gleichzeitig betreiben müssen, bildet ein solches Nebeneinander eine kaum zu bewältigende Schwierigkeit, da *in heterogenen Datenbanken weder die Aktualität noch die Konsistenz der Daten gewährleistet* ist. Erschwerend kommt hinzu, dass unzählige Anwendungsprogramme mit herkömmlicher Datenbanksoftware und unter grossen Investitionen entwickelt worden sind. Wie können nun

Unternehmen mit dieser oft zitierten Erblast längerfristig zurechtkommen?

5.4.1 Charakterisierung unterschiedlicher Migrationsvarianten

Im Hochschulbereich und in vielen Forschungszentren ist der kompatible Übergang von einer Datenbanksystemgeneration zur nächsten selten ein Thema, doch haben sich verschiedene Softwarehäuser und einige Firmen mit umfangreichen Informationssystemen als Pioniere hervorgetan, um ein «Generationenlifting» oder eine Koexistenz unterschiedlicher Datenbanksysteme softwaremässig zu unterstützen.

Grob lassen sich diese Massnahmen gemäss Abb. 5-10 wie folgt charakterisieren:

Migrationsvariante: Daten und Programme konvertieren

- *Daten- und Codekonversion von Anwendungsprogrammen (vgl. Abbildung 5-10, Variante 1):* Herkömmliche Datenbanken werden mit allgemeinen Abbildungsregeln (vgl. 5.3.1 und 5.3.2) auf relationale abgebildet und mit Hilfe von Extrakt- und Lade-Programmen in die entsprechenden Datenbanken übertragen. In einem separaten Arbeitsschritt übersetzt man die prozeduralen Datenbankaufrufe eines Anwendungsprogramms mit Hilfe von Konversionsroutinen weitgehend automatisch in relationale Datenbankaufrufe für das Zielsystem. Bei Spezialfällen oder bei Performance-Problemen müssen diese Konvertierungen von Hand nachgebildet oder abgeändert werden. In gewissen Fällen können Datenmigration und Programmkonvertierungen zum *Wechsel der gesamten Anwendungsseite* führen, so dass im besten Fall auf den Weiterbetrieb des herkömmlichen Datenbanksystems verzichtet werden kann.

Migrationsvariante: Datenbankaufrufe bestehender Programme transformieren

- *Übertragen der prozeduralen Zielsprache auf die gewünschte deskriptive Schnittstelle (Variante 2):* Zu diesem Zweck wird das relationale Datenbanksystem um eine allgemeine Softwareschicht (engl. *wrapper*) ergänzt, die jeden prozeduralen Aufruf in die relationale Abfrage- und Manipulationssprache übersetzt. Die existierenden Anwendungsprogramme werden nicht tangiert, da nur die Datenbankaufrufe softwaremässig auf das relationale Zielsystem umgeformt werden. Konkrete Untersuchungen an herkömmlichen und relationalen Datenbanksystemen haben allerdings – von Performance-Problemen ganz zu schweigen – grundsätzliche Schwierigkeiten aufgedeckt, für jeden prozeduralen Aufruf mit all seinen möglichen Verarbeitungsregeln einen äquivalenten mengenorientierten Aufruf zu finden.

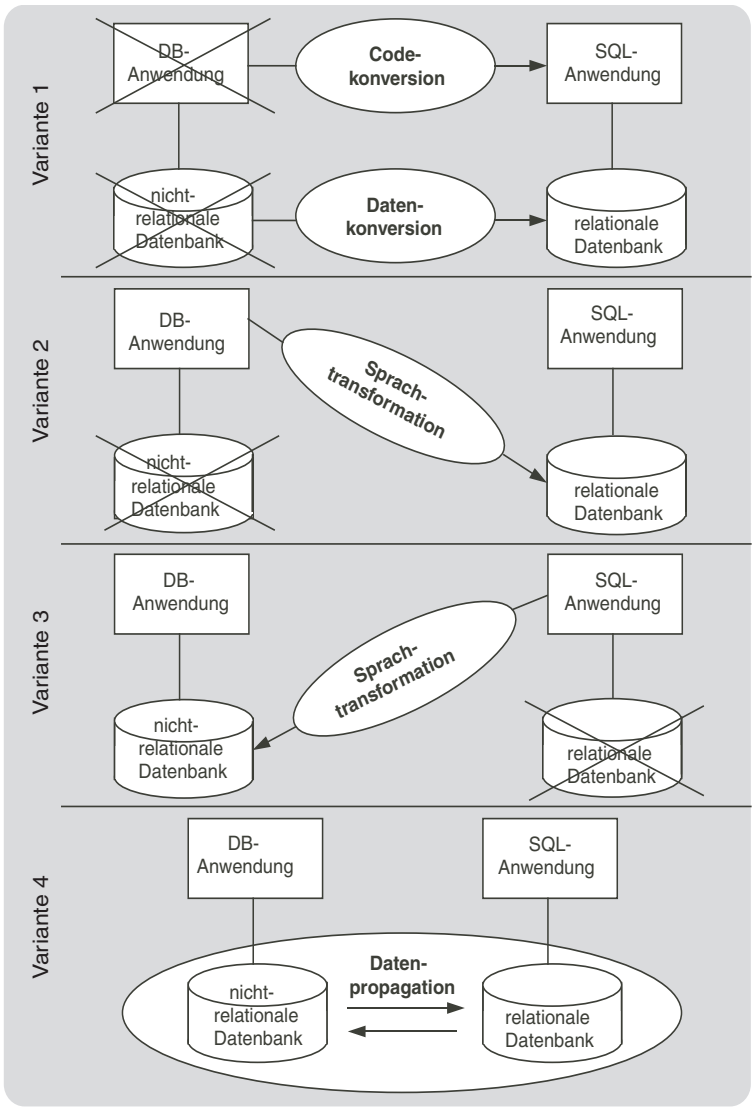


Abb. 5-10 Übersicht über grundlegende Migrations-varianten

- *Rückführen der relationalen Zielsprache auf die gewünschte prozedurale Schnittstelle (Variante 3):* Ein herkömmliches Datenbanksystem wird durch eine generelle Softwareschicht ergänzt, die erlaubt, deskriptive Aufrufe zu formulieren und an das herkömmliche Datenbanksystem zu richten. Dabei werden die mengenorientierten Ausdrücke auf die vorhandene prozedurale Schnittstelle des herkömmlichen Datenbanksystems zurechtgestutzt, was meist mit hohen Performance-Einbussen verbunden

Migrations-variante: SQL-Anwendungen auf herkömmliche Datenbanken ausrichten

ist. Nachteilig wirkt sich zusätzlich aus, dass die Datenhaltung nach wie vor mit dem herkömmlichen Datenbanksystem betrieben werden muss. Bestehende Anwendungen müssen dafür nicht umgeschrieben werden. Neue Anwendungen lassen sich mit einer relationalen Zielsprache und mit entsprechenden Entwicklungswerkzeugen realisieren.

Migrationsvariante: Zeitlich befristete Koexistenz

- *Konsistentes Nachführen paralleler Datenbanken durch Spiegelungsabbildungen (Variante 4):* Lediglich die Änderungen in herkömmlichen Datenbanken werden im relationalen Datenbanksystem gespiegelt. Umgekehrt können Änderungen in relationalen Datenbanken im Bedarfsfall in den Datenbanken des herkömmlichen Systems nachgeführt werden. In beiden Fällen müssen so genannte *Spiegelungsabbildungen* definiert werden, die die Datenbankstrukturen aufeinander beziehen. Eine eventuell befristete *Koexistenz von herkömmlichen und relationalen Datenbanken* (vgl. 5.4.2) macht es möglich, bei Neuentwicklungen aktuelle oder periodisch nachgeführte Datenbestände mit einzubeziehen oder bestehende Anwendungen ohne Termindruck umzuschreiben.

Im Folgenden wird die Koexistenzvariante etwas ausführlicher behandelt, da sie in der Praxis oft verwendet wird und zum Beispiel beim Aufbau eines Data Warehouse (vgl. Abschnitt 6.5) an Bedeutung gewonnen hat.

5.4.2 Systemkonforme Spiegelung von Datenbanken

Vorteile physischer Datenredundanz

Bei der Datenmodellierung gilt zwar die Maxime, dass in Anbetracht latent vorhandener Anomalien oder Konsistenzproblemen redundante Informationen vermieden werden sollten. Ein Verzicht auf Datenredundanz lässt sich aber in der Praxis nicht immer konsequent aufrechterhalten. Beim physischen Entwurf relationaler Datenbanken beispielsweise können sogar unnormalisierte Tabellen von Vorteil sein. Dies zeigen Performance-Überlegungen und die Erkenntnis, dass Tabellen redundanten Inhalts einfachere und effizientere Selektionsoperationen anstelle teurer Verbundoperationen zulassen. Deshalb müssen aus Optimierungsgründen beim physischen Datenbankentwurf oft Kompromisse eingegangen werden.

Koexistenz garantiert systemkontrollierte Datenredundanz

Eine andere Form von Datenredundanz ergibt sich aus dem Wunsch nach Koexistenz unterschiedlicher Datenbanksysteme oder nach längerfristiger Migration in ein relationales Zielsystem. Eine solche *Datenredundanz sollte aus Gründen der Datenkonsistenz systemkontrolliert bleiben,* wie es die Spiegelung zwischen herkömmlichen und relationalen Datenbanken nahelegt.

Die drei Phasen der Koexistenz-variante

Die Spiegelung einer herkömmlichen Datenbank auf eine relationale Datenbank geschieht gemäss Abb. 5-11 in drei Phasen, einer Definitions-, einer Initialisierungs- und einer Überführungsphase:

Spiegelungs-abbildungen im Datenkatalog festhalten

- *Definitionsphase:* Zuerst werden die *Spiegelungsabbildungen* definiert und im Datenkatalog abgelegt. Damit wird festgelegt, welche Datensatztypen in welche Tabellen abgebildet werden sollen. Für direkt abhängige Datensätze einer herkömmlichen Datenbank werden die entsprechenden Tabellen mit einer referentiellen Integritätsregel ausgerüstet.

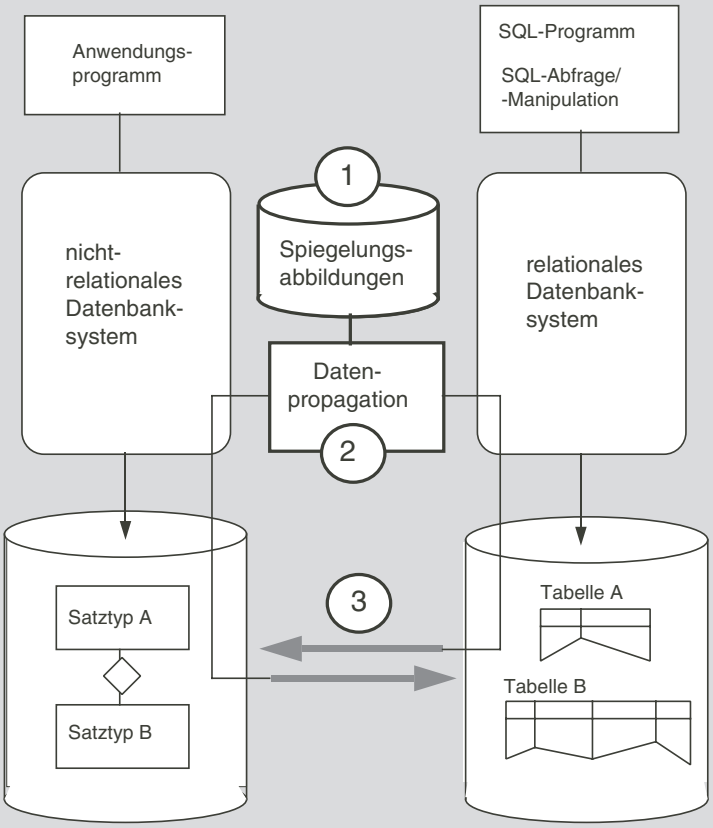


Abb. 5-11 System-kontrollierte Datenredundanz bei der Daten-propagation

Identische Datenbestände als Ausgangsbasis bereitstellen

- *Initialisierungsphase:* Ein Anwendungsprogramm, das eine herkömmliche Datenbank aufruft, muss nicht angepasst oder erweitert werden. Lediglich vor einem erstmaligen Gebrauch der Datenspiegelung müssen bereits existierende Dateninhalte vorab aus der herkömmlichen Datenbank in die entsprechenden Tabellen geladen werden. Damit soll ein *identischer Datenbestand* zwischen den zu spiegelnden herkömmlichen und relationalen Datenbanken als Ausgangsbasis geschaffen werden.

Wertveränderungen synchron oder asynchron nachführen

- *Überführungsphase:* Aufgrund der Spiegelungsabbildungen werden zur Laufzeit sämtliche Änderungsoperationen auf der herkömmlichen Datenbank automatisch durch relationale Aufrufe ergänzt (synchrone Spiegelung), so dass *lediglich Wertveränderungen auf der relationalen Seite nachgeführt* werden müssen. Es ist auch möglich, die Änderungen der herkömmlichen Datenbank zu sammeln und zu bestimmten Zeitpunkten auf der relationalen Seite nachzutragen (asynchrone Spiegelung). Die Spiegelungsabbildungen halten zudem fest, welche Datensatztypen auf die relationale Datenbank gespiegelt und welche Felder innerhalb eines Datensatzes in die entsprechenden Tabellenmerkmale transferiert werden.

Rückspiegelung mindert Risiken und schützt Investitionen

- Die Spiegelung herkömmlicher Datenbanken auf relationale ermöglicht die Verwirklichung neuer Anwendungen mit Hilfe relationaler Datenbanksprachen oder darauf aufbauender Entwicklungswerkzeuge. Falls bestimmte Teile eines Anwendungssystems mit relationaler Technologie neu geschrieben und die Daten nach wie vor auf den herkömmlichen Datenbanken parallel für bestimmte Funktionen bereitgestellt werden müssen, kommt die Rückspiegelungsvariante zum Zuge. Die Rückspiegelung von Änderungen einer relationalen Datenbank auf die entsprechende herkömmliche Datenbank des Quellsystems ist natürlich nur dann notwendig, wenn bereits bestehende Anwendungsprogramme von diesen Datenbeständen abhängig sind. Die *Rückspiegelung bildet somit einen Investitionsschutz* für die zahlreichen Programme, die nicht schlagartig umgeschrieben werden können. Auch aus Risikoüberlegungen heraus ist es vorteilhaft, dass die Koexistenz von herkömmlichen und relationalen Datenbanken ein zeitlich abgestuftes Hinüberwechseln in die relationale Datenbanktechnologie erlaubt.

5.5 Grundsätze der Integrations- und Migrationsplanung

Integration und Migration müssen langfristig geplant werden

Die Nutzung von Webtechnologien, der mögliche Wechsel eines Datenbanksystems sowie die Auswahl einer geeigneten Integrations- und Migrationsvariante bedarf einer exakten Planung, da ein solcher Schritt mit erheblichen Investitionen verbunden ist. Gleichzeitig sollte beim Beheben von Altlasten die Chance genutzt werden, mit den historisch gewachsenen *Unzulänglichkeiten in den Datenbeständen aufzuräumen* und die *Datenarchitektur auf die strategischen Ziele des Unternehmens auszurichten*. Als Grundlage für einen abgestimmten Integrations- und Migrationsplan dient deshalb eine unternehmensweite Datenarchitektur. Diese abstrahiert die strategischen Erfolgspositionen der Geschäftsbereiche und konzentriert sich auf die längerfristig gültigen Informationsbedürfnisse (vgl. 2.6).

Behebung von Altlasten zählt zur Unternehmensverantwortung

Eine Integrations- und Migrationsplanung kann nicht alleinige Aufgabe eines Spezialistenteams sein, vielmehr muss sie im Unternehmen unter den einzelnen Geschäftsbereichen abgesprochen werden, da die Restaurierung bestehender Informationssysteme und ein Wechsel der darunterliegenden Datenbanktechnologie mit grossen Investitionen verbunden ist. So ist es von Vorteil, die wichtigsten Grundsätze bei einem Integrationsprojekt resp. bei einer Datenmigration in Verhaltensregeln festzuhalten, denen in der alltäglichen Entwicklungsarbeit nachgelebt werden kann.

Planungsgrundsätze auf Erfolgspositionen ausrichten

Die folgenden Grundsätze sind von den Bedürfnissen eines international tätigen Unternehmens aus dem Dienstleistungssektor geprägt. Sie können nicht ohne weiteres auf andere Firmen übertragen werden. Sie sollen aber beispielhaft illustrieren, wie eine Integration und Migration von heterogenen Datenbeständen im Rahmen strategischer Erfolgspositionen gesehen und umgesetzt werden kann:

Grundsatz 1

Aufbau einer unternehmensweiten Datenarchitektur
Die Datenintegration und -migration hat im Rahmen der unternehmensweiten Datenarchitektur zu erfolgen.

Technologiewechsel alleine genügt nicht!

Dieser Grundsatz scheint offensichtlich, er ist aber in der Praxis nach wie vor umstritten. Erstaunlicherweise scheut man sich oft vor Aufwand, der der Strukturierung der Daten aus globaler Sicht des Unternehmens dient. Mit einer solchen Haltung wird die Datenintegration und eventuell notwendige Datenmigration auf die technischen Komponenten reduziert. Um nun doch einen Technologiewechsel im Datenbankbereich mit der Erarbeitung einer zukunftsgerichteten Datenarchitektur kombinieren zu können, wird dieser Grundsatz an den Anfang gestellt.

Grundsatz 2

Nutzung relationaler oder postrelationaler Datenbanktechnologie
Neue Anwendungen und neue Geschäftsfunktionen sind mit relationaler und objektrelationaler Datenbanktechnologie zu realisieren.

Systemkomplexität reduzieren und Schnittstellen abbauen!

Die Komplexität bei bestehenden Informationssystemen und bei der Nutzung des Webs verlangen, heterogene Architekturansätze zu vereinheitlichen und die Schnittstellen zu reduzieren. Für die Anwendungsentwicklung macht es deshalb Sinn, verbindliche Grundsätze bezüglich der Datenbanktechnologie und der Benutzung des Webs aufzustellen und durchzusetzen. Die unter stetigem Zeitdruck leidenden Entwicklungsabteilungen stehen sonst zu oft vor Entscheidungsproblemen und folgen der Macht der Gewohnheit mit herkömmlichen Methoden und Techniken.

Grundsatz 3

Verbot für unkontrollierte Extrakte
Datenbestände dürfen nur in bewilligten Ausnahmen auf Extraktbasis den Fachabteilungen zur Verfügung gestellt werden.

Extrakte sind nur im Rahmen einer Datawarehouse-Strategie zugelassen

Das regelmässige Extrahieren von Datenbeständen aus bestehenden Datenbanken ist für ein Dienstleistungsunternehmen äusserst problematisch und sollte nur im Rahmen einer abgestimmten Data Warehouse Strategie (siehe auch 6.5) erfolgen. Ein unkontrolliertes Extrahieren produktiver Datenbestände widerspricht der Zielsetzung, einen *24-Stunden-Betrieb aufrechterhalten* zu können. Hinzu kommt, dass periodisch erstellte Datenbestände immense Abstimmungsprobleme in den Fachbereichen aufwerfen. *Die Datenintegrität ist nur bei aktuellen oder zeitpunktbezogenen Informationen gewährleistet*. Unüberwindbare Schwierigkeiten ergeben sich deshalb, wenn bei Auswertungen durch die Fachabteilungen aktuelle Datenbestände und periodisch extrahierte Datensammlungen kombiniert werden.

Grundsatz 4

Pflicht zu ISO-Normen
Internationale Normen der International Organization for Standardization sind zu berücksichtigen.

Chance nutzen und ISO-Normen einführen!

Bei mehreren tausend Anwendungsprogrammen hat man normalerweise keine Chance, international ausgerichtete Codewerte für Länder, Branchen, Währungen oder Adressen einzuführen, da der Umstellungsaufwand zu gross ist. Ein Wechsel in der Datenbanktechnik resp. die Nutzung von Webtechnologien bietet deshalb die einmalige Gelegenheit, für die neuen Datenstrukturen die ISO-Normen von Anfang an zu verwenden.

Grundsatz 5

Ausrichtung auf Branchendatenmodelle und Online-Datenbanken
Branchendatenmodelle und Online-Datenbanken, die sich auf dem Markt durchsetzen, sind bei der Datenintegration und -migration mit einzubeziehen.

Externe Datenlieferanten und Informationsbroker berücksichtigen

In den letzten Jahren sind für unterschiedliche Anwendungsgebiete kommerziell erhältliche Datenmodelle und Frameworks aufgetaucht. Auf dem Markt zukaufbare Informationen aus Online-Datenbanken wie Wirtschaftsdaten, Marktanalysen und Geschäftsbilanzen ergänzen zudem das Angebot. Wo immer möglich und sinnvoll, sollten diese externen Datenlieferanten bei einer Umstellung der Informationssysteme berücksichtigt werden.

Ein Integrations- und Migrationsplan wird nun anhand der obigen Grundsätze entworfen. Er hat die bestehenden und historisch gewachsenen Datenbestände zu berücksichtigen und umfasst folgende Schritte:

Zielsystem mit Datenarchitektur abstimmen

- *Verfeinern der unternehmensweiten Datenarchitektur*: Diejenigen Entitäts- und Beziehungsmengen, die bei der Integration oder Migration berücksichtigt werden, sollten anhand der unternehmensweiten Datenarchitektur verfeinert und auf ein relationales Datenbankschema abgebildet werden.

Abbildungstypen auswählen

- *Festlegen von Abbildungen zwischen Quell- und Zielsystem*: Das aus der unternehmensweiten Datenarchitektur hergeleitete relationale Datenbankschema für das Zielsystem wird mit den bereits existierenden Datenbanken der Quellsysteme verglichen. Bei einer Abweichung muss geprüft werden, ob und gegebenenfalls welche der zur Verfügung stehenden Abbildungstypen ins gewünschte relationale Datenbankschema des Zielsystems führen.

Eventuell Integrations- und Migrationsvarianten kombinieren

- *Wahl der Integrations- und Migrationsvarianten*: Jetzt kann entschieden werden, welcher Ansatz der Integration oder Migration im jeweiligen Fall geeignet ist. Eventuell ist dies eine Kombination verschiedener Integrations- und Migrationswege. Bei der Koexistenz muss zusätzlich berücksichtigt werden, ob die Spiegelung synchron oder asynchron erfolgen soll. Bei einer synchronen Datenspiegelung liegen sowohl auf der nichtrelationalen wie auf der relationalen Seite aktuelle und konsistent nachgeführte Datenbestände vor. Die asynchrone Spiegelung kommt dann zum Zuge, wenn periodisch übermittelte Änderungen für die relationalen Datenbanken des Zielsystems genügen.

Bei Bedarf Quellsysteme korrigieren

- *Bereinigen von Abbildungskonflikten*: Liegen für einzelne Datensatztypen keine geeigneten Abbildungen vor, so muss entweder eine Bereinigung auf der entsprechenden herkömmlichen Datenbank vorgenommen werden (was eventuell zu Anpassungen bei einigen Anwendungen führen kann) oder es müssen individuelle Abbildungsregeln für diese Datensatztypen vorgesehen werden. Für die auf den Datenbanken des Quellsystems gültigen Verar-

beitungsregeln müssen die entsprechenden referentiellen Integritätsbedingungen auch fürs Zielsystem spezifiziert werden.

Benutzerschnittstelle auf künftige Datenarchitektur ausrichten

- *Definieren externer Schemas*: Können aufgrund der Abbildungstypen die Tabellen der relationalen Datenbank des Zielsystems nicht so ausgelegt werden, wie es die unternehmensweite Datenarchitektur empfiehlt, so sollten mit Hilfe von Sichten (Views) entsprechende externe Schemas definiert werden. Dieses starke Konzept relationaler Datenbanken ermöglicht, an der Benutzerschnittstelle dem Anwender schon frühzeitig die auf die unternehmensweite Datenarchitektur ausgerichtete Datensicht aufzuzeigen.

Durchhaltewillen für Umstellungsarbeiten aufbringen

Diese groben Planungsschritte illustrieren, wie vielfältig ein Integrationsprojekt resp. ein Wechsel in die relationale oder postrelationale Datenbanktechnik ausgestaltet werden kann. So können mehrere Integrations- und Migrationsvarianten miteinander kombiniert werden, und eventuell muss für einzelne leistungskritische Anwendungen auf einen Technologiewechsel ganz verzichtet werden. Auch der Zeitaspekt spielt bei einer Umstellung eine wesentliche Rolle, besitzen doch die Anwendungssysteme normalerweise eine mehrjährige Lebensdauer, bevor sie in grösserem Umfang überarbeitet werden.

5.6 Bibliografische Angaben

Seit kurzem gibt es eine ansehnliche Fachliteratur zu XML und Web-Datenbanken

Seit kurzem sind einige Fachbücher über Web-Datenbanken resp. über XML und Datenbanken entstanden. Loeser (2001) zeigt in seiner Arbeit den Einsatz objektrelationaler Datenbanken für webbasierte Informationssysteme auf. Rahm und Vossen (2003) haben verschiedene Spezialisten auf dem Gebiet Webservices und Datenbanken ermuntert, ihre Kenntnisse in einem Sammelband zu veröffentlichen. Meier und Wüst (2003) haben ein Werk über objektorientierte und objektrelationale Datenbanken verfasst, wobei unterschiedliche postrelationale Datenbanksysteme vorgestellt werden. Kazakos et al. (2002), Klettke und Meyer (2003) sowie Schöning (2003) widmen ihre Werke dem Thema XML und Datenbanken. Darin werden unter anderem die Auszeichnungssprache XML, das XML-Schema sowie die Anfragesprache XQuery behandelt und an Beispielen illustriert.

Forschungsliteratur zum Datenbankwechsel

Zur Abbildung von Datenbankschemas gibt es einige Forschungsarbeiten, die beim Aufkommen relationaler Datenbanksysteme Mitte der 70er Jahre publiziert wurden. Chen (1976) zeigt auf der logischen Ebene, wie das Entitäten-Beziehungsmodell in ein relationales, hierarchisches oder netzwerkartiges Datenbankschema überführt werden

kann. Date (1986) geht in seinem Buch über ausgewählte Themen relationaler Datenbanken auf die Abbildungsproblematik hierarchischer und relationaler Datenbanken ein. Gillenson (1990) stellt Konversionsregeln für physische Datenbanken unterschiedlicher Systeme auf.

Spezifische Arbeiten zur Migration von Datenbanken

Brodie und Stonebraker (1995) beschreiben allgemeine Migrationsstrategien für überalterte Informationssysteme. Meier und Dippold (1992) behandeln die Migration und die Koexistenz heterogener Datenbanken. Den Schutz der Investitionen bei einem Datenbankwechsel illustrieren Meier et al. (1993) und Meier (1997). Das Fachbuch von Dippold et al. (2001) geht neben Aspekten des Datenmanagements grundlegend auf die Datenbankmigration ein. Hüsemann (2002) illustriert in seiner Arbeit die Migration von relationalen zu objektorientierten Datenbanken, eine Methodik für die Datenbankmigration sowie softwaretechnische Unterstützungsmassnahmen.