Inhalt

- Einleitung und Übersicht
- Anforerungserhebung und –analyse
- Koneptueller Entwurf
- Logischer Entwurf
- SQL
- Transakstion und Synchronisationsverfahren

Fahrplan

- März: 9., 16., 23. , 30.
- April: 20., 27,
- Mai: 4., 11., 18., 25.,
- Juni: 8., 15., 22.
- Klausur: 29. Juni 2004

Vorlesung wir gehalten nach einem Lehrbuch:

Datenbanken

Im Unternehmen

G. Pernul und R. Unland

4

Vorlesung Informationssysteme 1
Foliensammlung
Kein Skriptum

- Univ.Prof. Dr. Roland Wagner
- A.Univ.Prof. Dr. Wolfram Wöß
- A.Univ. Prof. Dr. Josef Küng

- Institut für Anwendungsorientierte Wissensverwarbeitung

- Linz, Hagenberg, Wien, Prag

5

LITERATUR

- G.Pernul,R. Unland
  Datenbanken Im Unternehmen; Oldenbourg Verlag; 2001
- A. Kemper, A. Eickler
  Datenbanksysteme Eine Einführung, Oldenbourg Verlag, 1997
- G. Vossen
  Datenmodelle, Datenbanksprachen und Datenbankmanagement-Systeeme, Addison-Wesley, 1994
- A.Heuer, G. Sakke
  Datenbanken- Konzepte und Sprachen, Intern. Tomson, 1996
- P. Dadam
  Verteilte Datenbankn und Client/Server, Springer Verlag, 1996
- S.M. Lang, P.C. Lockemann
  Datenbankeinsatz, Springer Verlag, 1995

6

LITERATUR

- C.J. Date
  An Introduction to Database systems, Addision Weseley, 1994
- R. Elmasri, S.B. Navathe
  Fundamentals of Database Systems, redwood City, 1994
- J.D. Ullman, J. Widom
  A First Course in Database Systems, McGraw Hill Vook, 1997
- A.Silberschatz, H.F. Korth, S. Sudarashan
  Database Concepts, McGraw Hill, 1997
- J. D. Ullman
  Principles of Data and Knowledge Based Systems, Computer Science Press, 1998

7

ALLGEMEINES

Es steht folgende Frage als Ausgangspunkt

„*Wie komme ich von einer Situation der realen Welt zu einer guten Datenbank?*“

Die Vorlesung und Übung sucht nach einer Antwort auf die Frage mit Hilfe der Relationalen Datenbanken

Arbeitsweise von relationalen Datenbanken
Datenbanksprache SQL
Modellierungstechniken

Theorie und Praxis

8

Vorlesung und Übungskonzept

- Vorlesung vermittelt den Stoff
- Übung erarbeitet den Stoff der Vorlesung
- Axiom: Wer alle Übungen erarbeitet hat, hat kein Problem mit der Klausur.

9

Einleitung und Übersicht

Grundlegende Konzepte

Zum Begriff Informationssystem:

Ein System, das Information verarbeitet, d. h. erfasst, überträgt, transformiert, speichert und bereitstellt.

- ISe dienen der Bereitstellung von Informationen für alle Management-Ebenen. Sie sind Teil eines Anwendungssystems das die Informationsbereitstellung betont.

10

- [Mertens] vermeidet den Begriff IS und verwendet statt dessen Administrations- und Dispositions- sowie Planungs- und Kontrollsystem eines Anwendungssystems.
- [Scheer] verwendet den Begriff als Oberbegriff für Administrations-, Dispositions-, Management-Informations und Planungssystem.
- [Hansen] bezieht auch nicht automatisierte Teilsysteme ein.
- [Haerder] verwenden den Begriff Informations- und Kommunikationssystem im Sinne eines Mensch-Aufgabe-Technik-Systems, das automatisierte als auch nichtautomatisierte Teilsysteme einbezieht.

11

Grundlegende Eigenschaften von Datenbanken

Logisch integrierter Datenbestand wird von autorisierten Benutzern mittels DB-Software bearbeitet.

Logisch zusammenhängende Daten
vordefinierte Benutzer, Anwendungen
repräsentiert Realitätsausschnitt (Mini-Welt)

DB-Software unterstützt:

- ➲ Datendefinition
- ➲ Datenspeicherung
- ➲ Autorisierung und Benutzerverwaltung
- ➲ Bearbeitung (Manipulation) von Daten

- Integritätsüberprüfung und Fehlerbehandlung
- formatierte Auswertungen (Reports)
- Systemfunktionen

12

Geschichtliche Entwicklung

1. Generation: vor 1960

- ➲ sequentielle Verarbeitung der Daten
- ➲ primär Rechenoperationen
- ➲ Lochkarten, Magnetbänder

2. Generation: 1960 - 1970

- Entwicklung von Magnetplatten erlaubt den direkten Zugriff auf gespeicherte Informationen
- Beide Generationen können als Dateisysteme bezeichnet werden.

13

3. Generation: ab 1970

- Entwicklung von Datenmodellen zur logischen Beschreibung der Daten
  - Trennung zwischen logischer und physischer Information
  - Unterschiedliche Benutzer können unterschiedliche Sichten auf die Daten haben
  - logische Datenunabhängigkeit
  - physische Datenunabhängigkeit
  - ANSI/X3/SPARC - Architekturvorschlag

14

4. Generation

- Weiterentwicklung von Datenmodellen
- Verteilte Datenbanksysteme
- Multi- Datenbanksysteme
- DBS für Spezialanwendungen

15

Dateisystem vs. Datenbanksystem

Isolierte Dateiverwaltung

Charakteristikum: Jedes Programm verwendet seine eigenen Daten. Die Struktur der Dateien ist im Programm festgelegt.

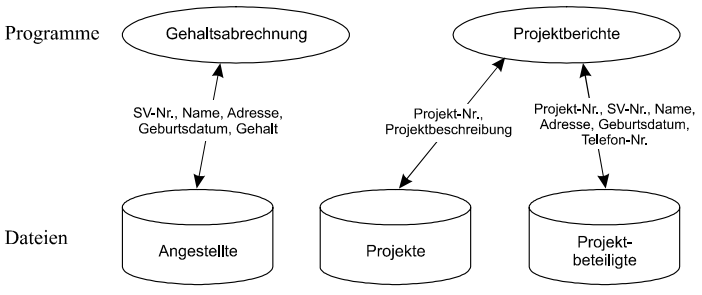
Nachteile:

- Abhängigkeit zwischen Programmlogik und physischer Datenstruktur (physische Datenabhängigkeit).
- Datenstrukturänderung erfordert Programmänderung.
- Daten gleicher Bedeutung werden u.U. mehrfach gespeichert (Redundanz, Gefahr von Inkonsistenz).
- Gleichzeitige Verwendung der Daten durch mehrere Programme nicht möglich.

16

Programm Gehaltsabrechnung:
Datei Angestellte (SV-Nr., Name, Adresse, Geburtsdatum, Gehalt)

Programm Projektberichte:
Datei Projekte (Projekt-Nr., Projektbeschreibung)
Datei Projektbeteiligte (Projekt-Nr., SV-Nr., Name, Adresse, Geburtsdatum, Telefon-Nr.)



Beispiel 1: Isolierte Dateiverwaltung

17

Integrierte Dateiverwaltung

Charakteristikum: Alle Dateien werden an zentraler Stelle gesammelt. Ein Datenkoordinator verwaltet den zentralen Datenbestand mit Hilfe einer allgemein zugänglichen Referenzdatei.Diese enthält Strukturbeschreibungen aller Dateien (Data Dictionary).

Vorteile:

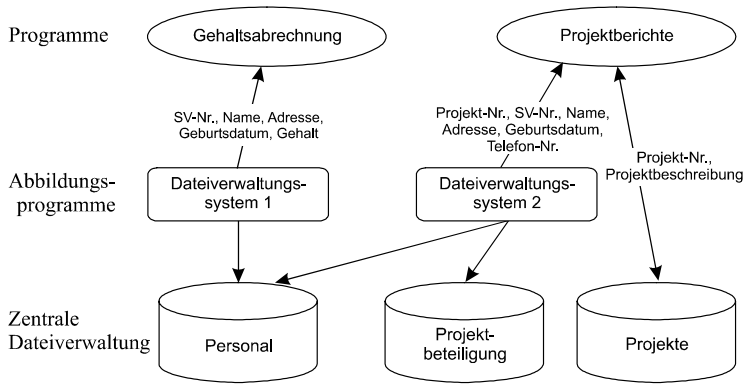
- Konsistenz wird verbessert
- Redundanz wird verringert

Nachteile:

- Änderung einer Datei erfordert Änderung von Programmmen
- aus zentralen Dateien müssen für jedes Programm passende Dateiauszüge erzeugt werden → weiterhin physische Datenabhängigkeit

18

Zentrale Dateien:
Personal (SV-Nr., Name, Adresse, Geburtsdatum, Gehalt, Telefon-Nr.)
Projektbeteiligung (Projekt-Nr., SV-Nr.)
Projekte (Projekt-Nr., Projektbeschreibung)



Beispiel 2: Integrierte Dateiverwaltung

19

Datenbanksystem

Charakteristikum: Trennung der bisherigen Einheit Daten-Programm in zwei separate Schichten. Programme arbeiten mit „logischen Daten“, physische Dateiorganisation für Benutzer nicht sichtbar.

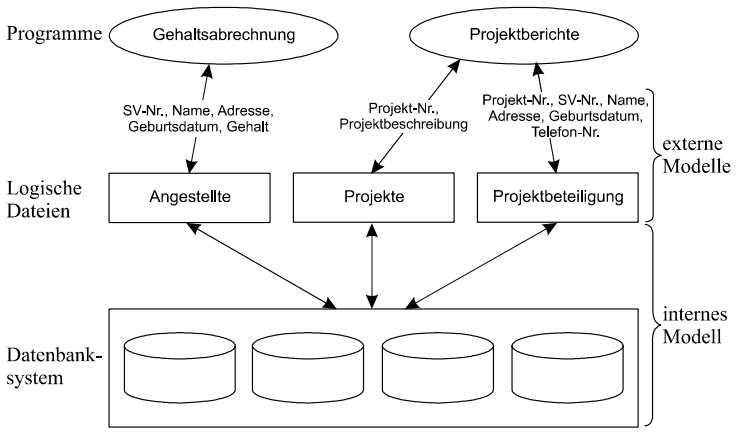
Vorteile:

- Redundanz läßt sich weitgehend vermeiden.
- Sicherung der Datenbankintegrität.
- Hohe Flexibilität wird erreicht.
- Schutz der Daten gegen unerlaubten Zugriff.
- Jedes Programm erhält eigene Sicht der gemeinsamen Daten.
- Belange der physischen Speicherung sind für Anwendungsprogramme irrelevant.

Nachteile:

- Zusatzkosten durch ungenutzte Flexibilität des DBS.
- Höhere Leistung der Hardware erforderlich.

20



Beispiel 3: Externes und internes Modell einer Datenbankverarbeitung

21

Das Drei-Schichten-Konzept für Datenbanksystme

Architektur „früher“ DBS:

Das interne Modell umfasst die logische und physische Beschreibung der Gesamtdatenbank. Eine logische Änderung der Gesamtdatenbank erfordert die Anpassung der darauf zugreifenden Programme.

Einführung einer dritten Schicht

Zwischen externer und interner Schicht wird zur Entkopplung eine konzeptionelle Schicht eingefügt.
Diese Schicht enthält nur die logische Beschreibung der Gesamt-datenbank, die interne Schicht nur die physische Beschreibung.

22

Vorteile:

- Logische Änderung in einem externen Schema wirkt nur auf konzeptionelles Schema.
- Physische Änderung beeinflußt nur Abbildung auf konzeptionelles Schema.

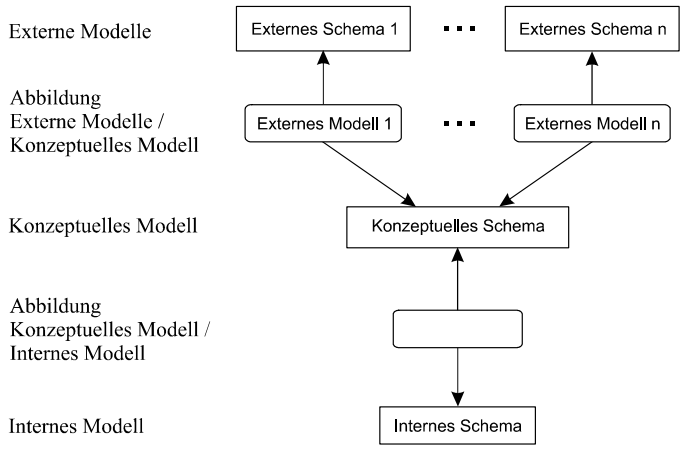
Aufbau

DBMS-Schemata in den drei Schichten:

- Internes Schema: physische Speicherstrukturen und Zugriffspfade
- Konzeptionelles Schema: Struktur und Bedingungen der gesamten Datenbank; führt die externen Sichten aller Benutzer zu einem Gesamtmodell zusammen
- Externe Schemata: beschreiben die verschiedenen Benutzersichten

23

ANSI/SPARC Architekturmodell



24

Charakteristische Eigenschaften von DBS

Persistenz

- Die Daten „überleben“ das Ende von Transaktionen oder Sitzungen.

Disk-Management

- Verwaltung von großen Datenmengen, üblicherweise auf Plattenspeicher.
- Datenbanken sind Ein-/Ausgabe intensiv.
- Spezifische Techniken zur Erhöhung der Performanz (Pufferung, Indexierung, Cluster, Abfrageoptimierung).

Mehrbenutzerbetrieb

- Mehrere Benutzer können gleichzeitig auf den Daten arbeiten.
- DB sorgt dafür, dass keine unerwünschte Wechselwirkungen durch gleichzeitige Manipulation derselben Daten eintreten.
- Erhaltung der Integrität.

25

Zuverlässigkeit der Daten

- Die Daten sind teuer und strategisch wichtig und müssen daher zuverlässig sein.
- DB bestätigt jede durchgeführte Änderung.
- Bei Systemfehler (Zentraleinheit, Hauptspeicher, Platte, Software) kann ein DB-Zustand wiederhergestellt werden, der genau alle bestätigten Änderungen enthält.

Datensicherheit

- Nicht jeder Benutzer darf alles mit allen Daten tun.
- Vergabe und Verwaltung von Zugriffsrechten.

ad hoc Abfragesprachen

- Retrieval von Daten ohne eigenes (prozedurales) Programm zu schreiben.

Wann soll man DB-Systeme einsetzen, wann reichen Dateisysteme aus?

Datenbanksysteme:

- Heterogene Benutzergruppen, gemeinsame Datenhaltung.
- Keine bzw. kontrollierte Redundanz wird gewünscht.
- Ein Benutzerverwaltung, Autorisierung ist Möglich.
- Unterschiedliche Schnittstellen für unterschiedlich geschulte Benutzer.
- Komplexe Beziehungen zwischen Daten.
- Integritätsmechanismen sollen genutzt werden.
- Backup, Fehlerbehandlung.
- Geringe Entwicklungszeit für sich ändernde Anwendungen.

Dateisysteme:

- Es entstehen geringe Investitionskosten.
- Die Anwendungen ändern sich nicht oder nur sehr wenig.
- Overhead für Benutzerverwaltung, Backup, Fehlerbehandlung, Transaktionsverarbeitung (concurrency control) wird nicht benötigt.
- Echtzeitverarbeitung.

28

Personen und Rollen im DB-Bereich

Datenbankadministrator (DBA)
- Verwaltet die Ressource DBS
- Internes Schema
- Vergabe von Zutrittsrechten
- Tuning und monitoring
- Sicherheit und Zuverlässigkeit

Unternehmensadministrator (Datenbankdesigner)
- Konzeptuelles Schema
- Externes Schemata
- Software Entwicklung

Systemanalytiker, Anwendungsprogrammierer
- Anforderungserhebung
- Software Entwicklung

29

Endbenutzer

gelegentlicher Benutzer
- unterschiedliche Info
- komplexe Auswertungen
- „schnell mal nachschauen“

- Manager
- Anfragen nicht vorhersehbar

parametrische Benutzer
- Anwendungsprogramme
- „canned transactions“
- Sachbearbeiter,...

„sophisticated“ Benutzer
- komplexe Anforderungen
- gute Kenntnis von DBS + Schnittstellen

- Techniker, Wissenschaftler, Analytiker,..

DBMS- und Tools-Designer, Implementierer

Operatoren und Wartungspersonal

30

Allgemeine Datenbankbegriffe

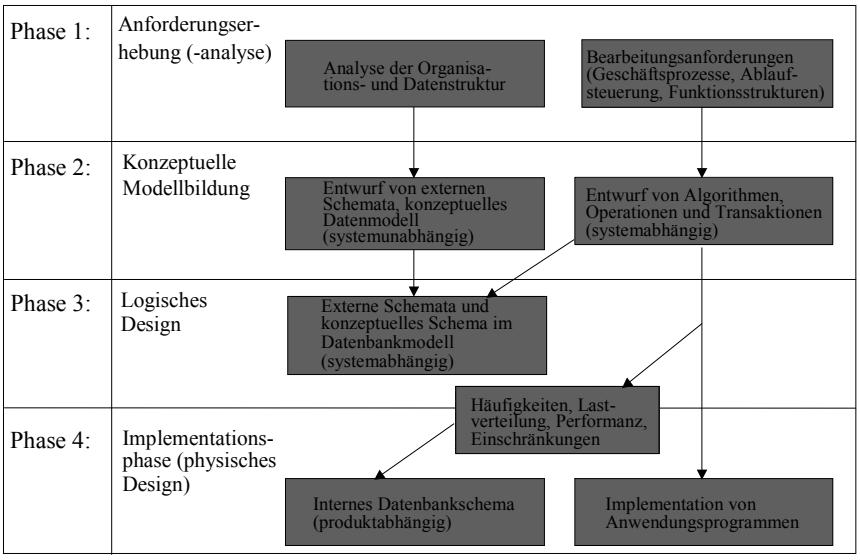
- **Datenmodell:** Eine formale Beschreibung des zu modellierenden Ausschnittes der Realität, wobei die strukturorientierte Betrachtungsweise dominiert.
- **Funktionsmodell:** Ein Funktionsmodell beschreibt eine Datenbankanwendung aus der Sicht der Datenbankfunktionen und der involvierten Prozesse auf unterschiedlichen Abstraktionsniveaus.
- **Integriertes Objektmodell:** Hier findet eine gemeinsame Betrachtung von Daten und der auf Daten ausführbaren Funktionen statt.
- **Datenbank:** Ein logisch integrierter Datenbestand, der unter Verwendung eines Datenbankmanagementsystems verwaltet wird und mit Hilfe eines Datenmodells beschrieben ist.

31

- **Datenbankmodell:** Bestehend aus einer vorgegebenen Menge genau festgelegter Modellierungskonzepte, mit denen ein Datenmodell einer Anwendung erstellt werden kann.
- **Datenbankschema:** Eine maschinell verarbeitbare Formulierung eines Datenmodells.
- **Datenbankmanagementsystem (DBMS):** Eine Sammlung von Programmen zur Erzeugung und Instandhaltung einer Datenbank.
- **Datenbanksystem:** Besteht aus einem DBMS und einer konkreten Datenbank.

32

Phasen des Datenbankentwurfs



33

Anforderungserhebung und -analyse

Ziel:

Die Erstellung eines Anforderungsdokumentes.

- Im Idealfall eine vollständige Spezifikation aller Sachverhalte, die in den Entwurfsprozess einer Datenbank einfließen müssen.
- Kann eine „unternehmensweite“ Sicht der Anforderungen an die Datenbank darstellen.

Voraussetzung für eine erfolgreiche Anforderungserhebung und -analyse ist ein gemeinsames Vorgehen von Systemanalytikern und ausgewählten Nutzervertretern (partizipative Systemanalyse).

34

Gründe für eine partizipative Systemanalyse:

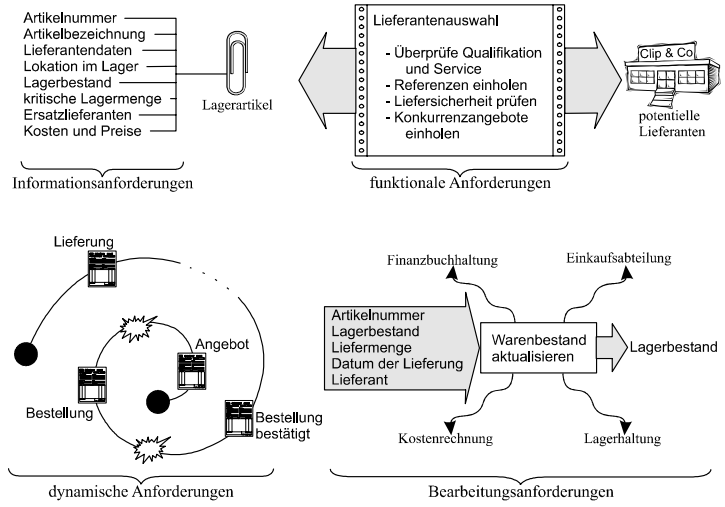
- Der partizipative Systemansatz führt bei den zukünftigen Anwendern zur Akzeptanzsteigerung, da sie unmittelbar in den Entwurfsprozess eingebunden sind und Entwurfsentscheidungen aktiv mitbestimmen können.
- Das Fachwissen der Anwender fließt in Entwurfsentscheidungen ein. Expertenwissen muss nicht extern eingekauft werden.
- Die Kommunikation zwischen Systemanalytikern und Anwendern wird vereinfacht.

35

Anforderungen einer Datenbank können auf verschiedene Art und Weise klassifiziert werden.

Klassifizierung nach der Art der Anforderung:

- Informationsanforderungen
  - statische Informationsbeschreibungen, welche das betreffende Datenbanksystem im späteren Betrieb verwalten wird.
- Funktionale Anforderungen
  - beschreiben betriebliche Vorgänge auf unterschiedlichen Verdichtungsstufen.
- Dynamische Anforderungen
  - repräsentieren den Lebenszyklus von Datenbank-Objekten in Form von Zuständen und Ereignissen.
  - bringen Ereignisse und zustände zueinander in Relation
- Bearbeitunganforderungen
  - beinhalten die verfügbaren Informationen über Transaktionen, die auf den Daten des Datenbanksystems



Beispiel 4: Unterschiedliche Anforderungstypen im Anforderungsdokument

Anforderungserhebung

- Umfasst die quantitative und qualitative Erfassung des Ist-Zustandes eines abgegrenzten betrieblichen Aufgabensystems.
- Es kann zwischen Primär- und Sekundärerhebung unterschieden werden.
- Kann von den betroffenen Benutzern als auch von den Systemanalytikern erstellt werden.

Methoden zur Erhebung der Anforderungen:

- Dokumentenanalyse
- Fragebogentechniken
- Interviews

- Selbstaufschreibung
- Beobachtung
- Berichtsmethode

38

Dokumentenanalyse

- Sofern bereits bestehende Unterlagen korrekt und aktuell sind, stellen sie einen guten Ausgangspunkt dar.
  - Organisationshandbücher, Stellen- und Arbeitsplatzbeschreibungen,..
  - Geschäftsberichte, Bilanzen, Kennzahlen, Statistiken,...
  - Betriebs- und Arbeitsablaufpläne, technische Verfahrensbeschreibungen,..
  - Kunden- und Lieferantenverzeichnisse, Telefonverzeichnisse,...
  - Vorhandene Programme, Programmdokumentationen, Handbücher,...
  - Formulare, ausgefüllte Vordrucke,..
- Sollte zu Beginn der Erhebung stattfinden.
- Liefert rasch eine breite Informationsbasis.
- Stört den Betriebsablauf nicht.
- Die Qualität der erhobenen Informationen hängt stark von der Richtigkeit und Aussagekraft der untersuchten Dokumente ab.

39

Fragebogentechnik

- Fragebogen besteht aus einer Menge von unterschiedlichen Fragen, die von den Befragten schriftlich beantwortet werden.
- Vor der endgültigen Formulierung des Fragebogens empfiehlt sich die Erprobung an mehreren Testpersonen.
- Er sollte nicht zu lang, gut strukturiert und an spezielle Benutzer gerichtet sein.
- Es wird zwischen verschiedenen Fragebogentypen unterschieden:
  - Standardfragebogen oder differenzierter Fragebogen
  - Fragebogen mit offenen bzw. geschlossenen Fragen
  - Individual- oder Gruppenfragebogen
- Leicht statistisch auszuwerten.
- Geringe Kosten.
- Hoher Aufwand der Auswertung von offenen Fragen.

40

Interviewmethode

- Häufigste und ergiebigste Erhebungstechnik.
- Persönliche Befragung potentieller Datenbanknutzer.
- Interviews werden nach folgenden Gesichtspunkten gestaltet:
  - standardisiertes bzw. nicht-standardisiertes Interview
  - Einzel-, Gruppenbefragung bzw. Konferenzmethode
  - offenes bzw. verdecktes Interview

41

- Phasen der Methode
  - Bestimmen der Interviewpartner
  - Erstellen des Fragebogens
  - Durchführung des Interviews
  - Dokumentation der ergebnisse
  - Aufarbeiten des Interviews
- Vertiefung der Befragung durch Zusatzfragen leicht zu ermöglichen.
- Persönlicher Kontakt zwischen Systemanalytikern und potentiellen Datenbanknutzern
- Hoher Zeitaufwand und Störung des Betriebsablaufs.

42

Berichtsmethode

- Wird allein vom Aufnahme-Team ohne Hinzuziehung Dritter vorgenommen.
- Wird oft mit einer Dokumentenanalyse gemeinsam durchgeführt und durch ein standardisiertes Interview ergänzt.
- Geringe Beeinträchtigung des Arbeitsablaufs.
- Hoher Zeitaufwand.

Selbstaufschreibung

- Aufschreibung der Anforderungen durch die zukünftigen Datenbanknutzer.
- Genaue Beschreibung der unterschiedlichen Arbeitsbereiche.
- Die einzelnen Mitarbeiter können ihre Vorstellungen und Lösungsansätze in den Entwurfsprozess sehr gut einbringen.
- Hoher Zeitaufwand und subjektive Darstellung der Mitarbeiter.

43

Beobachtungsmethode

- Aufnahme von Anforderungen unmittelbar an der Quelle ihres Entstehens durch optische Aufnahme.
- Verschiedene Formen der Beobachtungen:
  - offene und verdeckte Beobachtungen
  - strukturierte und unstrukturierte Beobachtung
  - Einzelbeobachtung, Dauerbetrachtung oder unterbrochene Beobachtung
  - direkte und indirekte Beobachtung
- Arbeitsablauf wird nicht beeinträchtigt.
- Sehr zeitaufwendig.

44

Anforderungsdokument

Wünschenswerte Eigenschaften von Anforderungsdokumenten:

- Korrektheit
- Vollständigkeit
- Konsistenz
- Einfachheit
- Eindeutigkeit

Strukturierung hinsichtlich des Anforderungstyps:

- Informationsanforderungen
- Funktionale Anforderungen
- Bearbeitungsanforderungen
- Dynamische Anforderungen

45

Informationsanforderungen

Informationsanforderungen beschreiben die Struktur und die Art der anfallenden Daten und eignen sich daher vorzüglich zur Durchführung einer strukturellen Analyse.

Es sind folgende Angaben von Interesse:

- Allgemeine Beschreibung der Daten
- Integritätsbedingungen
- Referenzen zwischen Daten
- Gültige Wertebereiche
- Minimal- und Maximalkapazitäten der Datenelemente

Informationsanforderungen beschreiben eine zeitunabhängige und damit rein statische Sicht auf die betrachtete Diskurswelt.

46

Analyse der Informationsanforderungen

Drei grundlegende Konzepte:

- Entity (Gegenstand, Entität, Instanz)
- Attribut
- Beziehung

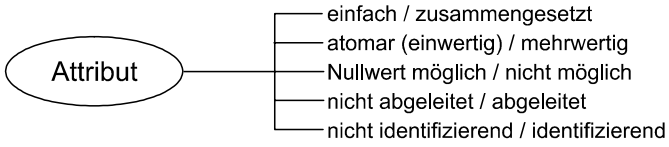
Entity
Ein Entity ist das zentrale Strukturierungskonzept einer Informationsanforderung. Es stellt eine Informationseinheit innerhalb der Diskurswelt dar. Jedes Entity hat beschreibende Eigenschaften, die Attribute genannt werden.

Beispiel: Ein Entity aus der Diskurswelt eines KFZ-Unternehmers ist der Lagerartikel „Zylinderkopfdichtung“ der durch die Merkmale Artikel-Nummer, Artikel-Bezeichnung, Einstandspreis usw. beschrieben wird.

47

Attribut
Attribute sind die kleinsten Informationseinheiten und den Entities zugeordnet.

Strukturierungsmerkmale:



48

Entitytyp

Entities mit ähnlichen Eigenschaften werden zu einem Entitytyp zusammengefasst. Ein Entitytyp besteht aus n unterschiedlichen Attributen während ein Entity dieses Typs durch n Attributwerte beschrieben wird.

Artikel (Artikel-Nr, A-Bezeichnung, Standort, Fahrzeugtyp, ...) } Entitytyp

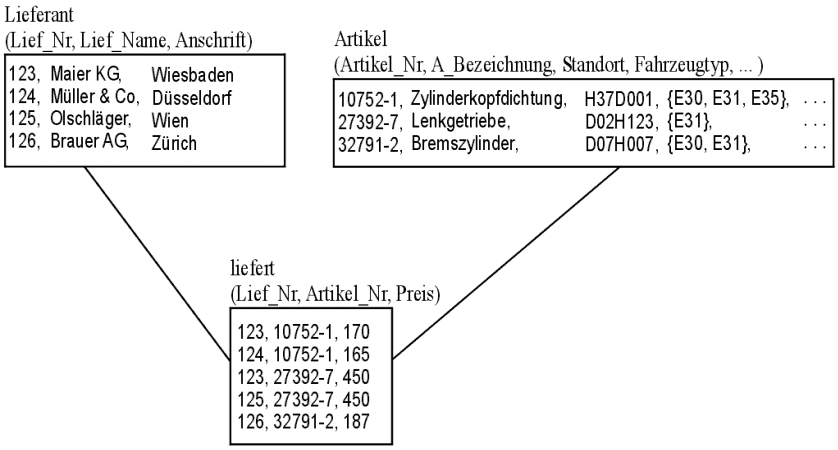
10752-1,	Zylinderkopfdichtung,	H37D001,	{E30, E31, E35},	. . .
27392-7,	Lenkgetriebe,	D02H123,	{E31},	. . .
32791-2,	Bremszylinder,	D07H007,	{E30, E31},	. . .
.				
.				
.				

} Entities

49

Beziehungstyp

Beziehungstypen beschreiben Beziehungen und Zusammenhänge zwischen Entitytypen.



50

Strukturierungsmerkmale für Beziehungstypen

- Grad: Die Anzahl der an der Beziehung beteiligten Entitytypen
- Optional/total
  - Bei einem totalen Beziehungstyp müssen alle Entities eines bestimmten Typs an einer Beziehung teilnehmen.
  - Bei einem optionalen Beziehungstyp ist die Teilnahme an einer Beziehung für Ausprägungen des betroffenen Entitytyps freigestellt.
- Kardinalität: <1:1>, <1:N>, <N:1> oder <N:M> beschreiben die Anzahl von Entities, die maximal an einer Beziehung teilnehmen können. Beispiel für <1:1> ist <Artikelart:Verpackung>.
- Rollen: Jeder Entitytyp, der an einem Beziehungstyp teilnimmt, nimmt an der Beziehung in einer entsprechenden Rolle teil.
- Beziehungstypen können Attribute besitzen.

51

Beispiel : Informationsanforderungen an einen Bestellvorgang

Ein Handelsunternehmen vergibt Bestellaufträge an Lieferanten. Aufträge bestehen aus Auftragspositionen, die wiederum genau einem bestellten Artikel entsprechen. Nach Durchführung der Lieferung schickt der Lieferant eine Rechnung, die unterschiedliche Rechnungspositionen aufweist. In der hier verwendeten vereinfachten Darstellung sind Aufträge durch eine eindeutige Auftragsnummer (Auftrags-Nr) und durch das Auftrags-Datum be-schrieben. Jede Auftragsposition bezeichnet den bestellten Artikel über seine eindeutige Artikel-Nr, die Artikel-Bezeichnung und seinen Preis. Des Weiteren enthält jede Auftragsposition noch die Bestellmenge (Anzahl). Über Lieferanten soll in der Datenbank der Name (Lief-Name), eine fortlaufende Nummer als Identifikator (Lief-Nr), ihre Anschrift und ihre Kontonummer (Konto-Nr) gespeichert werden. Rechnungen haben eine eindeutige Rechnungsnummer (Rech-Nr), weisen einen Endbetrag (Rech-Betrag) und einen Steuersatz auf und geben eine Zahlungsart vor. Jede Rechnungsposition muss einer Auftragsposition entsprechen. Über die bestellten Artikel soll die Artiklenummer (Artikel-Nr), ihre Bezeichnung und der Standort des Artikels im Lager gespeichert werden.

52

Beispiel: Fortsetzung

Kandidaten für Entitytypen:

Auftrag:	Auftrags-Nr, Auftrags-Datum
Auftragsposition:	Artikel-Nr, Artikel-Bezeichnung, Anzahl, Preis
Lieferant:	Lief-Nr, Lief-Name, Anschrift, Konto-Nr
Rechnung:	Rech-Nr, Rech-Betrag, Steuersatz, Zahlungsart
Rechnungspostion:	Artikel-Nr, Artikle-Bezeichnung, Anzahl, Preis
Artikel:	Artikel-Nr, Artikel-Bezeichnung, Standort

Kandidaten für Beziehungstypen:

„Geht_an“	Auftrag : Lieferant, <N:1>
„Enthält“	Auftrag :Auftragsposition, <1:N>
„Verweist_auf“	Auftragsposition : Artikel, <N:1>
„Legt“	Lieferant : Rechnung, <1:N>
„Besteht aus“	Rechnung : Rechnunsgposition, <1:N>
„Entspricht“	Rechnungsposition : Auftragsposition, <1:1>

53

Vorgehensweise für die statische Analyse

- Top-down-Strategie: Geht von im Problemraum erkannten Entitytypen aus, denen Attribute und Beziehungstypen hinzugefügt werden.

- Bottom-up-Strategie: Geht von bekannten Informationseinheiten aus, aus denen dann Entitytypen und Beziehungstypen gebildet werden.

54

Funktionale Anforderungen

Sie spezifizieren die betrieblichen Funktionen, die ein System oder eine Systemkomponente erfüllen bzw. zur Verfügung stellen muss.

Betriebliche Vorgänge werden algorithmisch beschrieben.

Funktionale Anforderungen beschäftigen sich mit Fragen wie:

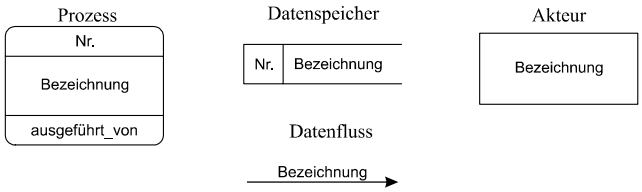
„Was leistet ein Prozess?“ bzw. „Wie erreicht ein Prozess sein Ziel?“

Die meisten Verfahren zur Analyse funktionaler Anforderungen unterstützen unterschiedliche Abstraktionsebenen.

55

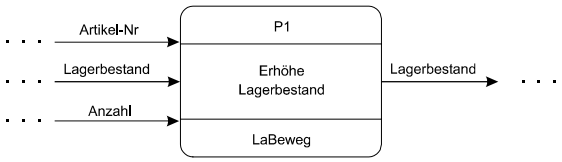
Analyse mit Datenflussdiagrammen

- DFD bestehen aus vier grundlegenden graphischen Konzepten:
  - Prozesse, die Daten transformieren
  - Datenflüsse, die Daten bewegen
  - Datenspeicher, die anzeigen, dass Daten auf Externspeicher abgelegt werden und
  - Akteure, die Daten generieren und konsumieren.

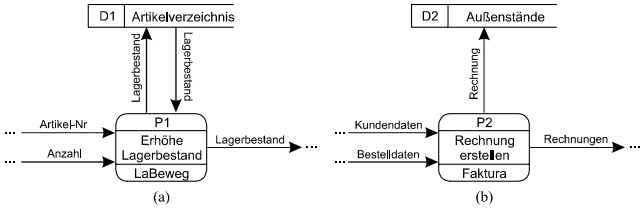


56

Beispiel: Prozess im Datenflussdiagramm

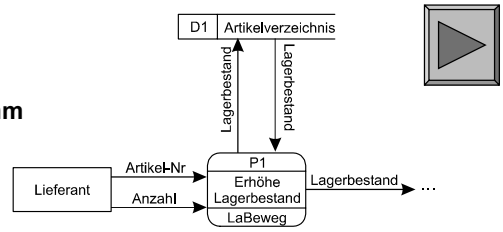


Beispiel: Datenspeicher im Datenflussdiagramm

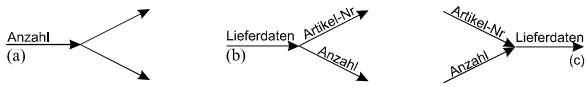


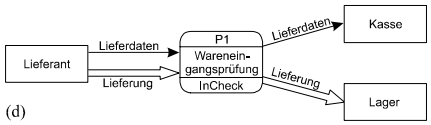
57

Beispiel:
Akteure im Datenflussdiagramm



Beispiel:
Formen von Wertfluss
im Datenflussdiagramm





58

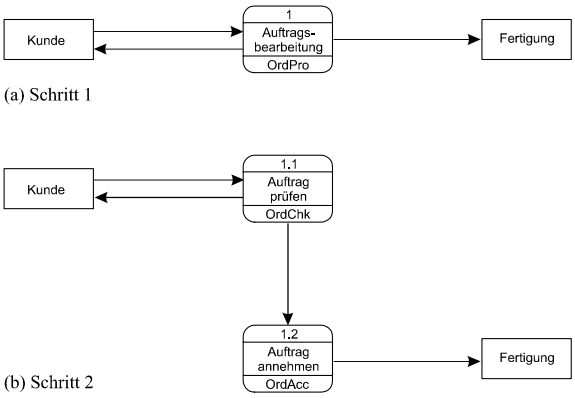
Vorgehensweise für die funktionale Analyse

Top-down-Vorgehensweise

- Die am häufigsten zur Anwendung kommende Vorgehensweise.
- Man beginnt mit einem Kontextdiagramm
  - Dieses zeigt die betriebliche Aufgabenstellung auf einem hohen Abstraktionsniveau und
  - besteht aus einem zentralen Prozess und seinen Schnittstellen nach außen.
- In weiteren Schritten wird der Elementarprozess in Teilprozesse zerlegt.
- Es wird solange verfeinert, bis der gewünschte Abstraktionsgrad erreicht bzw. keine Zerlegung mehr möglich ist.

59

Beispiel: Top-Down-Vorgehensweise



60

Mischform aus Top-down und Bottom-up:

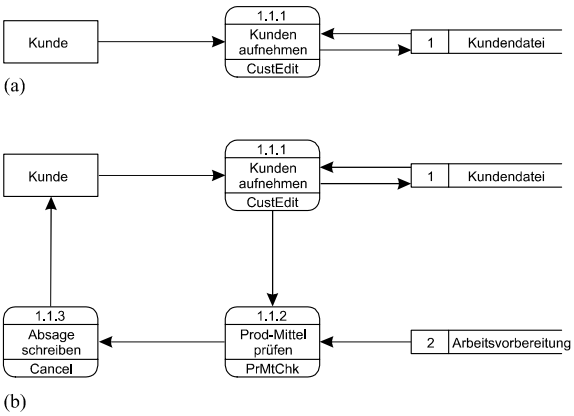
- In der Praxis wird die Top-down und Bottom-up Vorgehensweise oft kombiniert.
- Es ist dem Entwickler überlassen mit welcher Vorgehensweise er beginnt.
- Da der Wechsel zwischen den Methoden zum richtigen Zeitpunkt durchgeführt werden muss, setzt die Mischform große Erfahrung voraus.

Inside-out-Vorgehensweise

- Versuch ein Problem direkt zu lösen.
- Zur Analyse kleinerer Anwendungen geeignet.

61

Beispiel: Inside-out-Vorgehensweise



62

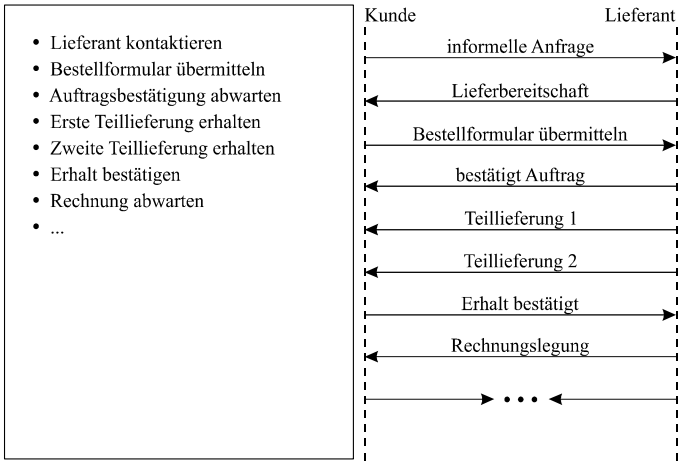
Bearbeitungsanforderungen und dynamische Aspekte

Dynamische Anforderungen

Sie zeigen das zeitabhängige Verhalten von Datenbankobjekten. Informationseinheiten können im Laufe ihrer Existenz verschiedene Werte annehmen. Zustandsveränderungen werden durch das Eintreten vordefinierter Ereignisse verursacht. Die Reaktion auf ein Ereignis hängt vom aktuellen Zustand der Informationseinheit ab.
Der Zyklus der Zustandsänderungen einer Informationseinheit wird als Lebenszyklus bezeichnet.
Es werden Ereignisfolgen identifiziert (Szenarios).
Dann werden die beteiligten Objekte identifiziert und in einem Ereignisabfolgediagramm dargestellt.

63

Beispiel: Szenario und Ereignisabfolgediagramm „Bestellvorgang“



Szenario für Bestellvorgang

Ereignisabfolgediagramm für Bestellvorgang

Bearbeitungsanforderungen

Bearbeitungsanforderungen werden während der Analyse der geplanten Transaktionen untersucht. Ziel ist es systembedingte Eigenschaften der Transaktionen (Bearbeitungsanforderungen) genau darzustellen.

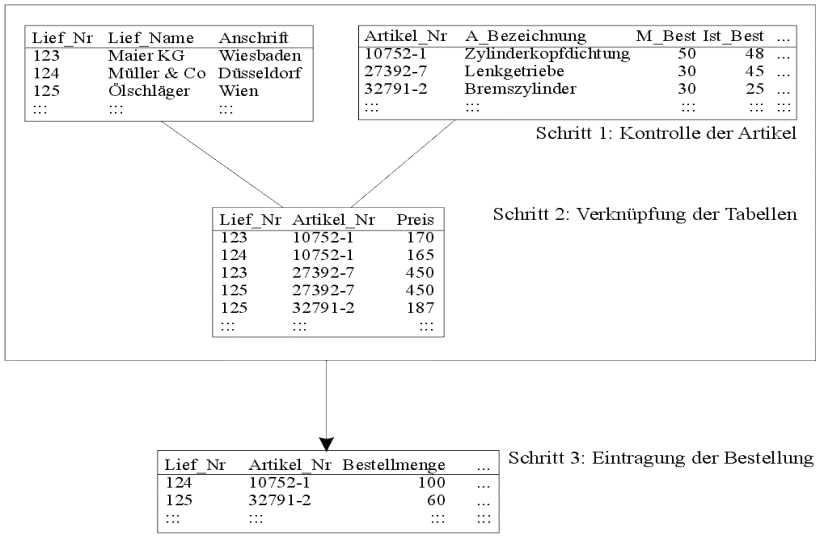
Transaktionsanalyse

- Unterscheidung der Transaktionen in Datenbankanfragen und Datenbankmanipulationen.
- Datenbankanfragen beinhalten ausschließlich die Ausgabe von Informationen.
- Eine Datenbankmanipulation ändert den Inhalt der Datenbank durch eines oder mehrere der folgenden Operationstypen:
  - delete (Datensätze werden aus der Datenbank entfernt)
  - insert (Datensätze werden in die Datenbank eingefügt)
  - update (Datensätze werden verändert)

- Eigenschaften der wichtigsten Transaktionen
  1. Transaktionstyp (delete, insert, update oder query)
  2. Erwartete Frequenz der Transaktion (z.B. Fünfmal täglich)
  3. Zeitbeschränkung (z.B. max. Laufzeit 10 Sekunden)
  4. Informationseinheiten, die von der Transaktion berührt werden
  5. Attribute, die zur Auswahl der Datensätze (Selektion) benutzt werden
  6. Attribute, deren Werte zur Verknüpfung von Informationseinheiten verwendet werden
  7. Attribute, deren Werte durch eine Datenmanipulation verändert werden

In der Praxis ist es meist nicht notwendig, alle Transaktionen bis ins kleinste Detail vorauszuplanen, da es nur wenige Transaktionen sind die die meiste Bearbeitungszeit beanspruchen. („80-20 Regel“)

66



Beispiel 14: Transaktion „Bestellungen erstellen“

67

Beschreibung:	Überprüfung des Lagerbestandes und Erstellen der Bestellungen
Transaktionstyp:	Insert
Häufigkeit:	Einmal täglich
Zeitbeschränkung:	keine (erfolgt nach Geschäftsschluss automatisch)

Informationseinheit	**Attribut**	**Verknüpfung**	**Selektion**	**Manipulation**
Lieferant	Lief_Nr	X		
liefert	Lief_Nr	X		
	Artikel_Nr	X		
Artikel	Artikel_Nr	X		
	Ist_Best		X	
	M_Best		X	
Bestellung	Lief_Nr			X
	Artikel_Nr			X
	Bestellmenge			X

Beispiel: Anforderungsformular der Transaktion „Bestellungen erstellen“

68

Konzeptueller Entwurf

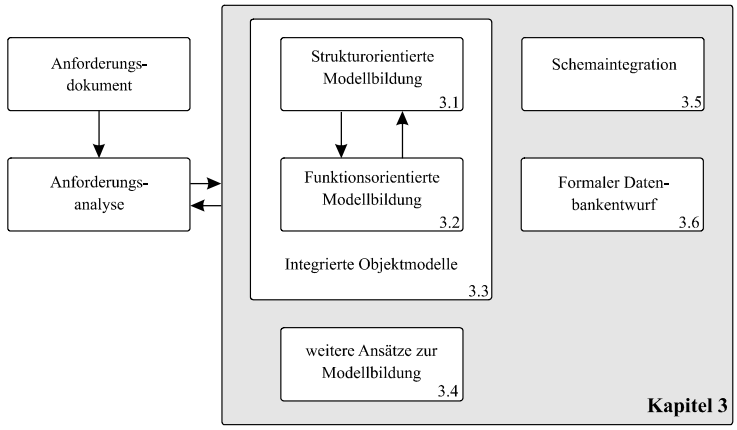
Die Hauptaufgabe dieser Entwurfsphase ist es, alle individuellen Anforderungen aus den Anforderungsdokumenten in einer einheitlichen Spezifikation, dem unternehmensweiten Datenmodell, zu repräsentieren und damit für alle Anwendungen eine einheitliche Schnittstelle für den Zugriff auf die DB zu schaffen.

Eine DB muss den aus der Sicht der zu bedienenden Anwenungsklassen relevanten Realweltausschnitt abdecken.

Man braucht Ausdrucksmittel, mit denen die Realwelt so vereinfacht dargestellt werden kann,

- dass die resultierende Darstellung abstrakt genug ist, um unwichtige Details kaschieren zu können,
- aber ausdrucksstark genug ist, um alle erwünschten Merkmale und Beziehungen auch ausdrücken zu können.

69



Ansätze zur konzeptuellen Modellbildung

Grundsätze ordnungsgemäßer Modellbildung

Grundsatz der Richtigkeit

Die *syntaktische* Richtigkeit wird eingehalten, wenn ein Modell den Notationsregeln der gewählten Modellierungsmethode entspricht.

Die *semantische* Richtigkeit bewertet den „Inhalt“ des Modells, also inwieweit das Modell dem entsprechenden abzubildenden Real-weltausschnitt entspricht.

Grundsatz der Relevanz

Das Modell sollte keine überflüssigen Informationen enthalten noch sollten relevante Informationen fehlen.

Grundsatz der Wirtschaftlichkeit

Dieser Grundsatz stellt den betriebswirtschaftlichen Wirtschaftlichkeitsaspekt dar. Die Wirtschaftlichkeit lässt sich nur schwer bewerten, weil sich Kosteneinsparungen am Modell erst in den späteren Phasen des Entwurfs rächen.

Grundsatz der Klarheit

In diesem Grundsatz werden die Aspekte der Strukturiertheit, der Übersichtlichkeit und der Lesbarkeit zusammengefasst.

72

Grundsatz der Vergleichbarkeit

Die syntaktische Vergleichbarkeit stellt die Kompatibilität von Modellen dar, die mit verschiedenen Modellierungstechniken erstellt wurden.

Die semantische Vergleichbarkeit beschreibt die inhaltliche Vergleichbarkeit von Modellen.

Grundsatz des systematischen Aufbaus

Bei der Erstellung eines Modells sollte eine systematische Vorgehensweise gewählt werden.

73

Strukturorientierte Modellbildung

Bei der strukturorientierten (datenorientierten) Modellbildung stehen die Daten und ihre Beschreibung in einem semantischen Datenmodell im Mittelpunkt der Betrachtung.
Semantische Datenmodelle eignen sich hervorragend zur Analyse der im Anforderungsdokument enthaltenen statischen Informationsanforderungen.

Die Modelle beschreiben die Welt als eine Menge von

- Systemkomponenten (reale oder abstrakte Gegenstände),
- Eigenschaften von Systemkomponenten und
- unterschiedliche Formen von Beziehungen zwischen Systemkomponenten.

Sie verwenden Abstraktionsprinzipien um elementare Eigenschaften der Realität zu aggregieren und zu Informationseinheiten eines höheren Abstraktionsniveaus zusammenzufassen.

74

Abstraktionsprinzipien

Klassifikation
Objekte mit ähnlichen Eigenschaften werden als Instanzen einer Objektklasse betrachtet. Ein Objekttyp besitzt alle Eigenschaften, die die Instanzen der Objektklasse beschreiben.

Assoziationen
Objekte bzw. Objekttypen können miteinander in Beziehung stehen. Beziehungen können selbst wieder Objekte sein.

Aggregation
Aggregation kann es auf verschiedensten Hierarchieebenen eines semantischen Datenmodells geben. Sie beschreibt, wie bestimmte Informationseinheiten zu einem Objekt einer höheren Informationseinheit zusammengefasst werden können.

Generalisierung
Ähnliche Objekte werden aufgrund unterschiedlicher Eigenschaften zu Spezialisierungen zusammengefasst.

75

Grundlagen des Entity Relationchip-Modells

Das Entity Relationchip-Modell (ERM) von Peter P.Chen ist das bekannteste semantische Datenmodell. In seiner Urform unterstützt es ausschließlich die Konzepte Entity, Beziehung und Attribut.

Im ERM werden gleichartige Entities zu einem Entitytyp und gleichartige Beziehungen zu einem Beziehungstyp zusammengefasst.

Die Struktur der Informationsanforderungen eines betrachteten Realitätsausschnittes kann durch Herleiten von Entitytypen, Beziehungstypen und Attributen durch ein ERM sehr gut analysiert und durch ein Entity-Relationchip-Diagramm (ERD) visualisiert werden.

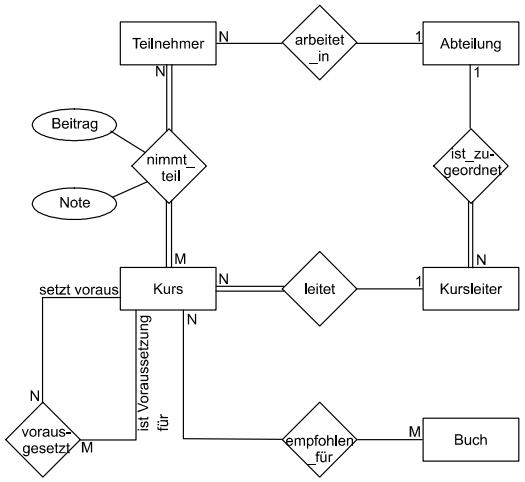
76

Beispiel: ERD einer Kursverwaltung

Relevante Entitytypen sind Kurs, Kursleiter, Teilnehmer, Buch und Abteilung, die miteinander in einem semantischen Zusammenhang stehen. Alle Kurse müssen einen Kursleiter haben, und ein Leiter kann mehrere Kurse abhalten. Da in dem ERM kein Zeitbezug dargestellt ist, können mehrere Kurse eines bestimmten Leiters zum selben Zeitpunkt (z.B.im gleichen Semester) stattfinden. Außerdem lässt die Beziehung „leitet“ auch Kursleiter zu, die keinen Kurs anbieten. Für bestimmte Kurse sind möglicherweise andere Kurse Voraussetzung, und jeder Kurs kann wiederum Voraussetzung für andere Kurse sein. Für bestimmte Kurse wird das Studium gewisser Bücher empfohlen. Ein bestimmtes Buch kann möglicherweise für mehrere Kurse genutzt werden. Kursleiter müssen, Teilnehmer an Kursen können Abteilungen zugeordnet sein. Kurse können nur stattfinden, wenn sie zugeordnete Teilnehmer haben. Die Beziehung „nimmt_teil“ besitzt die qualifizierenden Attribute Beitrag und Note. Kursteilnehmer können an mehreren Kursen teilnehmen.

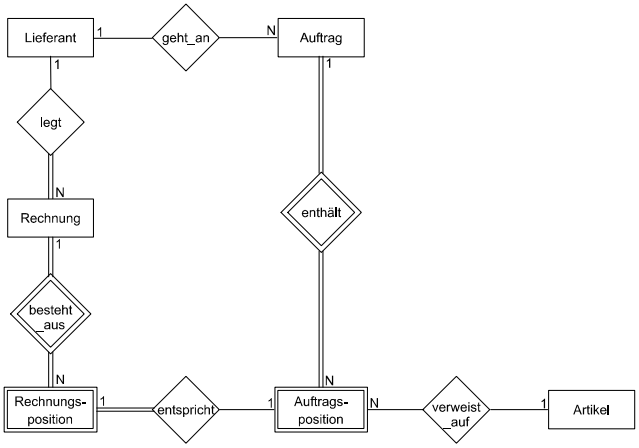
77

Beispiel: ERD einer Kursverwaltung (Fortsetzung)

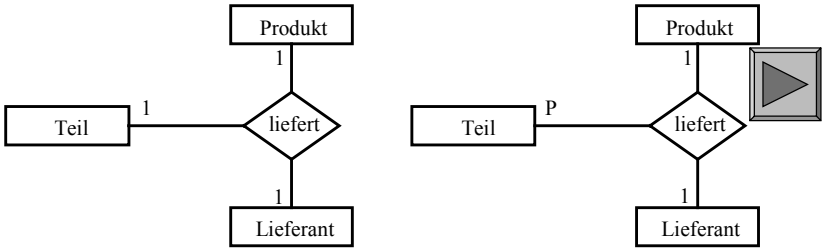


78

Beispiel: Bestellvorgang (siehe Bsp. Weiter oben)

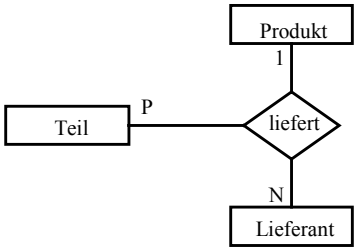


79

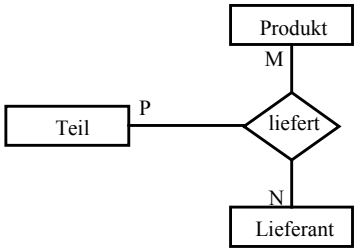


(a) <1 : 1 : 1>-Beziehungstyp

(b) <1 : 1 : P>-Beziehungstyp



(c) <1 : N : P>-Beziehungstyp



(d) <M : N : P>-Beziehungstyp

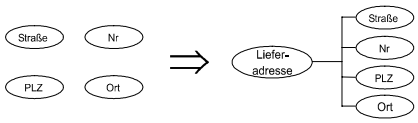
Beispiele dreistellige Beziehungstypen

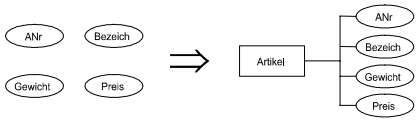
80

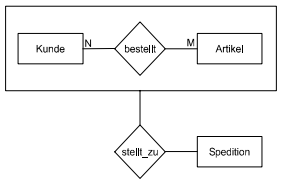
Weiterentwicklungen des ERM

Aggregation

Die Aggregation wird verwendet, wenn durch Zusammenfassen bestehender Konzepte ein neues Objekt eines höheren Abstraktionsniveaus erzeugt werden soll. Aggregation gibt es auf unterschiedlichstem Niveau.





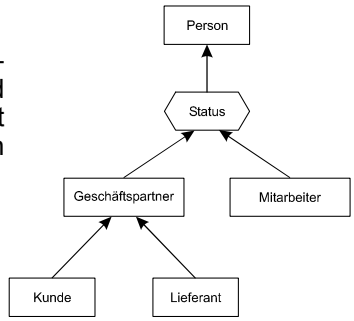


81

Generalisierung

Bei der Generalisierung werden Entities eines Typs aufgrund gemeinsamer Merkmalsausprägungen in Unterklassen (Sub-Klassen) zusammengefasst.

Sie definiert eine IS-A-Beziehung (ist-vom-Typ) zwischen Spezialisierung und generischem Objekttyp und impliziert eine Vererbung von Attributen von Super-Klassen zu Sub-Klassen.



Formen der Generalisierung:

- Generalisierunghierarchie
  Die Entitymengen der Spezialisierungen müssen disjunkt sein.
- Subtypenhierarchie
  Spezialisierungen können überlappen

Formaler Datenbankentwurf

Ziel:
Ziel des formalen Datenbankentwurfs ist es, durch sukzessive Transformation an einem Ausgangsdatenbankschema zu einem „optimalen“ logischen Datenmodell zu gelangen.

Eigenschaften einer Datenbank

- Der Inhalt der DB soll den abgebildeten Realitätsausschnitt wahrheitsgetreu widerspiegeln.
- Die Integrität der Datenbasis muss gewährleistet sein.
- Kontrolle der Datenredundanzen und Vermeidung unerwünschter Mehrfachspeicherung.

Das relationale Datenbankmodell

Bei der Anwendund des formalen Datenbankentwurfs wird das relationale Datenmodell als Ausgangsmodell und als Zielmodell des Datenbankentwurfs verwendet.

Vorteile:

- Das Modell verfügt über sehr einfache Grundlagen, die es erlauben, dem Nutzer eine übersichtliche und doch sehr mächtige Schnittstelle zur Verfügung zu stellen.
- Das Modell basiert aus formal gut zu analysierenden Konzepten.

84

Informelle Begriffe

Im Relationenmodell werden Datenbestände den Nutzern in Form von *Tabellen* präsentiert.

Eine *Tabelle* entspricht der Darstellung eines Objekttyps der Realität.

Die *Spalten* der Tabelle repräsentieren die Attribute, und jede *Zeile* der Tabelle entspricht einer Instanz des Objekttyps.

Die Reihenfolge von Zeilen und Spalten in Tabellen einer relationalen DB sind nicht von Bedeutung.

85

Formelle Begriffe

Ein relationales Datenbankschema besteht aus einer Menge von Relationenschemata.

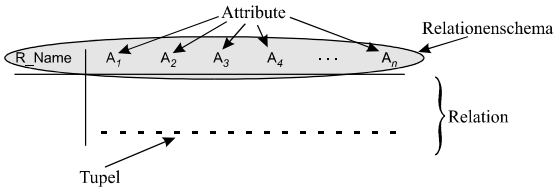
Einem Relationenschema $R(A_1, ..., A_n)$ ist eine Menge von Attributen $\{A_1, ..., A_n\}$ zugeordnet.

Jedes Attribut A_i $(1 \leq i \leq n)$ beschreibt eine von n Eigenschaften eines Objektes und ist über einen Wertebereich *dom* (A_i) definiert.

Eine Relation $r(R)$ mit Schema $R(A_1, ..., A_n)$ ist eine Menge von n-attributigen Tupeln $\{t_1, ..., t_m\}$ und jedes Tupel t_j $(1 \leq j \leq m)$ besteht aus einer Liste von n Werten $<v_1, ..., v_n>$.

Jeder Wert v_i $(1 \leq i \leq n)$ muss aus *dom* (A_i) sein und wird Attributwert genannt.

Die Menge der Tupel bezeichnet man auch als Ausprägungsebene der relationalen Datenbank.



Formale relationale Begriffe	Informelle Begriffe
Relationenschema	Tabellengerüst
Relation	Tabelle
(Menge von Tupeln	(Folge von Zeilen
- keine Duplikate	- Duplikate möglich
- keine Reihenfolge der Tupel)	- Reihenfolge der Zeilen)
Tupel	Zeile oder Satz
Kardinalität	Anzahl der Zeilen
Attribut	Spalte oder Feld
Schlüsselkandidat	eindeutige Identifikation
Wertebereich	Vorrat gültiger Werte

Relationen sind im mathematischen Sinne Mengen von Tupeln.
Es darf nie zwei Tupel mit identischen Attributwerten geben.
Um Tupel zu unterscheiden und dadurch die Mengeneigenschaft garantieren zu können, muss man Tupel identifizieren.
Im Relationen modell erfolgt dies über das Schlüsselkonzept.

Oberschlüssel

Eine Attributmenge *SK* ($SK \subseteq R$), die jedes Tupel einer Relation *r*(*R*) eindeutig bestimmt, heißt Oberschlüssel (super key) des Relationenschemas *R*.
Für *r*(*R*) gilt:

$$\forall t_i, t_j \in r : t_i \neq t_j \Rightarrow t_i[SK] \neq t_j[SK]$$

88

Schlüssel

Ist die identifizierende Attributmenge "minimal", d.h. ein Oberschlüssel, aus dem keine Attribute gestrichen werden können, ohne dass die Schlüsseleigenschaft verloren geht, so handelt es sich um einen *Schlüsselkandidaten K*.

Für ein Relationenschema kann es mehrere Möglichkeiten für Schlüsselkandidaten geben.

Aus der Menge der Schlüsselkandidaten wird ein Schlüssel als *Primärschlüssel* ausgewählt und durch Unterstreichung der zugehörigen Attribute gekennzeichnet.

Alle Attribute, die in einem Schlüsselkandidaten vorkommen, heißen *prim*, alle anderen *nicht-prim*.

89

Attribute werden in einem Schema R_i als Fremdschlüssel bezeichnet, wenn sie in einem Relationenschema R_j ($i \neq j$) Schlüsselkandidaten sind.

An die Auswahl eines Primärschlüssels ist eine besondere Integritätsbedingung geknüpft:

Primärschlüssel dürfen niemals die Nullmarke annehmen (entity integrity).

Aus praktischen Gründen sollten sie außerdem hinsichtlich ihres Speicheraufwandes minimal sein.

Beispiel: „Fuhrpark“

Schemaebene:

Fuhrpark (Kennzeichen, Motor#, . . ., Kennung)
Garage (Kennung, Anschrift, Kapazität, Öffnungszeiten)

Ausprägungsebene von Fuhrpark:

Fuhrpark	Kennzeichen	Motor#	Marke	Modell	Kennung
	E-AB 123	A642312	Opel	Tigra	G1
	D-XX 33	BI2543J	Toyota	Carina	G2
	DU-DU 1	AJ23K11	VW	Käfer	G3

Oberschlüssel: z.B. Motor# Marke
Schlüsselkandidaten: Kennzeichen*), Motor#
Primärschlüssel: Kennzeichen
Fremdschlüssel: Kennung

*) Wechselkennzeichen sind nicht vorgesehen.

Eigenschaften von Relationen

Eindeutigkeit von Tupeln

Relationen dürfen keine Duplikate enthalten.
Jedes Tupel einer Relation ist über seine Attributwerte eindeutig zu identifizieren.
$\Rightarrow$ Jede Relation muss einen Schlüsselkandidaten besitzen.

Attributwerte sind Atomar

Der Wertebereich eines Attributs muss atomar sein.

Keine Ordnung auf Tupeln oder Attributen

Weder die Attribute eines Tupels (von links nach rechts) noch die Tupel einer Relation (von oben nach unten) sind geordnet.

Operationen auf Relationen

Mengenoperationen:

Sind zwei Relationen über einem identischen Schema definiert, so können die Mengenoperationen $\cup$, $\cap$ und \ angewendet werden. Für zwei Relationen r(R) und s(R) ist

- *Vereinigung:* $r \cup s = \{t \mid t \in r(R) \vee t \in s(R)\}$
- *Durchschnitt:* $r \cap s = \{t \mid t \in r(R) \wedge t \in s(R)\}$
- *Differenz:* $r \setminus s = \{t \mid t \in r(R) \wedge t \notin s(R)\}$

Beispiel: Mengenoperationen

r:

A	B	C
a1	b1	c1
a1	b2	c1
a2	b1	c2

s:

A	B	C
a1	b2	c1
a2	b2	c1

$r \cup s$:

A	B	C
a1	b1	c1
a1	b2	c1
a2	b1	c2
a2	b2	c1

$r \cap s$:

A	B	C
a1	b2	c1

$r \setminus s$:

A	B	C
a1	b1	c1
a2	b1	c2

Selektion:

Unter Verwendung der Selektion werden Tupel, die einem Auswahlkriterium genügen, aus Relationen ausgewählt.

Die Selektion σ wählt all jene Tupel t aus einer Relation r(R) aus, die einem Selektionskriterium entsprechen.

Dabei ist ein beliebiger Vergleichsoperator θ mit $\theta \in \{<, \leq, =, >, \geq, \neq\}$ zulässig.

Selektionskriterien können über boolsche Ausdrücke zu einer komplexen Selektionsbedingung kombiniert werden.

Der Ausdruck t[A] bedeutet hierbei die Einschränkung eines Tupels t auf den Wert seines Attributes A.

- Selektion: $\sigma_{A\theta a}\, r(R) = \{t \mid t \in r(R) \wedge t\,[A]\ \theta a\}$

Beispiel: Selektion

r:

A	B	C
a1	b1	c1
a1	b2	c1
a2	b1	c2

s:

A	B	C
a1	b2	c1
a2	b2	c1

$\sigma_{(A<a2)}$ r:

A	B	C
a1	b1	c1
a1	b2	c1

$\sigma_{(A=a1) \wedge (B>b1)}$ s:

A	B	C
a1	b2	c1

96

Projektion

Die Projektion πX wählt aus einer Relation r(R) die Attributmenge X ($X \subseteq R$) aus. Duplikate Tupel in der Ergebnisrelation werden eliminiert.

- Projektion: $\pi X\ r(R) = \{t(X) \mid t \in r(R)\} = r(R)\ [X]$.

Beispiel: Projektion

r:

A	B	C
a1	b1	c1
a1	b2	c1
a2	b1	c2

s:

A	B	C
a1	b2	c1
a2	b2	c1

$\pi_{A,B}$ r:

A	B
a1	b1
a1	b2
a2	b1

π_B s:

B
b2

97

Kartesisches Produkt

Durch die Verwendung des Kartesischen Produktes werden zwei Relationen miteinander verknüpft.

Das Ergebnis des Kartesischen Produktes $r_1(R_1)$ x $r_2(R_2)$ mit $R_1 \cap R_2 = \{\}$ entspricht allen Kombinationen je eines Tupels aus Relation $r_1(R_1)$ mit einem Tupel aus Relation $r_2(R_2)$.

- Kartesisches Produkt: $r_1(R_1)$ x $r_2(R_2)$ =

$$\{t\,(R_1 \cup R_2) \mid (t\,[R_1] \in r_1(R_1)) \wedge (t_2\,[R_2] \in r_2(R_2))\}.$$

98

Beispiel: Kartesisches Produkt

r_1:

TNAME	TELNR
Berger	7
Hofer	2

r_2:

ANAME	Adresse
Berger	Dorfplatz 2
Hofer	Lange Gasse 8
Müller	Schnellstr. 80

$r_3 = r_1$ x r_2:

TNAME	TELNR	ANAME	Adresse
Berger	7	Berger	Dorfplatz 2
Hofer	2	Berger	Dorfplatz 2
Berger	7	Hofer	Lange Gasse 8
Hofer	2	Hofer	Lange Gasse 8
Berger	7	Müller	Schnellstr. 80
Hofer	2	Müller	Schnellstr. 80

$\sigma_{(TNAME = ANAME)}r_3$:

TNAME	ANAME	TELNR	Adresse
Berger	Berger	7	Dorfplatz 2
Hofer	Hofer	2	Lange Gasse 8

99

Verbund (join)

***Theta-Verbund* (theta join):**
Verknüpfung zweier Relationen $r_1(R_1)$ und $r_2(R_2)$ mit disjunkten Attributmengen ($R_1 \cap R_2 = \{\}$). $B_1 \in R_1$ und $B_2 \in R_2$ sind Attribute mit Wertebereichen, die einen Vergleich mit Hilfe eines Operators θ ($\theta \in \{<, >, \neq, =, \leq, \geq\}$) erlauben.
Der Ausdruck $[B_1 \, \theta \, B_2]$ wird Verknüpfungsbedingung genannt.

- *Theta-Verbund*: $r_1 \, [B_1 \, \theta \, B_2] \, r_2 =$

$$\{t \, (R_1 \cup R_2) \mid (t[R_1] \in r_1) \wedge (t[R_2] \in r_2) \wedge (t[B_1] \, \theta \, t[B_2])\}.$$

100

Beispiel: Gleichverbund

r_1:

TNAME	TELNR
Berger	7
Hofer	2

r_2:

ANAME	Adresse
Berger	Dorfplatz 2
Hofer	Lange Gasse 8
Müller	Schnellstr. 80

r_1[TNAME-ANAME] r_2 :

TNAME	ANAME	TELNR	Adresse
Berger	Berger	7	Dorfplatz 2
Hofer	Hofer	2	Lange Gasse 8

101

***Natürlicher Gleichverbund* (natural join):**
Um zwei Relationen $r_1(R_1)$ und $r_2(R_2)$ unter Verwendung des Natürlichen Gleichverbundes verknüpfen zu können, müssen ihre Schemata Attribute enthalten, die den gleichen Bezeichner tragen. Ist das nicht der Fall, so muss unter Verwendung des Umbenennungs-Operators die gleiche Bezeichnung erzeugt werden. Die Verknüpfung erfolgt dann auf den in R_1 und R_2 gemeinsamen Attributen.

Natürlicher Gleichverbund: $r_1 \ r_2 =$

$$\{t(R_1 \cup R_2) \mid (t[R_1] \in r_1) \wedge (t[R_2] \in r_2)\}$$

Beispiel: Natürlicher Gleichverbund

r_1:

TNAME	TELNR
Berger	7
Hofer	2

r_2:

ANAME	Adresse
Berger	Dorfplatz 2
Hofer	Lange Gasse 8
Müller	Schnellstr. 80

$r_1 * r_2$:

NAME	TELNR	Adresse
Berger	7	Dorfplatz 2
Hofer	2	Lange Gasse 8

Division:

Seien zwei Relationen $r_1(R_1)$ und $r_2(R_2)$,
mit $R_1 = X \cup Y$ und $R_2 = Y \cup Z$ und mit $X \cap Y = \{\}$ und $Y \cap Z = \{\}$.

- *Division*: $r_1 \div r_2 = r'(X) =$

$\{t(X) \mid t[R_1] \in r_1(R_1)$ und $\pi_Y r_2(R_2) \subseteq g_r(X)\}$,

mit $g_r(X) = \pi_Y(\sigma_{X=x}(r_1(R_1))$.

Für jedes Tupel $t(X)$ in der Relation $\pi_X r_1(R_1)$ bestimmt man $g_r(X)$, die Menge aller diesem X-Wert in $r_1(R_1)$ zugehörenden Y-Werte. Ist $\pi_Y r_2(R_2) \subseteq g_r(X)$, dann ist das Tupel $t(X)$ in der Ergebnismenge $r_1 \div r_2$ enthalten.

104

Beispiel: Division

Division $r_1 \div r_2$

r_1	*A*	*B*	*C*	*D*
	a1	b1	c1	d1
	a2	b2	c1	d1
	a2	b2	c2	d2
	a2	b2	c1	d2
	a3	b3	c2	d2

r_2	C	D	*E*
	c1	d1	e1
	c1	d1	e2
	c2	d2	e3

Schritt 1: Durchführen der Projektion

$\pi_X(r_1)$	*A*	*B*
	a1	b1
	a2	b2
	a3	b3

$\pi_Y(r_2)$	*C*	*D*
	c1	d1
	c2	d2

105

Schritt 2: Berechnen der Bildmengen $g_r(X)$

g_r (a1, b1) = {<c1, d1>}
g_r (a2, b2) = {<c1, d1>, <c2, d2>, <c1, d2>}
g_r (a3, b3) = {<c2, d2>}

Schritt 3: Beenden der Division durch überprüfen der Untermengenbeziehung

$\pi_y(r_2) \subseteq g_r(X)$

$r_1 \div r_2$	A	B
	a2	b2

Datenbankanomalien

Es gibt drei typische Datenbankanomalien die bei einem schlecht entworfenen Datenbankschema auftreten.

- Einfügeanomalie
- Löschanomalie
- Änderungsanomalie

Beispiel:

SVNr	M_Name	Geb-Dat	ANr	A-Bezeichnung	SVNr-ALeiter
AI232452	Müller	01.01.1959	1	Einkauf	AI232452
ZJ343452	Meier	30.03.1964	2	Marketing	EE23411
DF23452	Schmidt	15.06.1968	1	Einkauf	AI232452
EE23411	Schulz	31.05.1965	3	Beschaffung	EE23411

Funktionale Abhängigkeiten

Funktionale Abhängigkeiten (functional dependencies) (FD) stellen im Relationenmodell die wichtigste Klasse von Einschränkungen dar.

Sie werden verwendet, um die Anzahl der mathematisch möglichen Tupel (Kartesisches Produkt der Attributwertebereiche) auf die Menge der semantisch sinnvollen Tupel einzuschränken.

Die funktionale Abhängigkeit $X \rightarrow Y$ (Sprechweise X bestimmt Y funktional, Y hängt von X funktional ab) gilt im Schema R, falls alle beliebigen Tupel t_1, t_2 aus der Relation $r(R)$, die in ihren X-Werten übereinstimmen, auch in ihren Y-Attributen identische Werte aufweisen.

Funktionale Abhängigkeit

Sei $RS(S;E)$ ein Relationenschema und X, Y mit $X \cup Y \subseteq S$ zwei Attributmengen. Eine *funktionale Abhängigkeit* $f : X \rightarrow Y$ in $RS(S;E)$ ist eine Einschränkung

$f = <S$, "für je zwei Tupel $t_i, t_j \in r(S)$ gilt: $t_i[X] = t_j[X] \Rightarrow t_i[Y] = t_j[Y]>$.

Mehrwertige Abhängigkeit

Sei $RS(S;E)$ ein Relationenschema und X, Y mit $X \cup Y \subseteq S$ und Z mit $Z = S \setminus (X \cup Y)$ drei Attributmengen. Eine *mehrwertige Abhängigkeit* m: $X \rightarrow\rightarrow Y$ (X bestimmt Y mehrwertig) der Attributmenge X und Y in $RS(S;E)$ ist eine Einschränkung

$m = <S$, "für jeden X-Wert x und Z-Wert z gilt:

$$\sigma_{X=x}\, r(S)[Y] = \sigma_{(X=x) \wedge (Z=z)}\, r(S)[Y]> .$$

Inferenzregeln für FD

Axiomensystem nach Armstrong

Sei $RS(S;F)$ ein Relationenschema und $W, X, Y, Z \subseteq S$.

FD(1):	Reflexivität:	$Y \subseteq X \Rightarrow X \rightarrow Y$
FD(2):	Erweiterung:	$X \rightarrow Y$ und $Z \subseteq W \Rightarrow XW \rightarrow YZ$
FD(3):	Transitivität:	$X \rightarrow Y \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$

Die Axiome FD1 bis FD3 sind korrekt (sound), d.h. sie leiten von F nur wirklich gültige FD ab.

Sie sind auch vollständig (complete), d.h. sie erzeugen bei wiederholter Anwendung alle von F implizierten FD.

Aus den Armstrong-Axiomen können die folgenden weiteren nützlichen Abteilungsregeln hergeleitet werden:

FD(4):	Vereinigung:	$X \rightarrow Y$ und $X \rightarrow Z \Rightarrow X \rightarrow YZ$
FD(5):	Pseudotransitivität:	$X \rightarrow Y$ und $WY \rightarrow Z \Rightarrow WX \rightarrow Z$
FD(6):	Zerlegung:	$X \rightarrow YZ \Rightarrow X \rightarrow Y$ *und* $X \rightarrow Z$

Hülle

Die Hülle F^+ einer Abhängigkeitsmenge F eines Schemas $RS(S;F)$ beinhaltet alle FD, die entweder bereits in F enthalten sind oder aus den FD aus F abgeleitet werden können ($F^+ = \{f \mid F \Rightarrow f\}$).

Die Hülle X^+ einer Menge von Attributen $X \subseteq S$ bezüglich F ist die Menge aller Attribute A, so dass $X \rightarrow A$ aus F^+ ist ($X^+ = \{A \mid (X \rightarrow A) \in F^+\}$).

Der Algorithmus MEMBERSHIP

Algorithmus MEMBERSHIP (F, $X \rightarrow Y$) kann zur Anwendung kommen, um festzustellen, ob $X \rightarrow Y \in F^+$ gilt oder nicht.

Algorithmus CLOSURE (X, F)

Input: eine Attributmenge X und eine Menge F von FD.

Output: X^+ bezüglich F.

CLOSURE (X, F)

```
BEGIN   old := {};
        new := X;
        WHILE new ≠ old do
                BEGIN old := new;
                      FOR EACH FD W→Z∈F do
                      IF new ⊇ W THEN new := new ∪ Z;
                END;
        RETURN (new);
END
```


Algorithmus MEMBERSHIP (F, $X \rightarrow Y$)

Input: eine Menge F von FD und eine FD $X \rightarrow Y$.

Output: true falls $F \Rightarrow X \rightarrow Y$, false sonst.

MEMBER (F, $X \rightarrow Y$)

```
BEGIN IF Y ⊆ CLOSURE (X, F)      THEN RETURN (TRUE)
ELSE RETURN (FALSE)
END
```

Beispiel: Membershipproblem

Sei $RS(S;F)$ ein Relationenschema mit $S = \{A, B, C\}$ und $F = \{AB \rightarrow C\}$. Unter Anwendung der Armstrong-Axiome soll die Hülle F^+ errechnet werden, und mit Hilfe des Membership-Algorithmus soll geprüft werden, ob $AB \rightarrow AC \in F^+$ gilt.

$F^+(S) = \{A \rightarrow A, B \rightarrow B, C \rightarrow C$
AB→AB, AB→A, AB→B
BC→BC, BC→B, BC→C
AC→AC, AC→A, AC→C
ABC→ABC, ABC→AB, ABC→AC
ABC→BC, ABC→A, ABC→B, ABC→C
AB→C, AB→AC, AB→BC, AB→ABC}

Der Membership-Algorithmus liefert $(AB)^+ = \{ABC\}$ und da $AC \subseteq ABC$ folgt $AB \rightarrow AC \in F^+$.

114

Beispiel: Überprüfung der Schlüsseleigenschaft

$RS(S;F)$ mit $S = \{A, B, C, D\}$ und $F = \{A \rightarrow D, B \rightarrow C\}$ sei gegeben. Es soll geprüft werden, ob AB ein Schlüssel für RS ist, d.h. ob $\{AB\}^+ = S$ gilt.

1.	$A \rightarrow D$	gegeben
2.	$AB \rightarrow ABD$	FD(2): Erweiterung von 1. um AB
3.	$B \rightarrow C$	gegeben
4.	$ABD \rightarrow ABCD$	FD(2) Erweiterung von 3. um ABD
5.	$AB \rightarrow ABCD$	FD(3) angewendet auf 2. und 4.

115

Äquivalente Abhängigkeitsmengen

Zwei Mengen funktionaler Abhängigkeiten F und G sind äquivalent ($F \equiv G$), wenn $F^+ = G^+$ gilt.

Äquivalenz kann geprüft werden, indem für alle $g \in G$: $F \Rightarrow g$ und für alle f $\in F$: $G \Rightarrow f$ ermittelt wird.

Sind F und G äquivalent, so heißt G Überdeckung (cover) von F und umgekehrt.

Zur Überprüfung der Äquivalenz können sowohl die Armstrong-Axiome als auch der Membership-Algorithmus eingesetzt werden.

116

Minimale Überdeckung

F_1 heißt minimale Überdeckung von F, wenn F_1 zu F äquivalent ist und eine Teilmenge von F_1 diese Eigenschaft nicht aufweist.

Um eine minimale Überdeckung einer Abhängigkeitsmenge zu ermitteln, werden redundante Abhängigkeiten bzw. unnötige Attribute zu eruiert und aus F gestrichen.

Es für ein bestimmtes F mehrere unterschiedliche kanonische Überdeckungen geben.

Enthält F eine nicht redundante Menge an FD, kann kein Element aus F gestrichen werden, ohne dass sich die Hülle F^+ von F ändert.

Zur Bestimmung minimaler Überdeckungen kann auf die Armstrong-Axiome bzw. den Membership-Algorithmus zurückgegriffen werden.

117

Die Algorithmen NONRED (F), REDUCE (G), LEFTED (G) und RIGHTED (G)

Algorithmus NONRED (F)

Input: eine Abhängigkeitsmenge *F*.

Output: eine nicht redundante Abhängigkeitsmenge *F*'.

NONRED(F)

Begin FOR EACH FD $X \rightarrow Y$ DO

IF *MEMBER* ($F \backslash X \rightarrow Y$, $X \rightarrow Y$) DO $F = F \backslash X \rightarrow Y$;

RETURN (*F*);

END

Algorithmus REDUCE (G)

Input: eine Menge *G* von FD

Output: eine minimale Überdeckung von *G*

REDUCE (G)

BEGIN F := *RIGHTRED* (*LEFTRED*) (G)

lösche alle FD der Form $X \rightarrow \{\}$ aus *F*;

RETURN (*F*);

END

Algorithmus LEFTRED (G)

Input: eine Menge *G* von FD
Output: eine linksreduzierte Überdeckung *F* von *G*

LEFTRED (G)

```
BEGIN     F := G
          FOR EACH FD X→Y ∈ G DO
                FOR EACH attribute A ∈ X DO
                      IF MEMBER (F, (X\A)→Y) THEN
                      lösche A aus X in X→Y in F;
          RETURN (F)
END
```

120

Algorithmus RIGHTRED (G)

Input: eine Menge *G* von FD
Output: eine rechtsreduzierte Überdeckung *F* von *G*

RIGHTRED (G)

```
BEGIN  F := G
       FOR EACH FD X→Y ∈ G DO
            FOR EACH attribute A ∈ Y DO
                 IF MEMBER (F\{X→Y}∪{X→(Y\A)}, X→A)
                 THEN
                 lösche A aus Y in X→Y in F;
       RETURN (F)
END
```

121

Sei F eine Menge von Abhängigkeiten über einem Schema $RS(S;F)$ und $X \rightarrow Y \in F$.

a. $X \rightarrow Y$ heißt *links reduzierbar*, wenn es ein $X' \subset X$ gibt, so dass $X' \rightarrow Y \in F^+$.

b. $X \rightarrow Y$ heißt *rechts reduzierbar*, wenn es ein $Y' \subset Y$ gibt, so dass $X \rightarrow Y' \in F^+$.

c. Eine FD $X \rightarrow Y$ heißt *reduziert*, wenn sie links und rechts reduziert ist und $Y \neq \{\}$.

d. Eine links reduzierte Abhängigkeit heißt *voll funktionale Abhängigkeit.*

Beispiel: Minimale Überdeckung

Gegeben ist $F = \{A \rightarrow BC, BCD \rightarrow E, B \rightarrow CD, ABD \rightarrow E\}$. Es soll eine minimale Überdeckung entwickelt und gezeigt werden, dass die Lösung mit *F* äquivalent ist.

Schritt 1: links reduzieren

1. $A \rightarrow BC$
2. $B \rightarrow E \quad \Rightarrow B \rightarrow E$, da $B \rightarrow CD$ bereits in F ist.
3. $B \rightarrow CD$
4. $AD \rightarrow E \quad \Rightarrow AD \rightarrow E$, da $A \rightarrow B$ bereits in F ist.

Beispiel:Minimale Überdeckung (Fortsetzung)

Schritt *2:* *rechts* reduzieren, Streichen von Redundanzen

1. $A \rightarrow B \quad \Rightarrow A \rightarrow B$, da aus $A \rightarrow B$ und $B \rightarrow C$ (aus 3.) $A \rightarrow C$ bereits folgt.
2. $B \rightarrow E$
3. $B \rightarrow C$, $B \rightarrow D$
4. $AD \rightarrow E$ *fällt weg, da $A \rightarrow B$ (aus 1.) und $B \rightarrow E$ (aus 2.) $AD \rightarrow E$ bestimmen.*

$F' = \{A \rightarrow B, B \rightarrow CDE\}$ ist eine minimale Überdeckung von F.

$F' \equiv F$ gilt, da $A \rightarrow B$ und $B \rightarrow CD$ aus F' direkt in F enthalten sind. $B \rightarrow E$ aus F' kann in F folgend hergeleitet werden: $B \rightarrow CD$ (gegeben), daraus folgt aus FD(2) und Erweiterung um B: $B \rightarrow BCD$. $BCD \rightarrow E$ ist gegeben. $B \rightarrow BCD \wedge BCD \rightarrow E \Rightarrow B \rightarrow E$. Das bedeutet, alle FD aus F' sind in der Hülle von F enthalten.

124

Beispiel:Minimale Überdeckung (Fortsetzung)

Auch der umgekehrte Fall gilt, wie folgende Ableitungen zeigen:

- $A \rightarrow BC$: $A \rightarrow B \wedge B \rightarrow C \Rightarrow A \rightarrow BC$ (FD(3) + FD(4))
- $BCD \rightarrow E$: $B \rightarrow E \Rightarrow BCD \rightarrow E$ (FD(2))
- $B \rightarrow CD$: in F' direkt enthalten
- $ABD \rightarrow E$: $A \rightarrow B \wedge B \rightarrow E \Rightarrow A \rightarrow E$ (FD(3))
  $A \rightarrow E \Rightarrow ABD \rightarrow E$ (FD(2))

125

Schlüssel:

Sei $RS(S;F)$ ein Relationenschema, dann ist $X \subseteq S$ ein *superkey* (Oberschlüssel) für das Schema RS, falls $X \rightarrow S \in F^+$.
Sei $X \rightarrow S \in F^+$, dann ist X ein *key* (oder candidate key) von RS, wenn kein $X' \subset X$ existiert, mit $X' \rightarrow S$. Ist X key, dann heißt $X \rightarrow S$ *key dependency.*

Bearbeitungschritte zur Berechnung aller Schlüsselkandidaten:

Schritt 1:

Bilde alle Teilmengen X von S mit n-1 Attributen. Überprüfe für jedes X : $X \rightarrow S$. Ist das der Fall, ist X ein Oberschlüssel, ansonsten ein Nichtschlüssel.

126

Schritt 2:

In den weiteren i Schritten werden alle Teilmengen X von S mit (n-i) Attributen, die Teilmenge eines Oberschlüssels aber nicht Teilmenge eines Nichtschlüssels sind, gebildet.

Schritt 3:

Generiere alle Schichten bis n, solange Oberschlüssel vorhanden sind.

Schritt4:

Schlüsselkandidaten sind alle Attributmengen K, die selbst Oberschlüssel sind und für die gilt, dass keine Teilmenge Oberschlüssel ist.

127

Normalformen

Normalformen stellen ein Meßverfahren für die Qualität und Güte eines Relationenschemas dar.

Erste Normalform:

Ein Relationenschema $RS(\{A_1, ..., A_n\}; \{F\})$ ist in 1*NF*, falls $dom(A_i)(i=1..n)$ atomar ist.

Zweite Normalform:

Ein 1*NF* Relationenschema $RS(S;F)$ ist in 2*NF*, falls jedes nicht prime Attribut aus S von jedem Schlüssel von RS voll funktional abhängig ist.

Dritte Normalform:

Ein 1*NF* Relationenschema $RS(S;F)$ ist in 3*NF*, falls kein nicht-primes Attribut strikt transitiv vom Schlüssel abhängt.

Strikte transitive Abhängigkeit:

Sei X, $Y \subseteq S$ und A ein Attribut aus S. Attribut A ist von X über Y strikt transitiv abhängig, falls folgendes gilt:
$X \rightarrow Y$, $Y \not\rightarrow X$, $Y \rightarrow A$, $A \notin XY$.

Boyce-Codd-Normalform:

Ein 1*NF* Relationenschema $RS(S;F)$ ist in BCNF, falls für jedes $Y \subseteq S$ und für jedes Attribut $A \in S \setminus Y$ gilt: $Y \rightarrow A \Rightarrow Y \rightarrow S$.

Vierte Normalform:

Ein 1*NF* Relationenschema *RS*(*S*;*F*) ist in 4*NF*, falls für jede MVD der Form $X \rightarrow\rightarrow Y$ gilt: $X \rightarrow\rightarrow Y$ ist trivial oder X ist Oberschlüssel von *RS*.

Beispiel: Bestimmung der Normalform

Gegeben:
RS(*S*;*F*) mit *S* = {*A, B, C, D, E*} und *F* = {$AB \rightarrow CE$, $E \rightarrow AB$, $C \rightarrow D$}

F ist bereits minimal.

Schlüsselkandidaten: *AB, E*

Daraus folgt, dass die Attribute *CD* nicht prim sind.
Da weder *C* noch *D* von einer Teilmenge eines Schlüsselkandidaten abhängen, befindet sich *RS*(*S*;*F*) in 2*NF.*
3*NF* ist nicht gegeben, da das nicht-prime Attribut *D* strikt transitiv funktional vom Schlüsselkandidaten *AB* abhängt.

130

Beispiel: Bestimmung der Normalform

Gegeben:
RS(*S*;*F*) mit *S* = {*A, B, C, D*} und *F* = {$AC \rightarrow BD$, $D \rightarrow A$, $CD \rightarrow A$}

Minimale Überdeckung von F: *F'* = {$AC \rightarrow BD$, $D \rightarrow A$}

Schlüsselkandidaten: *AC* und *CD*

nicht prime Attribute*:* *B*

Das Schema *RS*(*S*;*F*) ist in 2*NF*, da *B* ausschließlich voll funktional von Schlüsselkandidaten abhängt.

Es ist in 3*NF*, da *B* von keinem Schlüsselkandidaten strikt transitiv abhängt.

Das Schema ist nicht in BCNF, da die funktionale Abhängigkeit $D \rightarrow A$ keine key dependency darstellt.

131

Normalisierung

Überführen eines Ausgangsschemas mit ungünstigen Eigenschaften in Fragmente höherer Normalform.

Semantik muss erhalten bleiben.

Es darf kein Informationsverlust stattfinden.

Gewährleistung von Abhängigkeits- und Verbundtreue.

Eine Zerlegung von $RS(S;F)$ in $\{RS_i(X_i;F_i)\}$ (i=1..n) ist gültig, wenn

- $\bigcup_{i=1}^{n} X_i = S$ (attributerhaltende Zerlegung)
- $(\bigcup_{i=1}^{n} F_i)^+ (S) \equiv F^+ (S)$ (abhängigkeitstreue Zerlegung)

  d.h. Vereinigung aller F_i auf S ist zu F äquivalent.
- jede Relation $r(S)$ mit Schema $RS(S;F)$ und Zerlegung $\{RS_i(X_i;F_i)\}$ (i=1..n) erfüllt die Bedingung

  $\bowtie_{i=1}^{n} r_i (X_i) = r(S)$ (verbundtreue Zerlegung)

  d.h. natürlicher Verbund der Projektionen von r(S) auf die Teilmengen X_i ergibt ursprüngliche Relation r(S).

Beispiel: Verletzung von Abhängigkeitstreue

RS ({A B C D E}; {A→BCD, CD→E, AE→B}) wird zerlegt in:
RS_1 ({A B C D}; {A→BCD}) und RS_2 ({C D E}; {CD→E})

Beispiel: Verletzung von und Verbundtreue

RS ({A B C D}; {A→B, C→D}) wird zerlegt in:
RS_1 ({A B}; {A→B}) und RS_2 ({C D}; {C→D})

Algorithmus NORMALIZATION (*RS*(*S*;*F*))

Input: ein Relationenschema *RS*(*S*;*F*) in ungünstiger Normalform

Output: eine gültige 3*NF*-Zerlegung von *RS*(*S*;*F*) in {$RS_i(X_i;F_i)$} (*i*=1..*n*)

NORMALIZATION (*RS*(*S*;*F*))

begin F := REDUCE (*F*)

berechne die Schlüssel von *RS*(*S*;*F*)
FOR EACH FD $X \rightarrow Y \in F$ do
berechne X^+;

berechne Gruppen äquivalenter FD
FOR EACH Gruppe do
erzeuge ein Schema $RS_i(X_i;F_i)$.
X_i sind alle Attribute und
F_i alle FD der Gruppe;

MERGE ($RS_i(X_i;F_i)$, $RS_j(X_j;F_j)$)

IF NOT verbundtreue Zerlegung THEN
erzeuge ein weiteres Schema $RS_i(X_i;F_i)$
mit X_i Attribute eines
Schlüsselkandidatenund F_i = {}

END

MERGE ($RS_i(X_i;F_i)$, $RS_j(X_j;F_j)$)

BEGIN IF $X_i \subseteq X_j$ *and* $RS_i(X_j \cup X_j;F_i \cup F_j)$ genügen der 3NF
THEN mische $RS_i(X_i;F_i)$and $RS_j(X_j;F_j)$)

END

136

BEISPIEL: Normalisierung

F={

Übung→Assistent,
/* Jede Übung wird von einem Assistenten betreut.*/

Übung, Student→Punkte,
/*Nur eine Punkteanzahl pro Student und Übung.*/

Zeit, Hörsaal→Übung,
/*Keine Übung teilt zur selben Zeit einen Hörsaal mit einer anderen.*/

Zeit, Student→Hörsaal,
/*Ein Student kann zu einer Zeit nur in einem Hörsaal sein.*/

Zeit, Assistent→Hörsaal,
/*Ein Assistent kann zu einer Zeit nur in einem Hörsaal sein.*/

137

BEISPIEL: Normalisierung (Fortsetzung)

Zeit, Student→Übung,
/* Ein Student kann nicht gleichzeitig mehrere Übungen besuchen.*/

Zeit, Assistent→Übung
/*Ein Assistent kann nicht gleichzeitig mehrere Übungen leiten.*/

Zeit, Übung→Hörsaal
/*Eine Übung kann zu einer Zeit nur in einem Hörsaal stattfinden*/

}

Für *F* ist eine kanonische Überdeckung zu bilden. Falls das Schema nicht in 3*NF* ist, sind daraus verbund- und abhängigkeitstrue 3*NF*-Fragmente herzuleiten.

Reduce (F) liefert:
F' = {*U*→*A*, *US*→*P*, *ZH*→*U*, *ZS*→*U*, *ZA*→*U*, *ZU*→*H*}

BEISPIEL: Normalisierung (Fortsetzung)

Gruppen äquivalenter FD:

$\{U\}^+$=	UA	Gruppe 1
$\{US\}^+$=	USP	Gruppe 2
$\{ZH\}^+$=	ZHUA	Gruppe 4
$\{ZS\}^+$=	ZSUAHP	Gruppe 3
$\{ZA\}^+$=	ZHUA	Gruppe 4
$\{ZU\}^+$=	ZHUA	Gruppe 1

ZH→*U*, *ZA*→*U* und *ZU*→*H* *äquivalente FD*

Dies führt zu:

RS_1 ({*A U*}; {*U*→*A*}),
RS_2 ({*U S P*}; {*US*→*P*})
RS_4 ({*Z S U*}; {*ZS*→*U*}) und
RS_3 ({*Z H U A*}; { *ZH*→*U*, *ZA*→*U*, *ZU*→*H*})

BEISPIEL: Normalisierung (Fortsetzung)

Mögliche Lösung:

Prüfung ({Übung Student Punkte}; {Übung Student→Punkte})
SK: Übung Student, BCNF

Zuordnung ({Zeit Hörsaal Übung Assistent}; {Zeit Hörsaal→Übung, Zeit Assistent→Übung, Zeit Übung→Hörsaal, Übung→Assistent})
SK: Zeit Hörsaal, Zeit Assistent, Zeit Übung, 3*NF*

Stundenplan ({Zeit Student Übung}; {Zeit Student→Übung})
SK: Zeit Student, BCNF

140

Logischer Entwurf

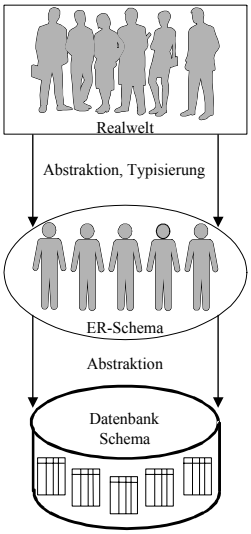
Aufgabe:
Übersetzung des semantischen Schemas der konzeptuellen Modellbildung in ein logisches Datenbankmodell.

Notation:
Die Bezeichnung von Entitytypen und Beziehungstypen aus dem ERM wird übernommen.

Primärschlüssel	Attribute des Primärschlüssels
Fremdschlüssel	Fremdschlüssel
Eigenschaften	Eigenschaften, die nicht prime Attribute darstellen; Nullmarken sind nicht gestattet

141

Abbildungsprozess Realwelt ↔ relationales Datenbankmodell



Realwelt
- sehr vielschichtig
- überwiegend Unikate
- umfangreiche Beziehungen

Semantisches Datenmodell
- Zusammenfassung von Entities zu Gruppen
- zwar schon stark abstrahierendes aber vergleichsweise immer noch mächtiges Datenmodell,
- Beziehungen explizit modellierbar
- komplexe Integritätsbedingungen

relationales Datenbankmodell
- Abbildung auf Tupel und (normalisierte) Relationen
- einfaches Datenmodell
- Beziehungen nur implizit modellierbar

142

Transformation von Entitytypen

Jeder Entitytyp wird in ein Relationenschema übersetzt.

Der Name bleibt unverändert.

Alle Attribute werden übernommen.

Primärschlüssel wird durch Unterstreichung kenntlich gemacht.

Transformation von Beziehungstypen

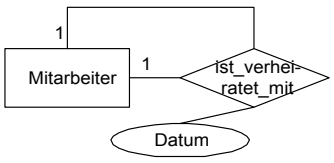
Sie werden als eigenes Relationenschema oder als Attribut in das logische Datenmodell aufgenommen.

143

Transformation rekursiver Beziehungtypen

Für <1:1> und <1:N> Beziehungstypen wird der umbenannte Primärschlüssel an das bereits bestehende Ralationenschema angfügt.
Bei einer Kardinalität <N:M> muss ein zusätzlich Relationen-schema erstellt werden.

Beispiel: Transformation unärer <1:1>-Beziehungstypen

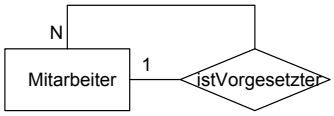


Ein Mitarbeiter kann mit einem anderen Mitarbeiter verheiratet sein.

Mitarbeiter (MitarbeiterNr, Name, ..., PartnerNr,Datum)

144

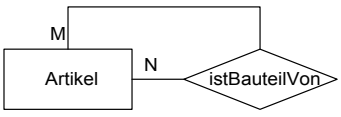
Beispiel: Transformation unärer <1:N>-Beziehungstypen



Ein Mitarbeiter kann Vorgesetzter mehrerer Mitarbeiter sein. Jeder Mitarbeiter hat ausschließlich einen Vorgesetzten.

Mitarbeiter (MitarbeiterNr, Name, ..., VorgesetzterNr)

Beispiel: Transformation unärer <1:N>-Beziehungstypen



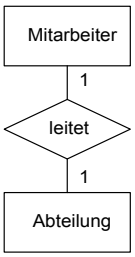
Jeder Artikel kann verschiedene Bauteile besitzen. Ein Bauteil kann Bestandteil in mehreren Artikel sein.
Artikel(ArtikelNr, Bezeichnung, ...)
ist_Bauteil_von(ArtikelNr, BauteilNr)

145

Transformation binärer Beziehungstypen

Beispiel: Transformation binärer <1:1>-Beziehungstypen

Nehmen beide Entitytypen optional am Beziehungstypen teil, so bedeutet dies, dass Abteilungen von einem Mitarbeiter geleitet werden können, aber durchaus auch Abteilungen existieren können, die keinen Leiter besitzen. Ebenso können Mitarbeiter existieren, die keiner Abteilung zugeordnet sind.

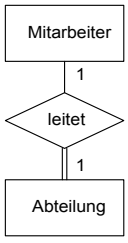


Mitarbeiter(<u>MitarbeiterNr</u>, Name, ...AbtNr)

Abteilung(<u>AbtNr</u>, AbtName, ...)

Beispiel: Transformation binärer <1:1>-Beziehungstypen (Fortsetzung)

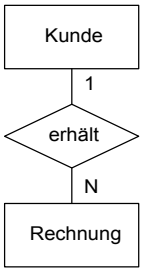
In dieser Variante ist die Teilnahme aller Abteilungen am Beziehungstyp bindend. Jede Abteilung muss von einem Mitarbeiter geleitet werden, aber nicht jeder Mitarbeiter muss auch Abteilungsleiter sein. In diesem Fall muss die MitarbeiterNr als Fremdschlüssel in das Relationenschema Abteilung aufgenommen werden und darf nicht die Nullmarke annehmen.



Mitarbeiter	(<u>MitarbeiterNr</u>, Name, ...)
Abteilung	(<u>AbtNr</u>, AbtName, ..., *MitarbeiterNr*)

Beispiel: Transformation binärer <1:N>-Beziehungstypen

Auch bei diesem Beispiel soll der Unterschied in der Transformation optionaler und totaler Teilnahmen an einem Beziehungstypen verdeutlicht werden. In der ersten Ausprägung des Beispiels kann jeder Kunde mehrere Rechnungen erhalten. Rechnungen sind dabei einem oder keinem Kunden zugeordnet. Die Transformation in das Relationenmodell ergibt folgendes Bild:



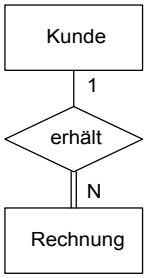
Kunde (<u>KundenNr</u>, Name, Vorname, ...)

Rechnung(<u>RechNr</u>, Datum, ..., KundenNr)

148

Beispiel: Transformation binärer <1:N>-Beziehungstypen (Fortsetzung)

Einen realistischeren Fall beschreibt die zweite Alternative, bei der jede Rechnung einem Kunden zugeordnet sein muss, aber Kunden auch weiterhin keine Rechnung erhalten können. Für das Realtionenmodell bedeutet dies, dass der Fremdschlüssel KundenNr in Rechnung keine Nullmarke annehmen darf.



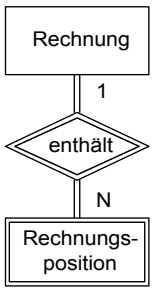
Kunde(<u>KundenNr</u>, Name, Vorname, ...)

Rechnung(<u>RechNr</u>, Datum, ..., *KundenNr*)

149

Beispiel: Transformation binärer <1:N>-Beziehungstypen (Fortsetzung)

Einen Sonderfall stellt die Transformation eines schwachen Entitytypen dar. In einem solchen Fall gehört der Fremdschlüssel mit zum Primärschlüssel des "schwachen" Entitytypen. Zum Beispiel sind Rechnungspositionen nur im Zusammenhang mit der Rechnungsnummer eindeutig identifizierbar.

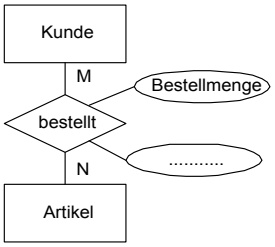


Rechnung (RechNr, Datum, ...)

RechnungsPos (RechNr, PosNr, Menge,..)

Beispiel: Transformation binärer <N:M>-Beziehungstypen

<N:M>-Beziehungstypen werden grundsätzlich in ein eigenes Relationenschema transformiert. In diesem Beispiel kann ein Kunde verschiedene Artikel bestellen und jeder Artikel kann von beliebig vielen Kunden bestellt werden.



Kunde (KundenNr, Name, Vorname, ...)

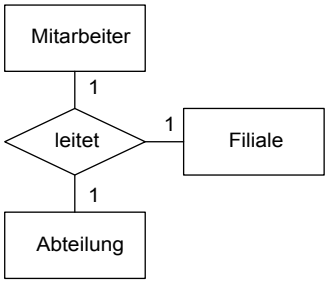
Artikel (ArtikelNr, Abezeichnung, ...)

bestellt (KundenNr, ArtikelNr, Bestellmenge, ..)

Transformation n-ärer Beziehungstypen

Beispiel: Transformation tenärer <1:1:1>-Beziehungstypen

Ein Mitarbeiter kann *pro* Filiale nur eine Abteilung leiten. Pro Filiale ist besonders zu betonen, da aus diesem Grund zwei Primärschlüssel der beteiligten Entitytypen nötig sind, um den Primärschlüssel des Beziehungstypen zu bilden.



Mitarbeiter(MitarbeiterNr, Name, ...)
Abteilung(AbtNr, AbtName, ...)
Filiale(FilialNr, Fbezeichnung, Ort,..)

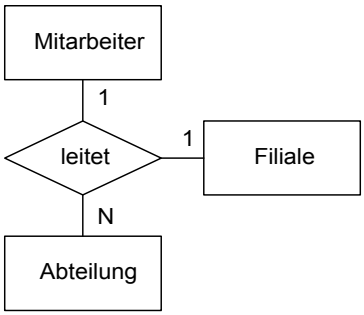
Alternative Transformationen des Beziehungstypen:

leitet(MitarbeiterNr, AbtNr, FilialNr)
leitet(MitarbeiterNr, AbtNr, FilialNr)
leitet(MitarbeiterN*r*, AbtNr, FilialNr)

152

Beispiel: Transformation tenärer <1:1:N>-Beziehungstypen

In diesem Fall kann ein Mitarbeiter mehrere Abteilungen einer Filiale leiten. Eine Abteilung kann pro Filiale weiterhin nur einen Leiter haben, aber in einer anderen Filiale kann dieselbe Abteilung von einem anderen Mitarbeiter geleitet werden.



Mitarbeiter(MitarbeiterNr, Name, ...)
Abteilung(AbtNr, AbtName, ...)
Filiale(FilialNr, Fbezeichnung, Ort,..)

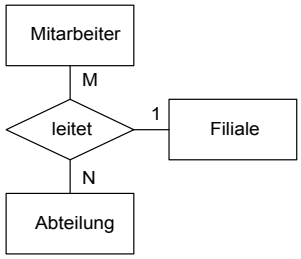
Alternative Transformationen des Beziehungstypen:

leitet(*MitarbeiterNr*, AbtNr, FilialNr)
leitet(MitarbeiterNr, AbtNr, *FilialNr*)

153

Beispiel: Transformation tenärer <1:N:M>-Beziehungstypen

Im Gegensatz zu kann nun eine Abteilung auch mehrere Abteilungsleiter haben. Aber immer noch gilt die Einschränkung, dass ein Mitarbeiter eine ganz bestimmte Abteilung nur in einer Filiale leiten kann. Er könnte jedoch durchaus einer anderen Abteilung (z.B. Beschaffung, Marketing, ...) der weiteren Filiale vorstehen. Die Transformation in das Relationenmodell ergibt folgends Bild:

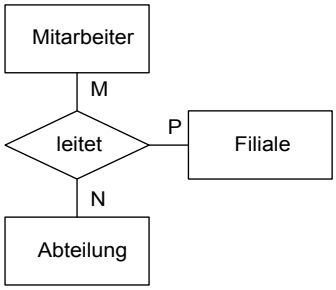


Mitarbeiter(MitarbeiterNr, Name,...)
Abteilung(AbtNr, AbtName, ...)
Filiale(FilialNr, Fbezeichnung, Ort,..)
leitet(MitarbeiterNr, AbtNr, *FilialNr*)

154

Beispiel: Transformation tenärer <N:M:P>-Beziehungstypen

In diesem Beispiel existieren keine Einschränkungen mehr hinsichtlich der Beziehung der verschiedenen Tupel der beteiligten Entitytypen. Das ERM wird in die nachfolgend dargestellten Relationenschemata transformiert.



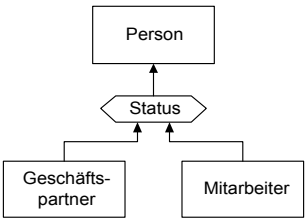
Mitarbeiter(MitarbeiterNr, Name,...)
Abteilung(AbtNr, AbtName, ...)
Filiale(FilialNr, Fbezeichnung, Ort,.)
leitet(MitarbeiterNr, AbtNr, FilialNr)

155

Transformation vopn Generalisierung und Subtypenhierachie

Beispiel: Transformation von Generalisierungshierarchien

Im Falle einer Generalisierung wird der Diskriminator explizit modelliert und ist Bestandteil des generischen Entitytypen. In diesem Beispiel entscheiden seine Werte über die Zugehörigkeit eines Entities vom Typ Person zu den disjunkten Klassen Geschäftspartner und Mitarbeiter.

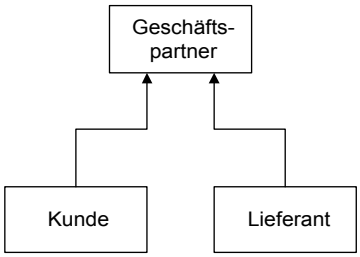


Person (<u>PersonNr</u>, Name, Vorname, ..., Status)
Geschäftspartner(<u>PersonNr</u>, Land, MwSt,..)
Mitarbeiter (<u>PersonNr</u>, Einstellungsdatum, Gehalt, ...)

156

Beispiel: Transformation von Subtypenhierarchie

Bei Subtypenhierarchien ist der Diskriminator lediglich implizit im Modell vorhanden, d.h. in dem Beispiel existieren zwar ein oder mehrere Attribute im Entitytyp Geschäftspartner, die entscheiden, ob ein Entity zum Entitytyp Kunde, Lieferant oder gar zu beiden gehört, aber diese werden nicht im ER-Diagramm dargestellt.



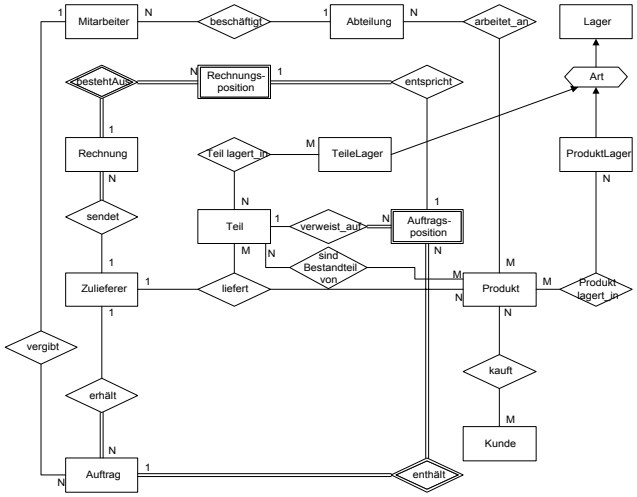
Geschäftspartner(<u>PersonNr</u>,Rabatt, MwSt, ...)

Kunde(<u>PersonNr</u>, Bonität, ...)

Lieferant(<u>PersonNr</u>, Skonto, ...)

157

Beispiel: Logischer Entwurf Produktionsunternehmen



Beispiel: Logischer Entwurf Produktionsunternehmen (Fortsetzung)

Transformation von Entitytypen

1. Mitarbeiter(mitarbeiterNr, mitarbeiterName,...
2. Abteilung(abteilungsNr, abteilungsName, mitarbeiterZahl, ...
3. Produkt(produktNr, produktBez, produktTyp, stueckKosten,...
4. Teil(teilNr, teilBez, ...
5. Lager(lagerNr, lagerBez, plz, ort, strasse, hausNr, art, ...
6. ProduktLager(lagerNr, ...)
7. TeileLager(lagerNr, ...)
8. Zulieferer(zuliefererNr, zuliefererName, plz, ort, strasse, hausNr,...)
9. Auftrag(auftragsNr, datum, ...)
10. AuftragsPosition(auftragsNr, auftragsPos, menge, ...)
11. Rechnung(rechnungsNr, datum, ...)
12. RechnungsPosition(rechnungsNr, rechnungsPos, menge, ...)
13. Kunden(kundenNr, kundenName, kundenVorname, plz, ort,...)

Beispiel: Logischer Entwurf Produktionsunternehmen (Fortsetzung)

Transformation von Beziehungstypen mit Grad > 2 oder Kardinalität <N:M>

1. ArbeitetAn(abteilungsNr, produktNr)
2. Liefert(*zuliefererNr*, teilNr, produktNr)
3. ProduktLagerIn(produktNr, produktLagerNr, istBestand)
4. TeilLagerIn(teileLagerNr, teilNr, istBestand)
5. SindBestandteilVon(teilNr, produktNr)
6. Kauft(kundenNr, produktNr)

160

Beispiel: Logischer Entwurf Produktionsunternehmen (Fortsetzung)

Transformation von Beziehungstypen mit Kardinalität <1:1> oder <1:N>

1. Mitarbeiter(mitarbeiterNr, mitarbeiterName, mitarbeiterVorname, geburtsDatum, plz, ort, strasse, hausNr, gehalt, einstellung, mitarbeiterStatus, abteilungsNr)
2. Abteilung(abteilungsNr, abteilungsName, budget, mitarbeiterZahl)
3. Produkt(produktNr, produktBez, produktTyp, stueckKosten, nettoPreis)
4. Zulieferer(zuliefererNr, zuliefererName, plz, ort, strasse, hausNr, bank, blz, kontoNr, ansprechpartner, ansprechpartTelNr)
5. Teil(teilNr, teilBez)

161

Beispiel: Logischer Entwurf Produktionsunternehmen (Fortsetzung)

6. Lager(lagerNr, lagerBez, plz, ort, strasse, hausNr, art)
7. ProduktLager(lagerNr)
8. TeileLager(lagerNr)
9. Auftrag(auftragsNr, bearbeiterNr, datum, *zuliefererNr*)
   Auftrag[auftragsNr]=AuftragsPosition[auftragsNr]
10. AuftragsPosition(auftragsNr, auftragsPos, menge, *teilNr*)
11. Rechnung(rechnungsNr, datum, *zuliefererNr*)
    Rechnung[rechnungsNr]=RechnungsPosition[rechnungsNr]
12. RechnungsPosition(rechnungsNr, rechnungsPos, menge, betrag, *auftragsNr*, *auftragsPos*)
13. Kunden(kundenNr, kundenName, kundenVorname, plz, ort, strasse, hausNr, rabatt, ansprechpartner, ansprechpartTelNr)

162