

1 Datenmodellierung

Abbildung 1: Einleitung	3
Abbildung 2: Dateisystem	3
Abbildung 3: Aussensicht	4
Abbildung 4: Innensicht	4
Abbildung 5: Schema-Konzept	4
Abbildung 6: Die Beziehung der 3 Ebenen anhand eines Beispiels	5
Abbildung 7: Geschichte der Datenbanken	5
Abbildung 8: Einleitendes Beispiel	6
Abbildung 9: Unterteilung der Informationen in mehrere Tabellen	6
Abbildung 10: Entität	7
Abbildung 11: Entitätsmengen	7
Abbildung 12: Entität(smengen)	7
Abbildung 13: vier Arten von Assoziationen	7
Abbildung 14: Beispiel für Beziehungen (relationship)	8
Abbildung 15: Kombination von Assoziationen-Arten	8
Abbildung 16: Bezeichnungen im RM	8
Abbildung 17: Namensgebung von Entitäten	9
Abbildung 18: Entitäten	9
Abbildung 19: Entitäten mit Verbindungslinien (Beziehung)	9
Abbildung 20: Kardinalitäten im E/R-Diagramm (UML)	9
Abbildung 21: Alternative E/R-Diagramme	10
Abbildung 22: UML versus ER	10
Abbildung 23: Illustration der Kardinalitäten	10
Abbildung 24: Lösung Manual	11
Abbildung 25: Kriterien einer guten Definition	11
Abbildung 26: Beispiel für ein Formular „Entitätsdefinition“ (ausgefüllt)	12
Abbildung 27: Beispiel für ein Formular „Beziehungsdefinition“ (ausgefüllt)	12
Abbildung 28: Das relationale Modell	12
Abbildung 29: Bezeichnungen im RM	13
Abbildung 30: Beispiel	13
Abbildung 31: Überleitung eines ERD's in ein RM	13
Abbildung 32: Defintion „Primärschlüssel“	13
Abbildung 33: Defintion „Fremdschlüssel“	14
Abbildung 34: Vorgehensweise ERD -> RM	14
Abbildung 35: M:N Beziehung	15
Abbildung 36: M:N Beziehung (aufgelöst)	15
Abbildung 37: Organisationsstruktur	15
Abbildung 38: Stückliste	15
Abbildung 39: Redundanzen	16
Abbildung 40: Beispieltabelle für Anomalien	16
Abbildung 41: Beispiel-Tabelle „Funktionale Abhängig“	16
Abbildung 42: Beispieltabelle vorher	16
Abbildung 43: Beispieltabelle in 1NF	16
Abbildung 44: Beispieltabelle in 2NF	16
Abbildung 45: Beispieltabelle in 3NF	17
Abbildung 46: Zusammenfassung	17
Abbildung 47: Globale und Lokale Attribute	17

Abbildung 48: Beispiel Globale und Lokale Attribute	17
Abbildung 49: Lösung Globale und Lokale Attribute	17
Abbildung 50: Strukturregel 2	17
Abbildung 51: Definition Boyce Codd Normalform	17
Abbildung 52: Aggregation	17
Abbildung 53: Beispiel Aggregation	18
Abbildung 54: Beispiel Redundanzfreiheit	19
Abbildung 55: Vergleich Redundanzfreiheit	19
Abbildung 56: Datenbankintegrität	19
Abbildung 57: Typen von Integritätsbedingungen	19
Abbildung 58: Strukturregel 3	19
Abbildung 59: ad Fremdschlüssel	19
Abbildung 60: Rekursive Beziehung - Vorgesetzter	20
Abbildung 61: Rekursive Beziehung - Bestandteil	20
Abbildung 62: Rekursive Beziehung - Parallele Beziehungen	20
Abbildung 63: Rekursive Beziehung	20
Abbildung 64: Strukturregel 4	20
Abbildung 65: Auflösung durch zusätzliche Relation (Person)	21
Abbildung 66: Auflösung durch zusätzliche Relation (Abteilung-Person)	21
Abbildung 67: disjunkt und vollständig	21
Abbildung 68: disjunkt und nicht vollständig	22
Abbildung 69: überlappend und (nicht) vollständig	22
Abbildung 70: Generalisation als Beziehung	22
Abbildung 71: Darstellung in Tabellen	23
Abbildung 72: Die zeitliche Dimension	23
Abbildung 73: Problem „Fehlende Gegenbuchung“	23
Abbildung 74: Konto-Umbuchung	24
Abbildung 75: Lost Update	24
Abbildung 76: Uncommitted Dependency	24
Abbildung 77: Inconsistent Analysis	25
Abbildung 78: Arten von Locks	25
Abbildung 79: Lock-Kompatibilitätsmatrix	25
Abbildung 80: Lock-Varianten	26
Abbildung 81: Probleme gelöst?	26
Abbildung 82: Deadlock	27
Abbildung 83: Zeitmarken-Verfahren	27
Abbildung 84: Beispiel „Wait-Die“	27
Abbildung 85: Beispiel „Wound-Wait“	27
Abbildung 86: Inkonsistentes Lesen	28
Abbildung 87: Konsistentes Lesen	28
Abbildung 88: Begriffsabgrenzung	29
Abbildung 89: Beispiel aus der Praxis für Konsistenz	29
Abbildung 90: Im Fehlerfall kann das Programm verschiedenartig reagieren	29
Abbildung 91: Im Beispiele für Integritätsregeln	29
Abbildung 92: Realisierungsmöglichkeiten in ORACLE	30
Abbildung 93: Beispiel für user define	30
Abbildung 94: 3-Schema-Konzept	30
Abbildung 95: 3-Schema-Konzept	31
Abbildung 96: Zusammenhang Speichermedium – Datenorganisation	31

Abbildung 97: Organisationsformen	31
Abbildung 98: Sequentielle Suche	32
Abbildung 99: Schrittweise Suche	32
Abbildung 100: Binäres Suchen	32
Abbildung 101: Vollständiger Index	33
Abbildung 102: Blockindex	33
Abbildung 103: Blockindex mit Teilsortierung	33
Abbildung 104: Überlaufblock	34
Abbildung 105: Block-Split	34
Abbildung 106: trivialer Index	34
Abbildung 107: Funktionseigenschaften	35
Abbildung 108: Vor- und Nachteile der Randomorganisation	35
Abbildung 109: Nicht-injektive (Hash)Funktion	35
Abbildung 110: Beispiel Hashverfahren	35
Abbildung 111: Lineare Kollisionstrategie	36
Abbildung 112: Verkettete Kollisionstrategie	36
Abbildung 113: Two Pass Load	36
Abbildung 114: Sortierte Liste	37
Abbildung 115: Sortierte Liste 2	37
Abbildung 116: Physische Zeiger	37
Abbildung 117: Logische Zeiger	38
Abbildung 118: Invertierte Liste	38
Abbildung 119: Übungsbeispiel invertierte Liste	39
Abbildung 120: Mehrere Sekundärschl.	39
Abbildung 121: 1:n Teil 1	39
Abbildung 122: 1:n Teil 2	40
Abbildung 123: n:m Teil 1	40
Abbildung 124: n:m Teil 2	40
Abbildung 125: Elemente eines Baumes	41
Abbildung 126: Suchbaum	41
Abbildung 127: Binärer Suchbaum	41
Abbildung 128: B(2,1)-Baum	41
Abbildung 129: Suche im B-Baum	42
Abbildung 130: Beispiel – Einfügen in B-Baum	42
Abbildung 131: Beispiel – Löschen aus B-Baum	42

Viele der größeren Computersysteme, die in der Welt funktionieren, sind undurchschaubar, ich meine damit diejenigen, die in Militär, Staat und Wirtschaft genutzt werden.

Ich meine damit nicht, daß niemand davon etwas versteht, aber ganz und gar durchschauen kann man sie nicht mehr

Joseph Weizenbaum

Wir haben 2000 Datenbestände, die zu verarbeiten und zu pflegen sind

Wir sind uns bewußt, daß unsere Daten einen sehr hohen Redundanzgrad aufweisen und wir demzufolge einen unverhältnismäßig hohen Pflegeaufwand zu leisten haben.

Niemand in unserer Firma ist in der Lage, unsere Computerapplikationen zu überblicken.

Wir wissen nicht, wie wir unser Redundanzproblem in den Griff bekommen und eine saubere Basis für zukünftige Entwicklungen schaffen können.

Leitender Angestellter eines deutschen Versicherungsunternehmens

=> Datenchaos !!!

Abbildung 1: Einleitung

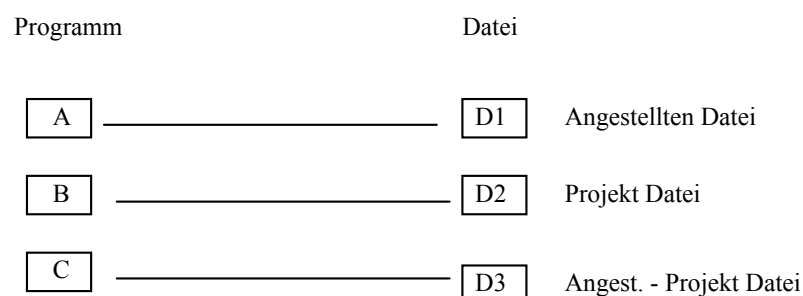


Abbildung 2: Dateisystem

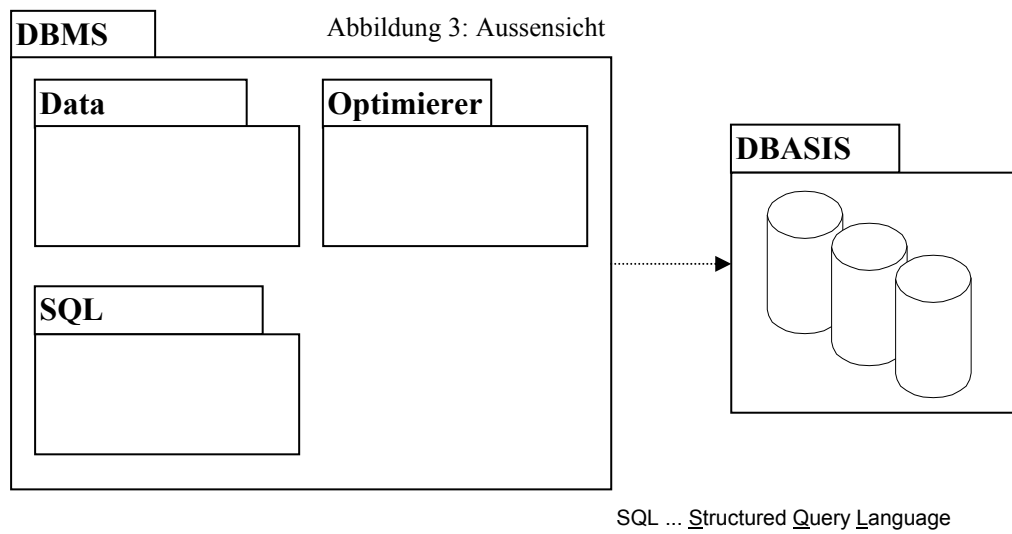
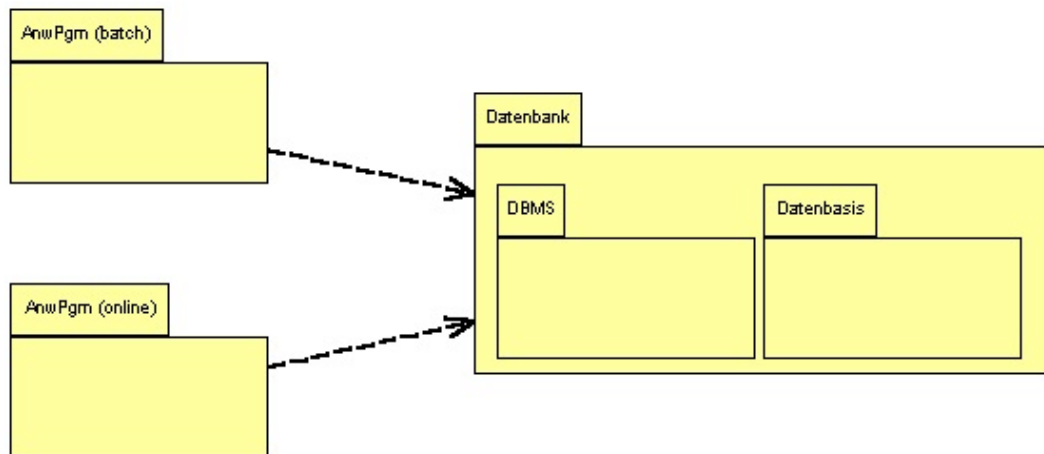


Abbildung 4: Innensicht

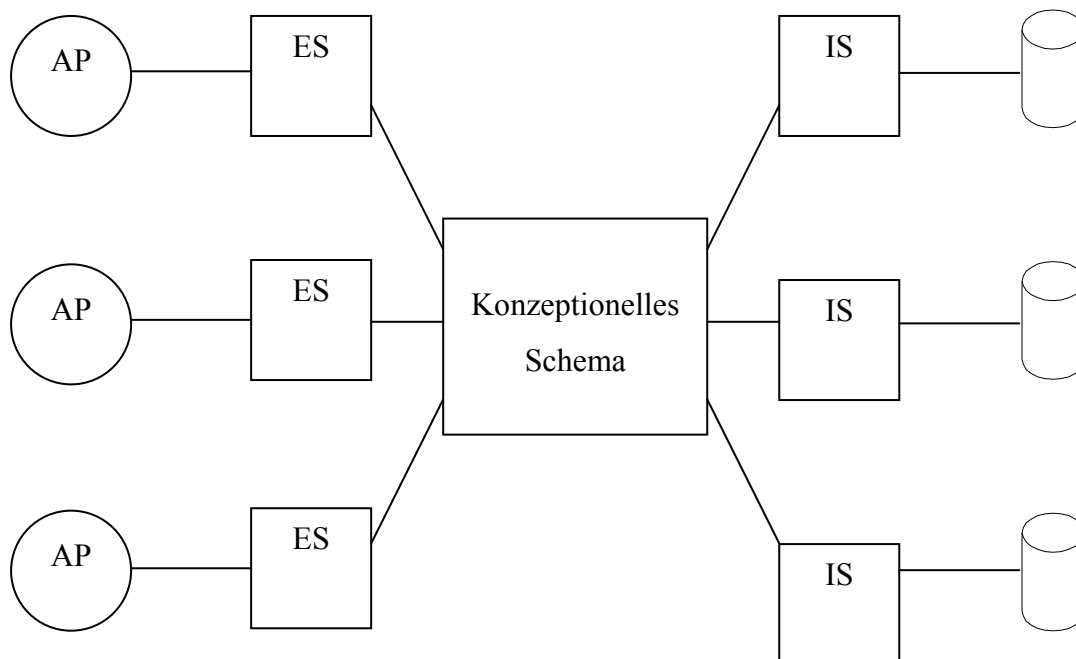


Abbildung 5: Schema-Konzept

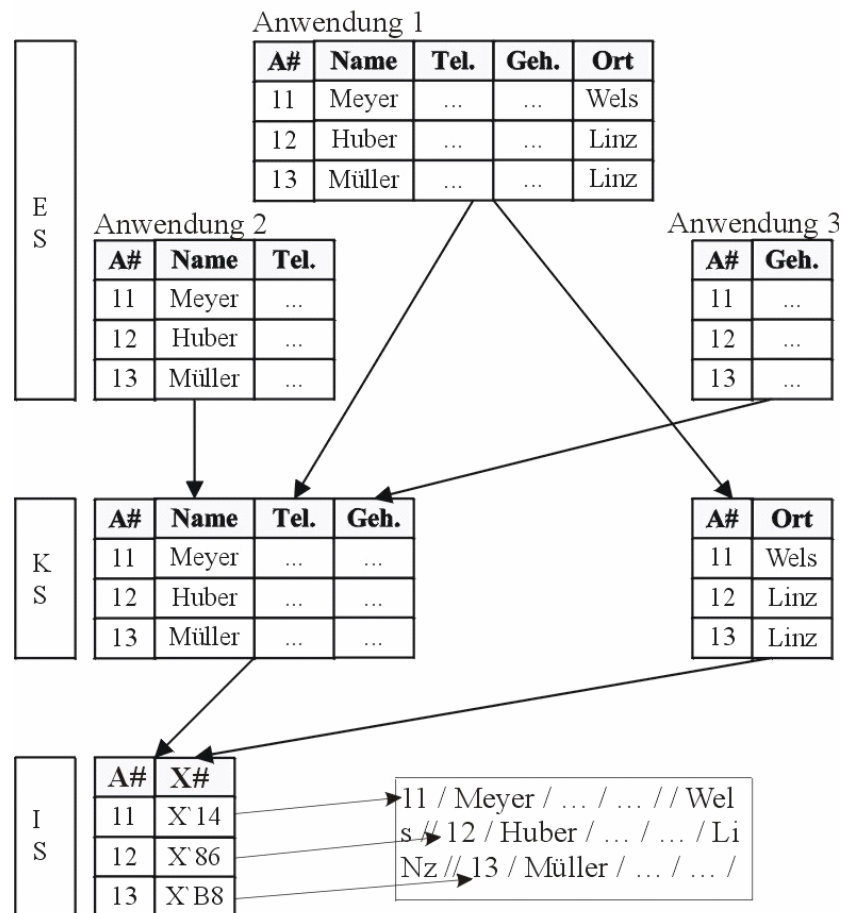


Abbildung 6: Die Beziehung der 3 Ebenen anhand eines Beispiels

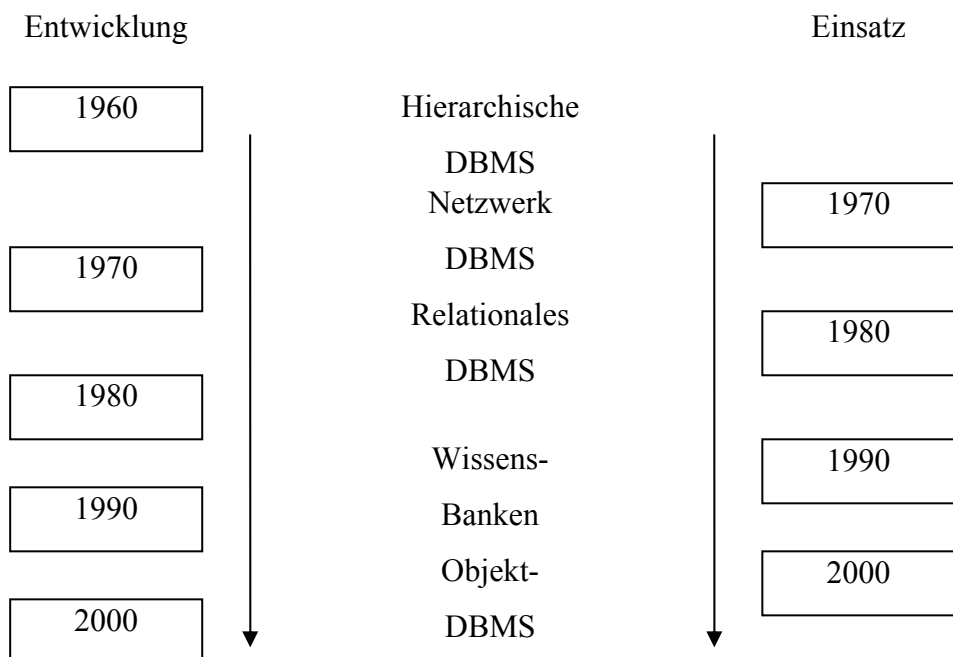


Abbildung 7: Geschichte der Datenbanken

Personalnummer	Name	Ort	Abteilung	Abteilungsname	Projekt nr	Projektname	Zeit
100	Susi	Linz	20	Org	23c	Kontobuchungen	40
101	Sabine	Wels	20	Org	23y 30a	OP Liste Umrechnung	50 5
102	Franz	Steyr	21	SW Entw.	23x 30a	Saldenliste Umrechnung	30 10
103	Otto	Linz	22	Test	23x 23y 30a	Saldenliste OP Liste Umrechnung	10 10 10

Abbildung 8: Einleitendes Beispiel

Projekt#	Projektname
23c	Kontobuchungen
23y	OP Liste
23x	Saldenliste
30a	Umrechnung

Pers#	Projekt#	Zeit
100	23c	40
101	23y	50
101	30a	5
102	23x	30
102	30a	10
103	23x	10
103	23y	10
103	30a	10

Pers#	Name	Ort	Abt#
100	Susi	Linz	20
101	Sabine	Wels	20
102	Franz	Steyr	21
103	Otto	Linz	22

Abt#	Abteilungsname
20	Org
21	SW Entw.
22	Test

Abbildung 9: Unterteilung der Informationen in mehrere Tabellen

Eine *Entität* ist ein individuelles und identifizierbares Exemplar von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt. Sofern eine Beziehung zwischen Entitäten eine Bedeutung hat, kann auch ein individuelles Exemplar einer solchen Beziehung als Entität aufgefaßt werden.

Abbildung 10: Entität

Ein wesentlicher Schritt bei der Modellbildung, d.h. bei der Abstraktion von konkreten Sachverhalten, besteht in der Gruppierung von Entitäten mit gleichen oder ähnlichen Merkmalen, aber unterschiedlichen Merkmalswerten zu *Entitätsmengen*.

Abbildung 11: Entitätsmengen

Disjunkte EM	Jede Entität in genau einer Entitätsmenge
	Bsp.: Personen und KFZ
Überlappende EM	Zumindest eine Entität in zumindest zwei EM
	Bsp.: Schüler und Fußballer
Unabhängige Entitäten	sind identifizierbar ohne eine weitere Entität.
	unabhängige Entität
	Fundamentale Entität
	Kernentität
	Bsp.: Kunden und Artikel.
Abhängige Entitäten	Existieren nur, wenn bestimmte Kernentität existiert
	Weak Entity
	Beispiele dafür sind Bestellung (abhängig von Artikel und Kunden) oder Zahlungen (abhängig von Konten).

Abbildung 12: Entität(smengen)

Assoziationstyp (EM1, EM2)	Entitäten aus EM2, die der Menge EM1 zugeordnet sind
1: einfache Assoziation	genau eine (1)
c: konditionelle Assoziation	keine oder eine (0, 1)
m: multiple Assoziation	mindestens eine (= 1)
mc: multiple-konditionelle Assoziation	keine, eine oder mehrere (= 0)

Abbildung 13: vier Arten von Assoziationen

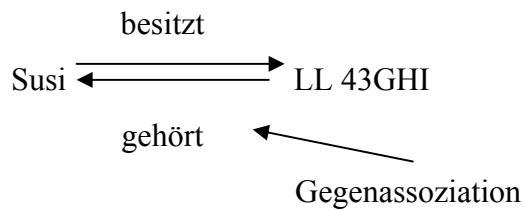


Abbildung 14: Beispiel für Beziehungen (relationship)

Entitätsmenge 1	Entitätsmenge 2	Beziehungstyp	Beziehung
rechte Schuhe	linke Schuhe	1 - 1	Paare
Abteilungen	Personal	c - 1	Abteilungsleiter
Personal	Abteilungen	m - 1	Abteilungszugehörigkeit
Kinder	Ehepaare	mc - 1	Familienzugehörigkeit
Frauen	Männer	c - c	Heirat
Personen	Parteien	m - c	Parteizugehörigkeit
Angestellte	Angestellte	mc - c	ist Vorgesetzter
Standorte	Standorte	m - m	Distanz
Vorlesungen	Studenten	mc - m	Einschreibung
Personen	Personen	mc - mc	Freundschaften

Abbildung 15: Kombination von Assoziationen-Arten

Entitätsmenge: „Person“
 Entität: Die Person mit den Namen
 „Tumfart“
 Attribut: „Name“
 Wertebereich: „Gehaltsgruppen“
 Eigenschaftswert: „Tumfart“

Attribute	→	Person				←	Entitätsmenge
		PersNr	Name	Plz	Ort		
Entität	→	27	Mair	4600	Wels		
		17	Tumfart	4020	Linz		
		45	Zopf	4020	Linz		

Abbildung 16: Bezeichnungen im RM

- stets im **Singular**
- wenn möglich **keine Abkürzungen**
- **Einheitlich** bleiben (und auch konsistent)
- **Begriffe der Geschäftswelt** verwenden
- **keine EDV Fachausdrücke** (Zielgruppe ist ja die Fachabteilung)
- **Keine Synonyme** (verschiedene Wörter für gleiche Begriffe)

zB.: BANK, GELDINSTITUT

- **Keine Homonyme** (1 Wort mit mehreren Bedeutungen)

zB.: BANK (Sitzbank, Geldinstitut)

Abbildung 17: Namensgebung von Entitäten



Abbildung 18: Entitäten

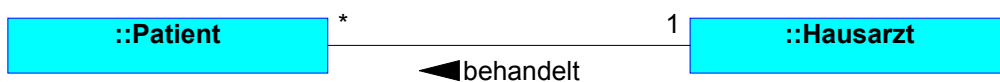


Abbildung 19: Entitäten mit Verbindungslinien (Beziehung)

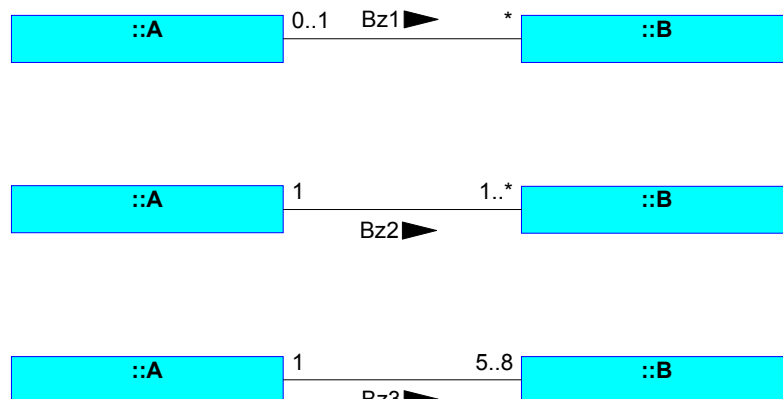


Abbildung 20: Karinalitäten im E/R-Diagramm (UML)

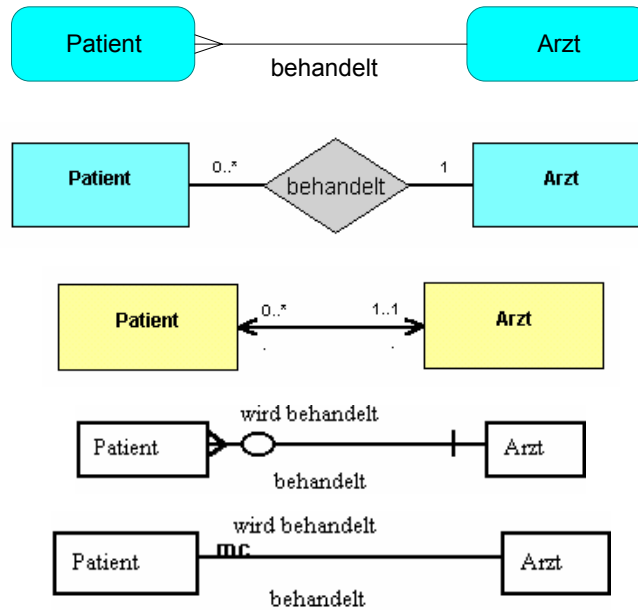


Abbildung 21: Alternative E/R-Diagramme

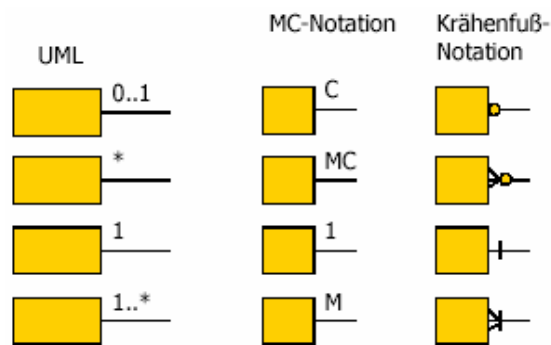


Abbildung 22: UML versus ER

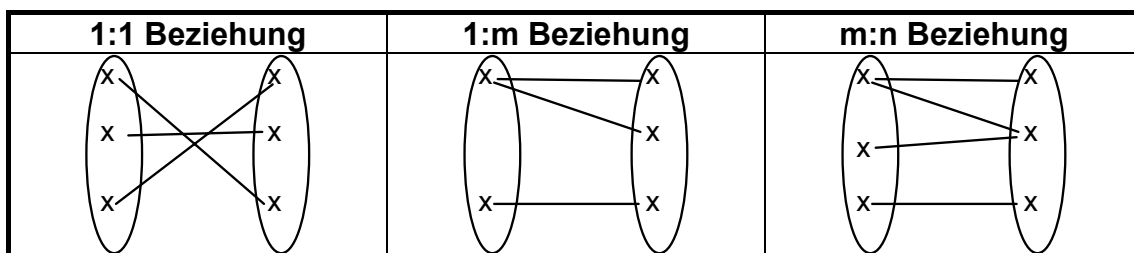


Abbildung 23: Illustration der Kardinalitäten

Stelle folgende reale Situation durch ein E/R-Diagramm dar:

- Ein Manual besteht aus dem Hauptwerk und eventuell aus weiteren Ergänzungsblättern.
- Ein Manual bezieht sich auf ein bestimmtes Fachgebiet
- Jedes Manual ist samt seinen Ergänzungsblättern an einem bestimmten Ort abgelegt.
- Ein Manual kann samt Ergänzungsblättern von den Mitarbeitern der Firma ausgeliehen werden.
- Ein Manual samt Ergänzungsblättern wird von einem bestimmten Lieferanten geliefert.

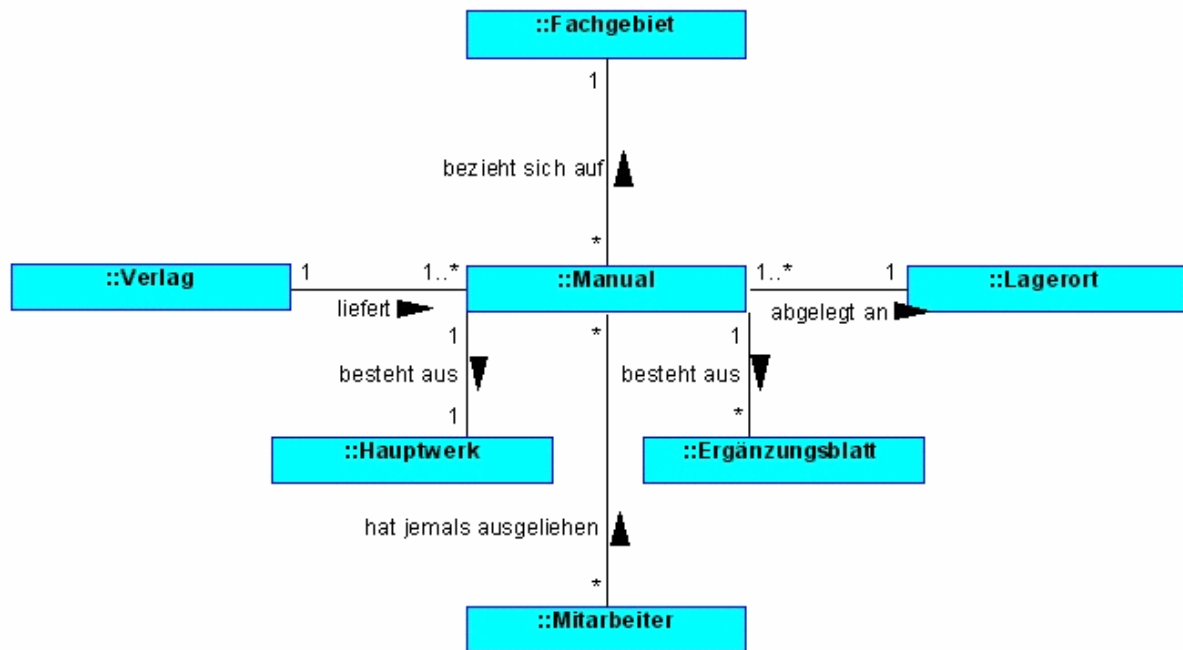


Abbildung 24: Lösung Manual

Klarheit und Kürze	Negativbeispiel: <i>„Kunde“ ist eine Gruppe, bestehend aus ein oder mehreren Personen, die nach außen als eine Unternehmung auftreten und mit der wir in Geschäftsbeziehung stehen, in dem wir ihnen Produkte verkaufen, die sie benötigen, um eine oder mehrere ihrer Geschäftsfunktionen erfüllen zu können.</i>
Vollständigkeit	<i>„Kunde“ ist eine Organisation, die unsere Produkte für den persönlichen Gebrauch einkauft.</i> <i>„Organisation“ und „Produkt“ muss ebenfalls definiert sein.</i>
Präzision	<i>„Einkauft“? Wann ist der Einkauf vollzogen?</i> <i>... Ein Verkauf gilt als vollzogen, wenn der Vertrag unterschrieben ist - also unabhängig von Zahlung und Geldverkehr.</i>
Konsistenz	Negativbeispiel: <i>Kunde = Organisation, die unsere Produkte kauft.</i> <i>Wiederverkäufer = Organisation, die von uns Produkte einkauft, um diese an den Kunden weiterzuverkaufen.</i> Positivbeispiel: <i>Kunde = Organisation, die unsere Produkte für den persönlichen Gebrauch einkauft.</i> <i>Wiederverkäufer = Organisation, die unsere Produkte kauft, um diese weiterzuverkaufen.</i>

Abbildung 25: Kriterien einer guten Definition

ENTITÄT: KUNDE ANALYTIKER:	KONTAKTPERSON: ORG.EINHEIT: DATUM: VERSION:
DEFINITION: eine Organisation, die zumindest eines unserer Produkte gekauft hat ENTITÄTSTYP: fundamental GESCHÄFTSREGELN: ein Produkt gilt als verkauft, wenn eine definitive Bestellung eingegangen ist EINSCHRÄNKUNGEN, AUSNAHMEN: Organisationen, die eine Produktprobe erhalten haben, gelten auch als Kunde EXISTENZBEDINGUNGEN, REGELN: <ul style="list-style-type: none"> • Erstellen eine Organisation wird zum Kunden, wenn eine gültige Bestellung einem Verkäufer übergeben wurde • Löschen führt die Bestellung dennoch zu keinem Kauf, so wird die Organisation nicht als Kunde betrachtet • Integritätsregeln wenn ein Kunde gelöscht wird, so dürfen keine offene Bestellungen vorhanden sein ANZAHL DER AUSPRÄGUNGEN: durchschnittlich: 1000 min. max: zwischen 200 und 5000 Wachstumsrate: 100 pro Jahr, gleichverteilt	

Abbildung 26: Beispiel für ein Formular „Entitätsdefinition“ (ausgefüllt)

RELATION: Kunde kauft Produkt	KONTAKTPERSON: Fr. Mauer
ANALYTIKER: Hr. Knoll	ORG.-EINHEIT: Verkauf
DATUM: 24.10.90 VERSION: 2.0	
DEFINITION, ZWECK: Ein Kunde kauft ein Produkt ein, wenn eine Versandbestätigung über das Produkt ausgefolgt wurde. dazugehörige ENTITÄTEN: Kunde, Produkt VON - NACH - BEZIEHUNG: Kunde kauft Produkt Kardinalität: min: 0 max: m UMGEKEHRTE BEZIEHUNG: Produkt wird gekauft von Kunde Kardinalität: min: 0 max: m INTEGRITÄTSREGELN: Die Versandbestätigung muß von einem autorisierten Verkäufer ausgestellt worden sein.	

Abbildung 27: Beispiel für ein Formular „Beziehungsdefinition“ (ausgefüllt)

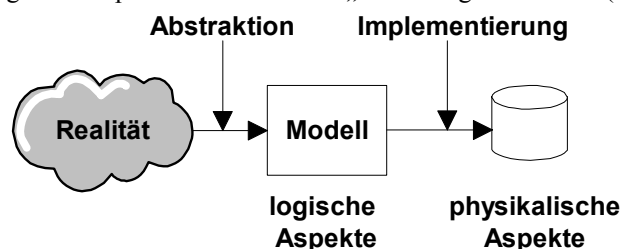


Abbildung 28: Das relationale Modell

Entitätsmenge:

PERSON

Relation(enschema)

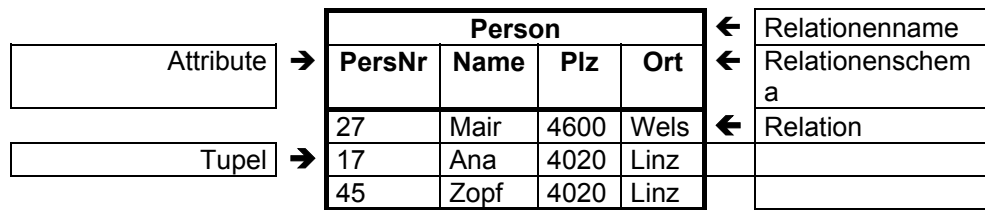


Abbildung 29: Bezeichnungen im RM

Beispiel: $W_1 = \{1,2,3\}; W_2 = \{A,B\}$ $W_1 \times W_2 = \{(1,A),(1,B),(2,A),(2,B),(3,A),(3,B)\}$ $R = \{(1,A),(2,B)\} \subseteq W_1 \times W_2$	Beispiel2: Name: {Meier, Schmidt} Ort: {Hamburg, München}		
	Meier	Hamburg	
	Meier	München	
	Schmidt	Hamburg	
	Schmidt	München	

Abbildung 30: Beispiel

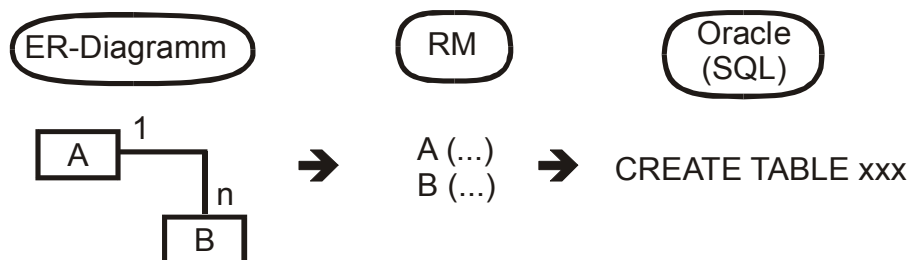


Abbildung 31: Überleitung eines ERD's in ein RM

Jede Tabelle hat mindestens einen Schlüsselkandidaten (Candidate Key), der aus einem oder mehreren Attributen zusammengesetzt wird und dessen Wertekombination den eindeutigen Zugriff auf eine Zeile der Tabelle ermöglicht. Würde man ein Attribut daraus entfernen, so würden diese Attribute nicht mehr eindeutig identifizierend sein; er ist in gewisser Hinsicht minimal (z.B. Name oder PersNr).

Ein Primärschlüssel (Primary Key) ist eine bestimmter, willkürlich festgelegter Schlüssel der Schlüsselkandidaten (z.B. PersNr).

Oft gibt es für eine Relation mehr als einen Candidate Key. Solche, die nicht als Primary Key definiert werden, nennt man Alternate Keys (z.B. Name).

Abbildung 32: Definition „Primärschlüssel“

Ein Menge von Attributen, die in einer anderen Tabelle den Primärschlüssel bildet, wird als Fremdschlüssel (foreign key) bezeichnet.

Abbildung 33: Definition „Fremdschlüssel“


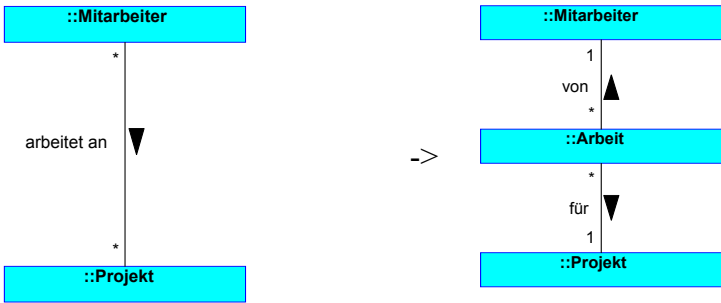
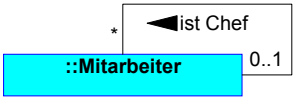
Schritt	Erklärung, Beispiel, Anmerkung, ...
1 Jede Entitätsmenge wird Relation mit einem Primärschlüssel	Strukturregel 1 (SR1): Bei der Darstellung von Entitätsmengen durch Relationen muss für jede Relation ein Primärschlüssel existieren. Z.B.: EM Mitarbeiter -> Mitarbeiter(<u>PersNr</u> , ...) EM Projekt -> Projekt(<u>ProjektId</u> , ...) EM Kunde -> Kunde(<u>KundeId</u> , ...)
2 1:n Beziehungen werden Fremdschlüssel	Als Fremdschlüssel in der abhängigen Relation (n-Seite) wird der Primärschlüssel der unabhängigen Relation (1-Seite) eingefügt; er verweist daher auf eine Zeile (Row) in dieser Relation ähnlich einem Pointer oder einem Link. Z.B.: Beziehung Projekt-Kunde -> Projekt(..., KundeId, ...) 
3 m:n Beziehungen werden assoziative Tabellen	Die neu gebildete assoziative Relation enthält die Primärschlüssel der beiden in Beziehung stehenden Relationen. Sie erhält ihren Primärschlüssel entweder direkt als Kombination der beiden Fremdschlüssel oder es wird („künstlich“) ein neuer Primärschlüssel zugeteilt; die beiden Fremdschlüssel werden dann als Nichtschlüsselattribute verwendet. Z.B.: Beziehung Projekt-Mitarbeiter -> Arbeit(..., ProjektId, PersNr, ...) 
4 Eigenschaften werden Spalten	Alle Eigenschaften werden als Attribute abgebildet; die Eigenschaftswerte finden sich dann in der Spalte „darunter“. Z.B.: Mitarbeiter(..., Name, Vorname, Adresse, Plz, Ort, ...) Projekt(..., Bezeichnung, Startdatum, Endedatum, ...) Kunde(..., Name1, Name2, Adresse, Plz, Ort, Umsatz, ...) Arbeit(..., AnteilArbeitszeit, ...)
5 Jede Beziehung wird zu einem Fremdschlüssel	Auch 1:1 – Beziehungen und insbesondere rekursive Beziehungen werden als Fremdschlüssel abgebildet. Z.B.: Beziehung „ist Chef von“ -> Mitarbeiter(..., ChefId, ...) 
6 Normalisierung	Folgt später!
7 Aggregation	Folgt später!

Abbildung 34: Vorgehensweise ERD -> RM

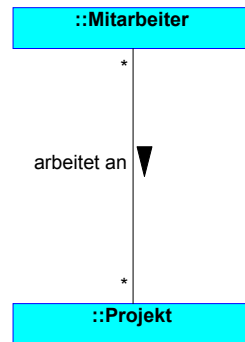


Abbildung 35: M:N Beziehung

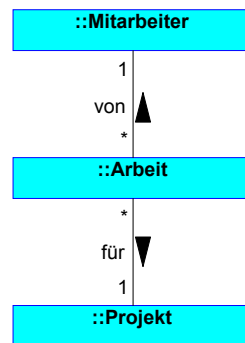
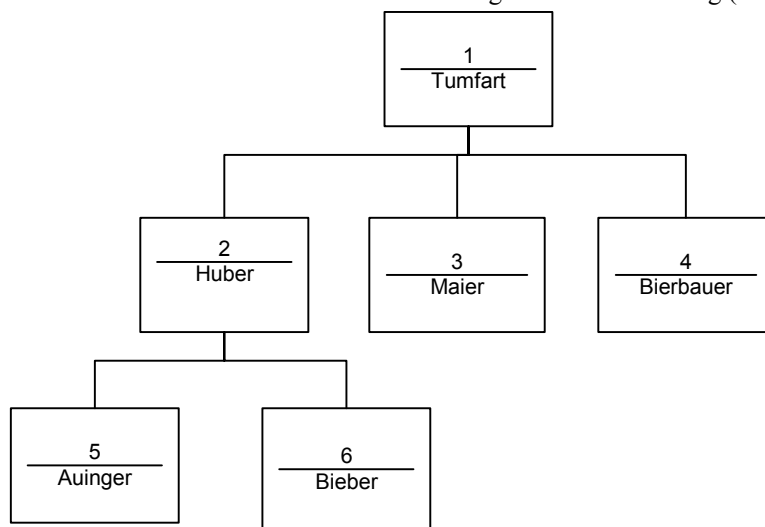
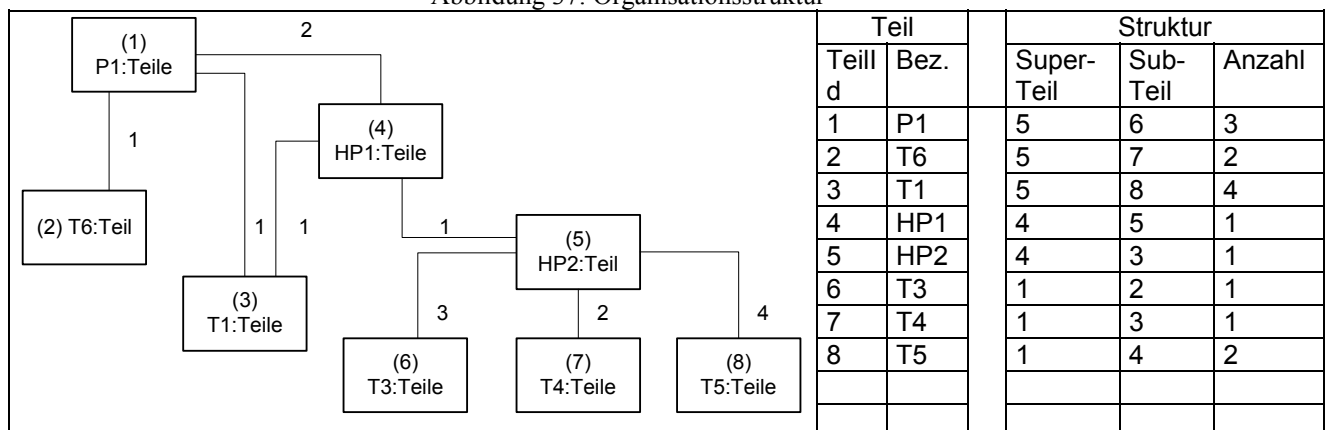


Abbildung 36: M:N Beziehung (aufgelöst)



Stelleld	Super-Stelleld	Name
6	2	Bieber
5	2	Auinger
2	1	Huber
3	1	Maier
4	1	Bierbauer
1	1	Tumfart

Abbildung 37: Organisationsstruktur



Teil		Struktur		
Teill d	Bez.	Super-Teil	Sub-Teil	Anzahl
1	P1	5	6	3
2	T6	5	7	2
3	T1	5	8	4
4	HP1	4	5	1
5	HP2	4	3	1
6	T3	1	2	1
7	T4	1	3	1
8	T5	1	4	2

Abbildung 38: Stückliste

Redundanz ist in einem Datenbestand genau dann vorhanden, wenn ein Teil des Bestandes (redundante Information) ohne Informationsverlust weggelassen werden kann. Auf der konzeptionellen Ebene ist Redundanz grundsätzlich zu vermeiden, einmal wegen des Speicheraufwandes, besonders aber, weil *Anomalien* auftreten können.

Abbildung 39: Redundanzen

Studenten#	Vorlesungs#	Studentenname	Studentenadresse	Vorlesungsname
S21	8725	Josef	Linz	Datenbank
S21	8730	Josef	Linz	PL/1
S33	8725	Maria	Wels	Datenbank

Abbildung 40: Beispieltabelle für Anomalien

PersNr	Vorname	Wohnsitz
1	Hans	Linz
2	Hans	Linz
3	Elke	Wels
4	Manfred	Wels
5	Sabine	Steyr
1	Hans	Steyr
4	Manfred	Steyr

Abbildung 41: Beispiel-Tabelle „Funktionale Abhängig“

PeNr	Name	AbtNr	AbtName	PrNr	PrName	Zeit
100	Hans	1	Physik	11,12	A,B	60,40
101	Paul	2	Chemie	13	C	100
102	Georg	2	Chemie	11,12,13	A,B,C	20,50,30

Abbildung 42: Beispieltabelle vorher

Projekt						
PeNr	Name	AbtNr	AbtName	PrNr	PrName	Zeit
100	Maier	1	Physik	11	A	60
100	Maier	1	Physik	12	B	40
101	Müller	2	Chemie	13	C	100
102	Huber	2	Chemie	11	A	20
102	Huber	2	Chemie	12	B	50
102	Huber	2	Chemie	13	C	30

Abbildung 43: Beispieltabelle in 1NF

Person				Projekt		Arbeitszeit		
PeNr	Name	AbtNr	AbtName	PrNr	PrName	PeNr	PrNr	Zeit
100	Maier	1	Physik	11	A	100	11	60
101	Müller	2	Chemie	12	B	100	12	40
102	Huber	2	Chemie	13	C	101	13	100
						102	11	20
						102	12	50
						102	13	30

Abbildung 44: Beispieltabelle in 2NF

Person			Abteilung		Projekt		Arbeitszeit		
PeNr	Name	AbtNr	AbtNr	AbtName	PrNr	PrName	PeNr	PrNr	Zeit
100	Maier	1	1	Physik	11	A	100	11	60
101	Müller	2	2	Chemie	12	B	100	12	40
102	Huber	2			13	C	101	13	100
							102	11	20
							102	12	50
							102	13	30

Abbildung 45: Beispieltabelle in 3NF

Primärschlüssel Alle Attribute vom Schlüssel abhängig.

:

1NF: Nur skalare Attribute.

2NF: Kein Attribut von Teilschlüssel abhängig.

3NF: Kein Attribut von anderem Nicht-Schlüsselattribut abhängig.

Abbildung 46: Zusammenfassung

Ein Attribut heißt global, wenn es mindestens in einer Relation im Primärschlüssel vorkommt.

Ein Attribut heißt lokal, wenn es nur in einer Relation und dort nicht im Primärschlüssel vorkommt.

Abbildung 47: Globale und Lokale Attribute

Beispiel:

Student	(MatrikelNr , SName, Plz, Ort, Str)
Mitarbeiter	(SVNr , MName, Plz, Ort, Str, Lohnklasse)

Abbildung 48: Beispiel Globale und Lokale Attribute

Person	(PNr , Name, Plz, Ort, Str)
Student	(PNr , MatrikelNr)
Mitarbeiter	(PNr , SVNr, Lohnklasse)

Abbildung 49: Lösung Globale und Lokale Attribute

Strukturregel 2 (SR2): Die Datenbank muss aus Relationen in dritter Normalform bestehen, welche nur Global- und Lokalattribute enthalten.

Abbildung 50: Strukturregel 2

Ein Attribut A wird Determinante genannt, wenn von A ein anderes Attribut B voll funktional abhängig ist.

Eine Relation ist in Boyce-Codd Normalform, wenn jedes Attribut, das eine Determinante ist, auch als Schlüssel verwendet werden könnte.

Eine Relation ist dann in BCNF, wenn alle Determinanten zugleich Schlüsselkandidaten sind.

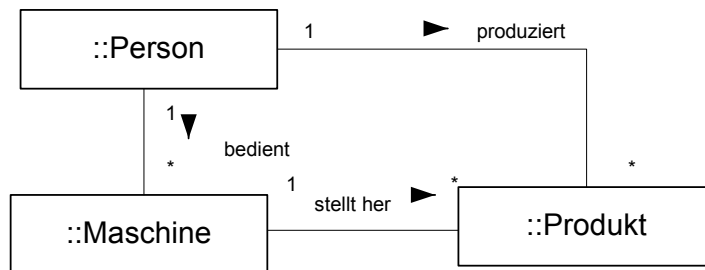
Abbildung 51: Definition Boyce Codd Normalform

Schema:	
ElemRel1	(<u>TSA1</u> , <u>TSA2</u> , NSA1)
ElemRel2	(<u>TSA1</u> , <u>TSA2</u> , NSA2)
AggregRel	(<u>TSA1</u> , <u>TSA2</u> , NSA1, NSA2)

Abbildung 52: Aggregation

Beispiel (aus Übung 1):

- Eine Person bedient mehrere Maschinen und produziert dabei mehrere Produkte.
- Eine Maschine wird immer nur von einer bestimmten Person bedient, kann aber mehrere Produkte produzieren.
- Die Herstellung eines Produktes erfordert immer nur eine Maschine sowie eine Person.

1) E/R-Diagramm**Maschine - Produkt - Person**2) Unmittelbar lassen sich folgende Tabellen ableiten:A: PERS(Pe#,...)B: MASCH(Masch#,...)C: PROD(Pr#,...)3) Die 1:N Beziehungen werden mit Hilfe von Fremdschlüsseln aufgelöst:D: MASCH_PERS(Masch#,Pe#,...)E: PROD_PERS(Pr#,Pe#,...)F: PROD_MASCH(Pr#,Masch#,...)4) Tabellen mit identischem Primärschlüssel werden zusammengefaßt:Aus C,E und F: PROD(Pr#, Pe#, Masch#,...)Aus B und D: MASCH(Masch#, Pe#,...)5) Verstoß gegen die 3.NF in der Tabelle PROD, da Pe# von Masch# funktional abhängig ist.

Daher Normalisierung:

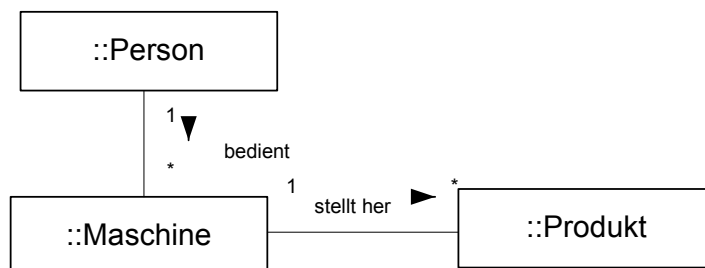
PROD(Pr#, Masch#,...)MASCH_PERS(Masch#, Pe#,...)6) Da die Tabellen MASCH_PERS und MASCH den gleichen Sachverhalt festhalten, kann auf die Tabelle MASCH_PERS verzichtet werden.Daraus ergibt sich eine Änderung des ERD:**Maschine - Produkt - Person**

Abbildung 53: Beispiel Aggregation

KUNDE (Kunden#, Stammdaten, ...)

KONTO (Konto#, Kunden#, ...)

Abbildung 54: Beispiel Redundanzfreiheit

Konto suchen	Konto suchen
Kunde suchen	SKz überprüfen
SKz überprüfen	Bewegung durchführen
Bewegung durchführen	
= 4 Operationen / 2 Zugriffe	= 3 Operationen / 1 Zugriff

Abbildung 55: Vergleich Redundanzfreiheit

Datenbankintegrität ist die „Richtigkeit“ der Daten in der Datenbank. Die Integrität muß nach jeder Datenbankoperation bzw. Transaktion erfüllt sein.

Integritätsbedingungen (integrity rules) beschreiben diesen Zustand, werden meist bei der Tabellenerstellung angegeben und das DBMS sorgt dafür, dass sie zu jedem Zeitpunkt eingehalten werden.

Abbildung 56: Datenbankintegrität

Typen	Kurzbeschreibung	Beispiel
Die Wertebereichs-Integrität (domain integrity)	a) einem gewissen Datentyp entspricht, b) NULL werden darf (oder eben nicht), c) zusätzlichen Bedingungen	a) Datentypen number, char, boolean, ... b) not null, c) check (deptno between 10 and 99)
Die Primärschlüssel-Integrität (entity integrity)	Primärschlüssel darf keine Nullwerte beinhalten und muss eindeutig sein.	primary key(playerno)
Die Fremdschlüssel-Integrität (referential integrity)	Jeder Fremdschlüsselwert muss in der entsprechenden Tabelle als Primärschlüssel enthalten sein.	foreignkey(playerno) references players(playerno)
Benutzerspezifische Integritätsregeln (user defined integrity)	Zusätzlich zu den erwähnten Integritätsregeln kann es notwendig sein benutzerdefinierte Regeln aufzustellen.	

Abbildung 57: Typen von Integritätsbedingungen

Strukturregel 3 (SR3): - Lokal-Attribute müssen statische Wertebereiche verwenden.

- Jedes Global-Attribut darf nur in einer Relation auf einem statischen Wertebereich basieren und muss in dieser Relation Primärschlüssel sein. In allen anderen Relationen muss es auf einem dynamischen Wertebereich basieren (d.h. als Fremdschlüssel zu einer anderen Relation)

Abbildung 58: Strukturregel 3

Person			
SVNr	Name	Ort	Abt#
101	Hans	Linz	1
102	Paul	Wels	1
103	Kurt	Linz	2

Abteilung	
Abt#	AName
1	Sales
2	Buchhaltung
3	Programmierung

Abbildung 59: ad Fremdschlüssel

Person

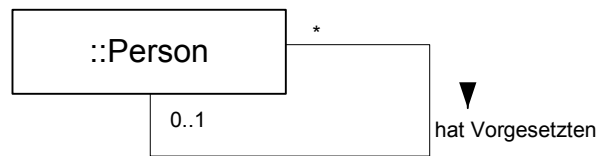


Abbildung 60: Rekursive Beziehung - Vorgesetzter

Teil

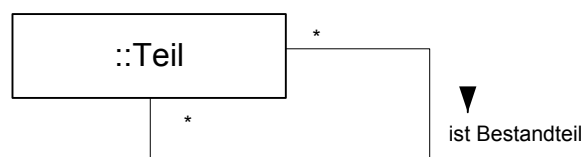


Abbildung 61: Rekursive Beziehung - Bestandteil

Abteilung - Person

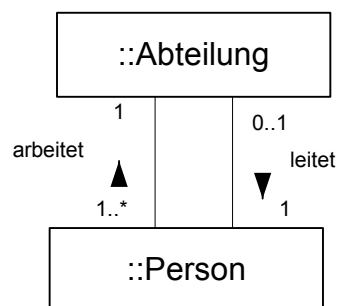


Abbildung 62: Rekursive Beziehung - Parallele Beziehungen

- ① PERSON (PersNr, ChefNr,)
- ② TEIL (TeilNr,)
STUECKLISTE (HauptTeilNr, UnterTeilNr,)
- ③ ABTEILUNG (AbtNr, AbtLeiterNr,)
PERSON (PersNr, AbtNr,)

Abbildung 63: Rekursive Beziehung

Strukturregel 4 (SR4): Rekursive Beziehungen zwischen Relationen sind untersagt; ein Globalattribut in einer Relation darf nur mit einem solchen Fremdschlüssel gebildet werden, dessen Ausgangsrelation unabhängig von dieser Relation definiert werden kann.

Abbildung 64: Strukturregel 4

①

PersonPERSON (PersNr, ChefNr,)CHEF (ChefNr,)

Abbildung 65: Auflösung durch zusätzliche Relation (Person)

③

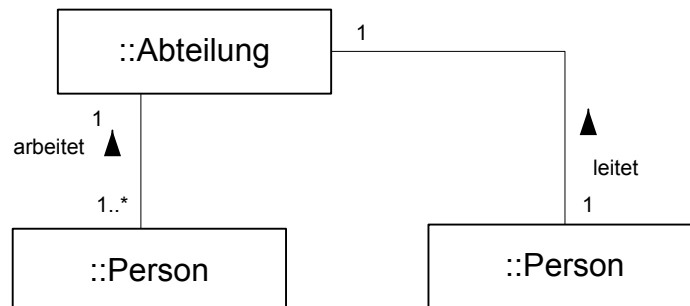
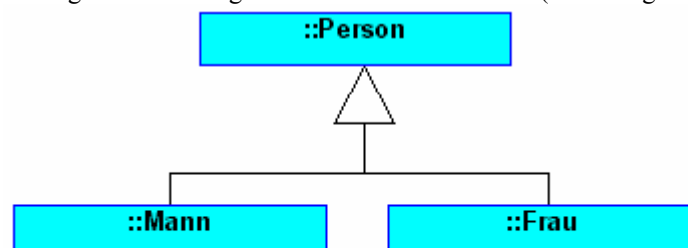
Abteilung - PersonABTEILUNG (AbtNr, AbtLeiterNr,)ABTEILUNGSLEITER (LeiterNr,)PERSON (PersNr, AbtNr,)

Abbildung 66: Auflösung durch zusätzliche Relation (Abteilung-Person)



disjunkt: entweder "Mann" oder "Frau"
 vollständig: keine "Person" weder "Mann" noch "Frau"

Abbildung 67: disjunkt und vollständig

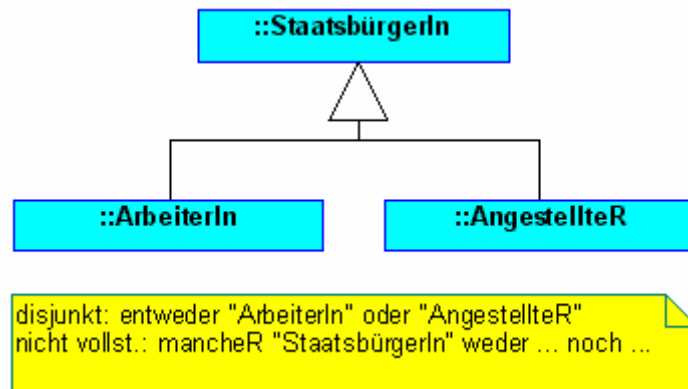


Abbildung 68: disjunkt und nicht vollständig

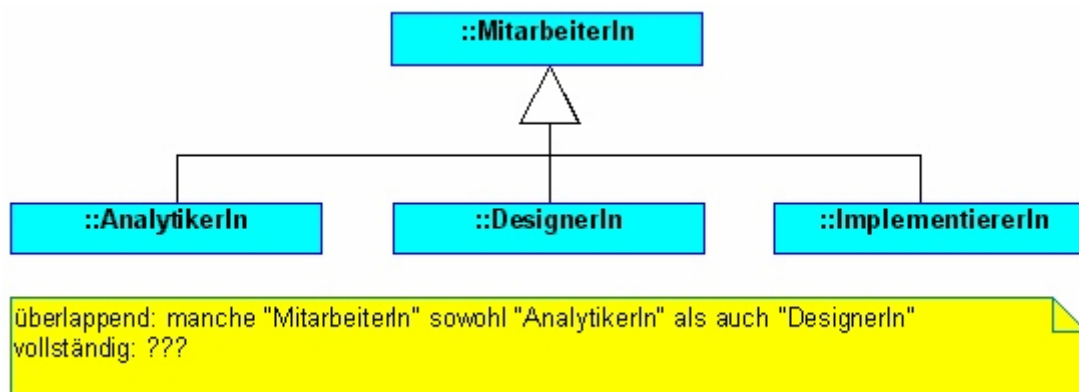


Abbildung 69: überlappend und (nicht) vollständig

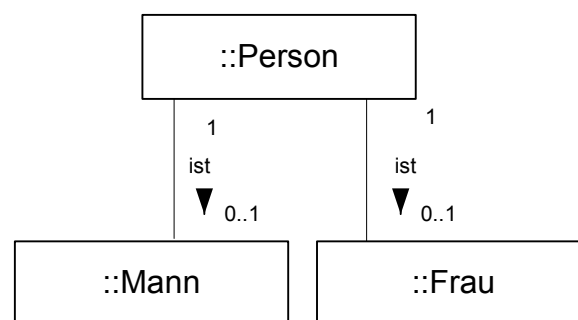


Abbildung 70: Generalisation als Beziehung

Jede Entität wird eine eigene Tabelle:

Mit Sicherheit die flexibelste Art der Implementierung, da a) die Subtypen sowohl voneinander als auch vom Supertyp unabhängig sind und b) jederzeit ein Subtyp hinzugefügt / gelöscht werden kann. Außerdem fallen in dieser Lösung keine Null-Werte an – sie ist somit als die sauberste anzusehen.

Zur vereinfachten Navigation kann bei disjunkten Subtypen auch ein Attribut eingeführt werden, in dem (redundant!) der Subtyp gespeichert wird.

Attribute des Supertyps in den Subtypen:

Man gewinnt dadurch in vielen Fällen an Performance, bezahlt dies jedoch mit der Redundanz bei überlappenden Generalisationen. Somit ist diese Art der Auflösung bei a) disjunkten Subtypen und b) Supertypen mit wenigen Attributen sinnvoll.

Zu beachten ist auch, dass plötzlich alle Staatsbürger nicht mehr in einer Tabelle gespeichert werden – will man auf diese zugreifen, ist jedes Mal eine Union-Operation nötig.

Attribute der Subtypen im Supertyp:

Hinsichtlich vieler Zugriffsarten scheint diese Lösung optimal zu sein (Ausnahme: Verarbeitung aller Entities eines Subtyps), leider bedingt dieses Design die extensive Verwendung von NULL, was eigentlich in Hinblick auf die schwere Verständlichkeit von Codd untersagt wird: wenn wir annehmen, dass es zehn Subtypen gibt, dann müssen für jeden Subtypen die Attribute aller neun anderen Subtypen NULL gesetzt werden!

Und, und, und ...

```
StaatsbürgerIn
(Sv#, Subtyp,
 NName, VName, ...)
ArbeiterIn
(Sv#, Lohn, ...)
AngestellterR
(Sv#, Gehalt, ...)
```

```
ArbeiterIn
(Sv#, NName,
 VName, Lohn, ...)
AngestellterR
(Sv#, NName,
 VName, Gehalt,
 ...)
```

```
StaatsbürgerIn
(Sv#, Subtyp,
 NName, VName,
 Lohn, Gehalt, ...)
```

Abbildung 71: Darstellung in Tabellen

- Zeitpunktbezogen: Für diskrete Informationen eignet sich eine zeitpunktsbezogene Speicherung der Art: Kontobewegung (KontoId, Zeitpunkt, Betrag, ...)
- Zeitraumbezogen: Für sprunghaft wechselnde Informationen (zustandserhaltend unstetig) eignet sich eine Speicherung wie o.a.; weiteres Beispiel: Kontostand (KontoId, GültigVon, GültigBis, Saldo, ...)
- Versionsbezogen: Will man jede einzelne Version des Entities benennen, so bietet sich eine versionsbezogene Speicherung an: Modul (ModulId, Version, ...) – dies bildet einen ähnlichen Sachverhalt wie die zeitraumbezogene Speicherung ab.

Abbildung 72: Die zeitliche Dimension

update Konto	update Konto
set Saldo = Saldo – 500	set Saldo = Saldo + 500
where KontoNr = 4711	where KontoNr = 0815

Abbildung 73: Problem „Fehlende Gegenbuchung“


```

exec sql  whenever sqlerror goto UnDo;
...
Betrag = ...
KontoQuelle = ...
KontoZiel = ...
...
exec sql  update Konto
          set Saldo = Saldo – Betrag
          where KontoNr = KontoQuelle
...
exec sql  update Konto
          set Saldo = Saldo + Betrag
          where KontoNr = KontoZiel
...
exec sql  commit work;
...
UnDo:
exec sql  rollback work;

```

Abbildung 74: Konto-Umbuchung

Zeit	Transaktion A		Preis	Transaktion B	
1	Transaktion liest einen bestimmten Wert „Preis“ (100)...	select	100,-	-	-
2	... um ihn zu analysieren und dann ...	-	100,-	select	In der Transaktion wird genau das gleiche Anwendungsprogramm durchgeführt: also Preis lesen (100)...
3	... abhängig von vielen Faktoren zu verändern. In diesem Fall soll es eine Erhöhung um 10% sein.	update	110,-	-	... analysieren ...
4	120,-	update	... und um 20% erhöhen.
5

Abbildung 75: Lost Update

Zeit	Transaktion A		Preis	Transaktion B	
0	-	-	100,-	-	-
1	Die Transaktion verändert den Preis (120) ...	update	120,-	-	-
2	... kommt in einen Ausnahmestand ...	-	120,-	select	Die Transaktion liest denn Preis (120) ...
3	... und macht die Veränderung wieder rückgängig um in einen konsistenten Zustand zu kommen.	rollback	100,-	-	... um ihn zu analysieren und zu verarbeiten.
4	-	-	100,-	-	-
5	-	-	100,-	... 120 ...	Obwohl inzwischen die Veränderung schon wieder zurückgenommen worden ist, rechnet die Transaktion mit 120 statt mit 100!

Abbildung 76: Uncommitted Dependency

Zeit	Transaktion A		Kto 1	Kto 2	Transaktion B	
0	-	-	100,-	200,-	-	-
1	Transaktion summiert alle Kontostände: sie liest Kontostand 1 (100,-) ...	select	100,-	200,-	-	-
2	-	-	100,-	200,-	select	Transaktion bucht 10,- von Konto 2 auf Konto 1: sie liest Kontostand 2 (200) ...
3	-	-	100,-	190,-	update	... verändert ihn, ...
4	-	-	100,-	190,-	select	... liest Kontostand 1 ...
5	-	-	110,-	190,-	update	... und verändert auch diesen.
6	-	-	110,-	190,-	commit	Ein commit bestätigt das Ganze – aus Sicht dieser Transaktion ist alles ok!
7	... und Kontostand 2 (190,-) ...	select	110,-	190,-	-	-
8	... und gibt die falsche (!) Summe (290,-) aus!		110,-	190,-	-	-

Abbildung 77: Inconsistent Analysis

Exclusive Lock (X Lock):	Falls die Transaktion A den Datensatz R durch einen X Lock sperrt, wird jeder Lock-Versuch einer anderen Transaktion für den gleichen Datensatz abgelehnt - egal, ob die andere Transaktion einen X Lock oder S Lock versucht. Die letztere Transaktion wird in einen Wartezustand versetzt: sie wartet, bis der Datensatz R von A freigegeben wird.
Shared Lock (S Lock):	Falls eine Transaktion A einen S Lock für den Datensatz R aufrechterhält, muss jede andere Transaktion warten, die einen X Lock anfordert - bis A das Objekt freigibt; S Locks werden jedoch gewährt. Es können also mehrere Transaktionen auf ein und dasselbe Objekt einen S Lock unterhalten.

Abbildung 78: Arten von Locks

Zustand \ Versuch	X Lock	S Lock	unlocked
X Lock	✗	✗	✓
S Lock	✗	✓	✓

Abbildung 79: Lock-Kompatibilitätsmatrix

Beispiel:

Betrachten wir eine Überweisung von 100,- von Konto A auf ein Konto B:

Variante 1	Lock A B	Variante 2	Lock A B
transaction Ü1 begin		transaction Ü2 begin	
X-Lock (KtoA);		S-Lock (KtoA);	
X-Lock (KtoB);	X		
if KtoA > 100 then	X X	if KtoA > 100 then	S
begin		begin	
KtoA = KtoA - 100;	X X	X-Lock (KtoA);	S
KtoB = KtoB - 100;	X X	X-Lock (KtoB);	X
end		KtoA = KtoA - 100;	X X
else fehlermeldung ("KtoA	X X	KtoB = KtoB - 100;	X X
überzogen!");		end	
end-transaction;		else fehlermeldung ("KtoA	X X
		überzogen!");	
		end-transaction;	

Abbildung 80: Lock-Varianten

Lost Update:

- Transaktion A setzt zu Zeitpunkt 1 ein X Lock („for update“) ab, bekommt es und liest den Inhalt.
- Transaktion B beantragt zu ZP 2 ebenfalls ein X Lock, bekommt es aber nicht und wartet daher auf die kritische Ressource.
- Transaktion A verändert den gesperrten Preis zu ZP 3 und beendet die LUW und damit auch den Lock durch ein commit.
- Transaktion B bekommt erst jetzt den X Lock, beendet sein wait, und liest den korrekten Preis.
- ...

Uncommitted
Dependency:

- Transaktion A beantragt zu ZP 1 einen X Lock („update“), bekommt exklusive Kontrolle, und verändert den Preis.
- Transaktion B versucht zu ZP 2 einen S Lock zu bekommen, scheitert jedoch und wird in einen Wartezustand versetzt.
- Transaktion A beendet den Lock und die LUW mit einem Rollback.
- Transaktion B erhält nun den S Lock und liest den korrekten Preis.
- ...

Inconsistent Analysis:

- Transaktion A bekommt einen S Lock auf Kto 1 und Kto 2 (ZP 1).
- Der Versuch von Transaktion B die Ressourcen exklusiv zu sperren schlägt zu ZP 2 fehl (ist ja schon durch A gesperrt); B wartet ...
- Transaktion A wertet die Konti richtig aus und gibt sie anschließend frei (ZP 3 – 8)
- Erst jetzt darf B weitermachen, bekommt den X Lock und verändert die Kontostände.

Abbildung 81: Probleme gelöst?

ZP	Transaktion A		Preis	Transaktion B	
1	SELECT	S	100	-	
2	-	S	100	SELECT	S
3	UPDATE 10%	Anfrage X → warten	110	-	S
4	-	warten	120	UPDATE 20%	Anfrage X → warten
5	...	warten	warten

Abbildung 82: Deadlock

Wait-Die: Falls A älter ist als B, wartet A
sonst wird A zurückgenommen

Wound-Wait: Falls A älter ist als B, wird B zurückgenommen
sonst wartet A

Abbildung 83: Zeitmarken-Verfahren

Zeitpunkt	Transaktion A (älter)		Preis	Transaktion B (jünger)	
1	SELECT	S	100	-	
2	-	S	100	SELECT	S
3	UPDATE 10%	Anfrage X → warten	110	-	S
4	-	warten	120	UPDATE 20%	Anfrage X → ROLLBACK & Retry
5	...	X	...	Retry zu späterem Zeitpunkt	

Abbildung 84: Beispiel „Wait-Die“

Zeitpunkt	Transaktion A (älter)		Preis	Transaktion B (jünger)	
1	SELECT	S	100	-	
2	-	S	100	SELECT	S
3	UPDATE 10%	X	110	-	Rollback & Retry
4	...	X		Retry zu späterem Zeitpunkt	

Abbildung 85: Beispiel „Wound-Wait“

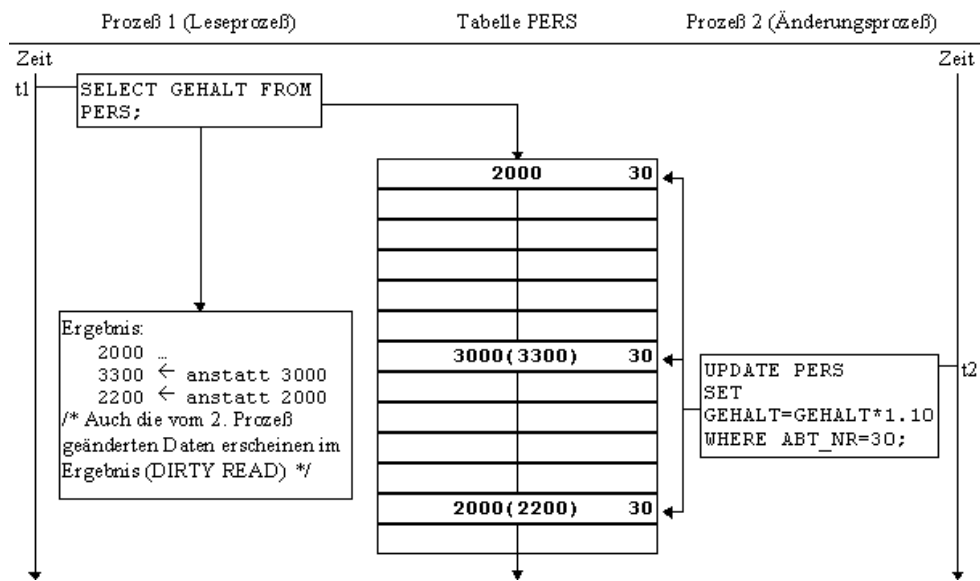


Abbildung 86: Inkonsistentes Lesen

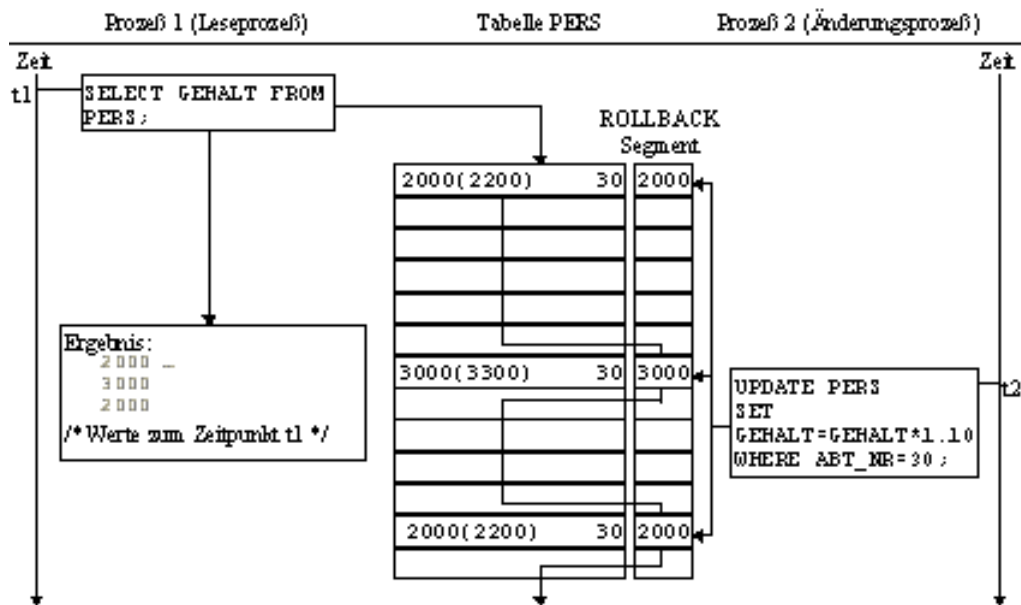


Abbildung 87: Konsistentes Lesen

Integrität (integrity)	Ein System ist dann in einem integren Zustand, wenn alle definierten Datenbank-Regeln (integrity rules) eingehalten werden; die Integrität wird vom DBMS selbstständig überprüft und sichergestellt. Datenintegrität umfasst alle Aspekte, welche das korrekte und zuverlässige Arbeiten mit Datenbanken sicherstellen und unterstützen (Datenkonsistenz, Datensicherheit, Datenschutz).
Konsistenz (consistency)	Ein System ist dann konsistent, wenn es keine widersprüchlichen oder inkorrekten Dateninhalte enthält, wie z.B. eine falsche Adresse eines Mitarbeiter; sie kann vom System normalerweise nicht überprüft werden. Man unterscheidet <u>logische Konsistenz</u> (die Inhalte der Datenbank entsprechen der modellierten Wirklichkeit), <u>semantische Integritätsbedingungen</u> (Wertebereiche, Constraints usw.) und <u>physische Konsistenz</u> (korrekte interne Repräsentation und Speicherung der Daten im Datenbanksystem).
Plausibilität	Bei der Plausibilitätsprüfung überprüft man, ob ein vom Benutzer eingegebener Wert plausibel ist, wie z.B. dass ein menschliches Lebensalter zwischen 0 und 110 Jahren sein muss.
Validierung (validation)	Bei der Validierung überprüft man, ob das Produkt die Erwartungen des Anwenders erfüllt; sie ist ohne Anwender nicht durchführbar. Die Fragestellung lautet dabei: Are we doing the right things?
Verifikation (verification)	Bei der Verifikation überprüft man, ob das Produkt bereits vorab niedergeschriebene Anforderungen erfüllt; dies ist auch ohne Anwender überprüfbar. Die Fragestellung lautet: Are we doing things right? Abbildung 88: Begriffsabgrenzung

- Kontobewegungen im einzelnen und als Summe
- Erfassung von Kindern bei jedem Elternteil
- Abspeichern der Umsatzsteuer bei Rechnungen

Abbildung 89: Beispiel aus der Praxis für Konsistenz

- Fehlermeldung und Abbruch
- Fehlermeldung und Zurückweisung
- Fehlermarkierung ohne Meldung
- Fehlermeldung mit Fehlerprotokollierung

Abbildung 90: Im Fehlerfall kann das Programm verschiedenartig reagieren

- Lieferantenummer soll die Form Snnnn haben mit nnnn vierstellige Ziffer
- Kundenstatus soll im Bereich 1 - 100 liegen
- Teilgewicht soll größer 0 sein
- Verkaufsmengen sollen ein Vielfaches von 100 sein

Abbildung 91: Im Beispielen für Integritätsregeln

- Datentypen
- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY: update and delete restrict, delete cascade
- CHECK
- Database Trigger

Abbildung 92: Realisierungsmöglichkeiten in ORACLE

- Einzelnes Attribut
 - Zivilstand: (ledig, verheiratet, geschieden, verwitwet)
 - Alter: [0..110]
- Einzelnes Tupel: Zusammenhänge zwischen verschiedenen Attributen
 - Zivilstand: (ledig, verheiratet, geschieden, verwitwet)
 - Alter: [0..110] falls Zivilstand = ledig; [16..110] sonst
 - Provision darf nicht höher als Gehalt sein
- Eine Relation: Bedingungen über gesamte Tabelle
 - Summe der Gehälter darf einen bestimmten Budget-Betrag nicht überschreiten
- Mehrere Relationen:
 - Ist das Gehalt von Mitarbeiter 1234 (Gehaltsstufe X) in der Höhe von Y zulässig?

Abbildung 93: Beispiel für user define

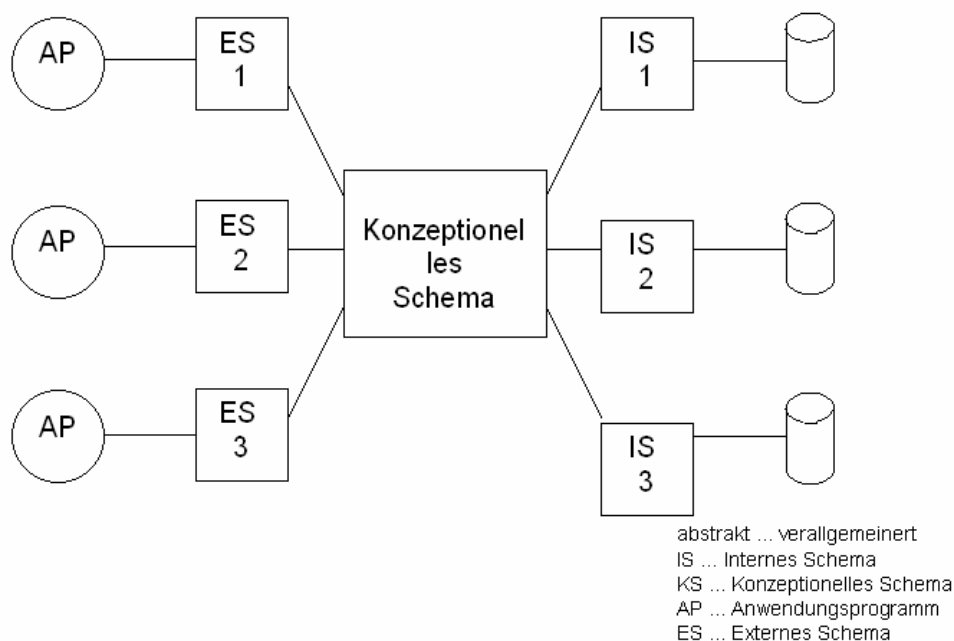


Abbildung 94: 3-Schema-Konzept

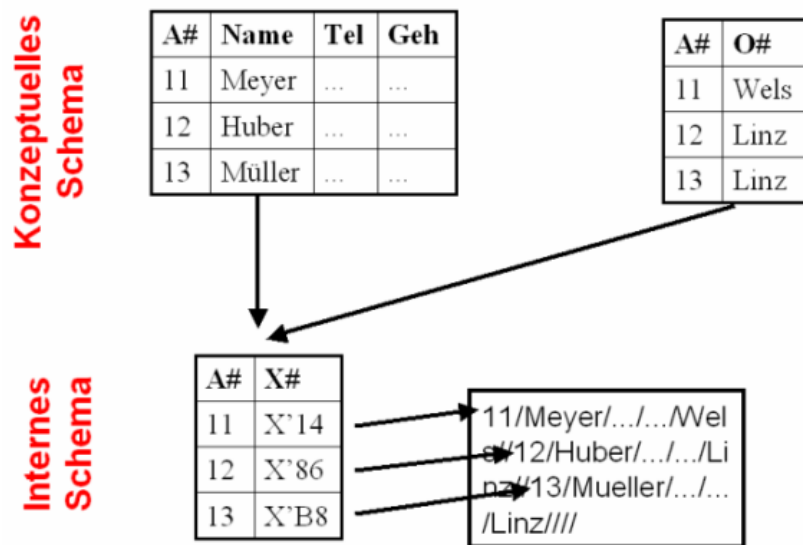


Abbildung 95: 3-Schema-Konzept

	Sequentielle Medium	Direkt- adressierbares Medium
Sequentielle Organisation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Indexorganisation		<input checked="" type="checkbox"/>
Randomorganisation		<input checked="" type="checkbox"/>
Hashorganisation		<input checked="" type="checkbox"/>

Abbildung 96: Zusammenhang Speichermedium – Datenorganisation

Sequentielles Medium: Datensätze werden unmittelbar hintereinander gespeichert und können auch nur in dieser Form verarbeitet werden.

Direkt-adressierbares Medium: Jeder beliebige Datensatz kann mit Kenntnis der Satzadresse sofort gelesen / erstellt / geändert / gelöscht werden.

Sequentielle Organisation: Zugriff auf Datensätze nur in der Reihenfolge, wie sie gespeichert wurden. Dabei zwei Möglichkeiten: unsortiert (Heap-Organisation) und sortiert

Indexorganisation: Zugriff über „Inhaltsverzeichnis“.

Random-Organisation: Wahlfreier Zugriff auf eine Satzadresse mit Hilfe einer Formel $adr = f(key)$

Hash-Organisation: W.o., jedoch können für zwei verschiedene Schlüsselwerte gleiche Satzadressen errechnet werden: dann muss eine sog. Kollisionsstrategie herangezogen werden.

Abbildung 97: Organisationsformen


```

procedure Seq_Suche (Low_bound, High_bound, Search_arg)

  A = Low_bound
  Record = read(A)

  do while (A < High_bound and Search_arg <> Record.Key)
    inc A
    Record = read(A)
  end while

  if Search_arg <> Record.Key
  then return(null)
  else return(Record)
end procedure

```

Abbildung 98: Sequentielle Suche

```

procedure Schritt_Suche(Lo, Hi, SKey)
  Record = read(Lo)
  if SKey < Record.Key
  then return(null)
  Record = read(Hi)
  if SKey > Record.Key
  then return(null)

  X = floor(sqrt(Hi - Lo + 1))

  A = Lo
  Record = read(A)

  do while (A + X < Hi and SKey < Record.Key)
    A = A + X
    Record = read(A)
  end while

  case of
    SKey = Record.Key: return(Record )
    SKey > Record.Key: return(Seq_Suche(A-X+1, A , SKey))
    SKey < Record.Key: return(Seq_Suche(A-X+1, Hi, SKey))
  end case
end procedure

```

Abbildung 99: Schrittweise Suche

```

procedure Bin_Suche(Lo, Hi, SKey)
  if Hi < Lo
  then return(null)

  A = floor((Hi + Lo) / 2)

  A = Lo
  Record = read(A)

  case of
    SKey = Record.Key: return(Record )
    SKey > Record.Key: return(Bin_Suche(A+1, Hi , SKey))
    SKey < Record.Key: return(Bin_Suche(Lo , A-1, SKey))
  end case
end procedure

```

Abbildung 100: Binäres Suchen

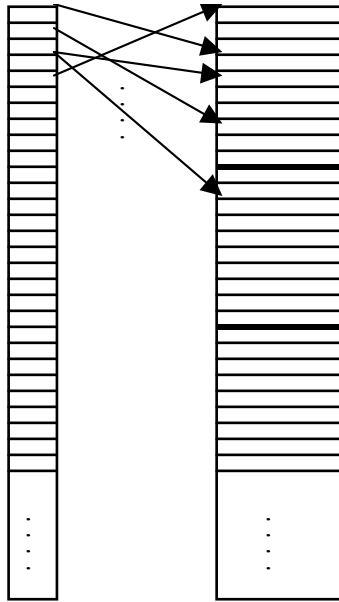


Abbildung 101: Vollständiger Index

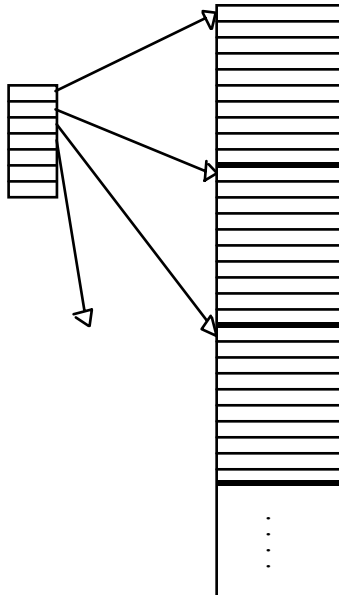


Abbildung 102: Blockindex

Indextabelle

Anne	1
Kurt	9
Willhelm	6

Datentabelle

1	Anne	
2	Berta	
3	Daniel	
4	Elke	
5		
6	Willhelm	
7	Xaver	
8		
9	Kurt	
10	Ludwig	
11	Martha	
12	Norbert	
13		

Abbildung 103: Blockindex mit Teilsortierung

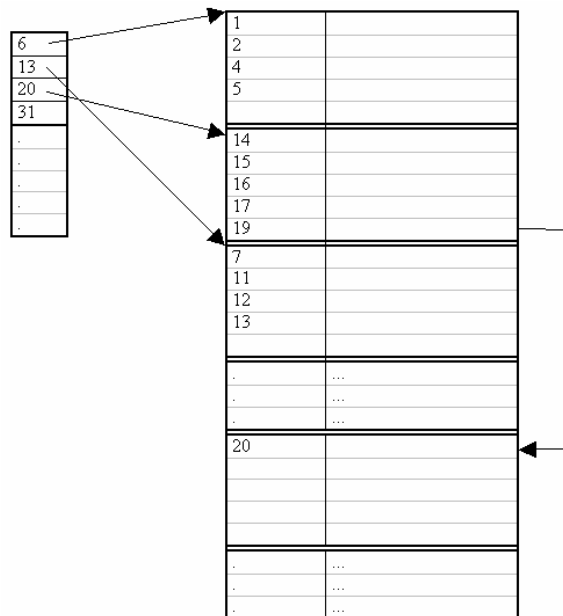


Abbildung 104: Überlaufblock

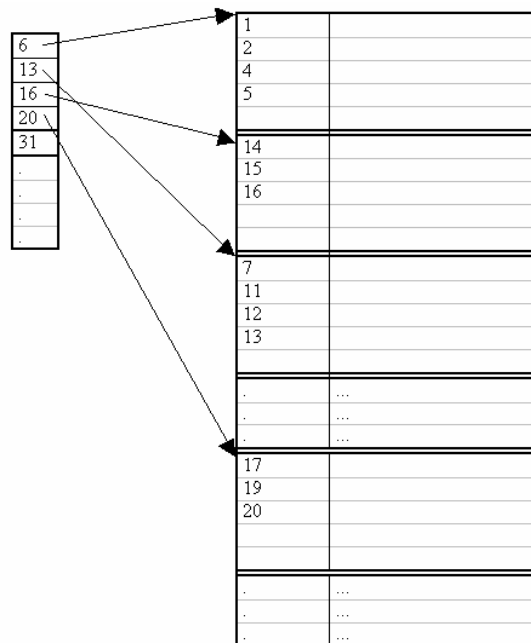


Abbildung 105: Block-Split

Schlüsselwert	Satzadresse
101	0
102	1
103	2
...	...
199	198
200	199

Abbildung 106: trivialer Index

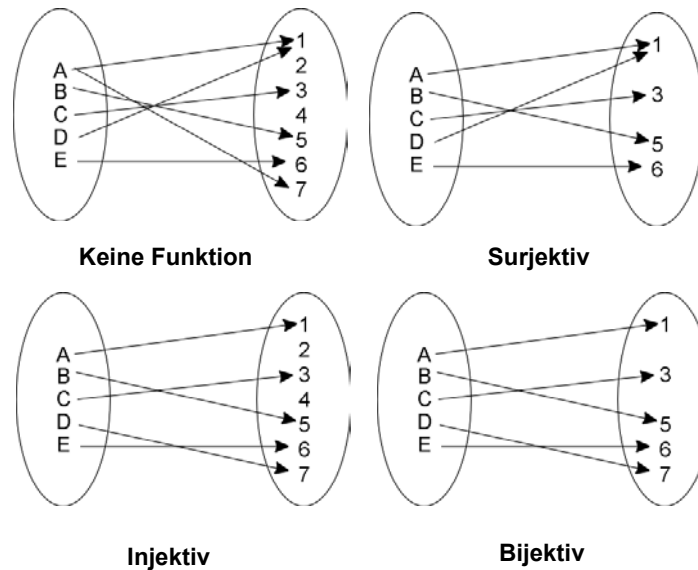


Abbildung 107: Funktionseigenschaften

Vorteile	Nachteile
Zugriff geht nicht besser, da wirklich nur ein einziger notwendig ist – egal ob Satz vorhanden oder Satz nicht vorhanden.	Je Wert des Schlüssels wird eine Speicherplatz reserviert: ist der Schlüssel sehr lückenhaft belegt, so wird enorm viel Platzverschwendet (siehe nachfolgendes Beispiel)
Speicherung des Schlüssels ist nicht notwendig, da man ihn jederzeit aus der Satzadresse errechnen kann.	

Abbildung 108: Vor- und Nachteile der Randomorganisation

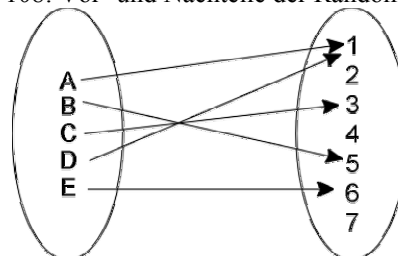


Abbildung 109: Nicht-injektive (Hash)Funktion

Mindestgröße der Datei:

$$q = \frac{\text{AnzahlBelegterPositionen}}{\text{AnzahlVerfügbarerPositionen}} \Leftrightarrow$$

$$\text{AnzahlVerfügbarerPositionen} = \frac{\text{AnzahlBelegterPositionen}}{q} =$$

$$= \frac{600}{0,85} = 705,88 = 706 \text{ Sätze}$$

Optimale Größe: nächste Primzahl = n = 709

Funktion: $h(x) = s \text{ MOD } 709$

Adresse: $\text{Adresse} = h(231011) = 231011 \text{ MOD } 709 = 586$

Abbildung 110: Beispiel Hashverfahren

Beispiel:

Hashalgorithmus: $h(s) = (s \text{ MOD } 7) + 1$ 7, weil Originalspeicherraum 1 – 7

Kollisionsstrategie: $k(s,i) = ((s + i) \text{ MOD } 15) + 1$, weil Überlaufbereich 1 – 15

Einzufügen sind Sätze mit folgendem Schlüsselwert:

s	=	16	14	19	28	24	7	40	39	12	2	4
h(s)	=	A03	A01	A06	A01	A04	A01	A06	A05	A06	A03	A05
k(s,1)	=				A15		A09	A12		A14	A04	A06
k(s,2)	=										A05	A07
k(s,3)	=										A06	A08
k(s,4)	=										A07	

Abbildung 111: Lineare Kollisionstrategie

Beispiel:

Hashalgorithmus: $h(s) = (s \text{ MOD } 7) + 1$ 7, weil Originalspeicherraum 1 – 7

Einzufügen sind Sätze mit folgendem Schlüsselwert:

s	=	16	14	19	28	24	7	40	39	12	2	4
h(s)	=	A03	A01	A06	A01	A04	A01	A06	A05	A06	A03	A05
k(s,1)	=				A09		A10	A11		A12	A13	A14

Abbildung 112: Verkettete Kollisionstrategie

Beispiel:

Hashalgorithmus: $h(s) = \text{mod}(s,7) + 1$ 7, weil Originalspeicherraum 1 – 7

Einzufügen sind Sätze mit folgendem Schlüsselwert:

s	=	16	14	19	28	24	7	40	39	12	2	4
Schritt		A0	A0	A0	*	A0	*	*	A0	*	*	*
1)		3	1	6		4			5			
Schritt					A0		A1	A1		A1	A1	A1
2)					9		0	1		2	3	4

Abbildung 113: Two Pass Load

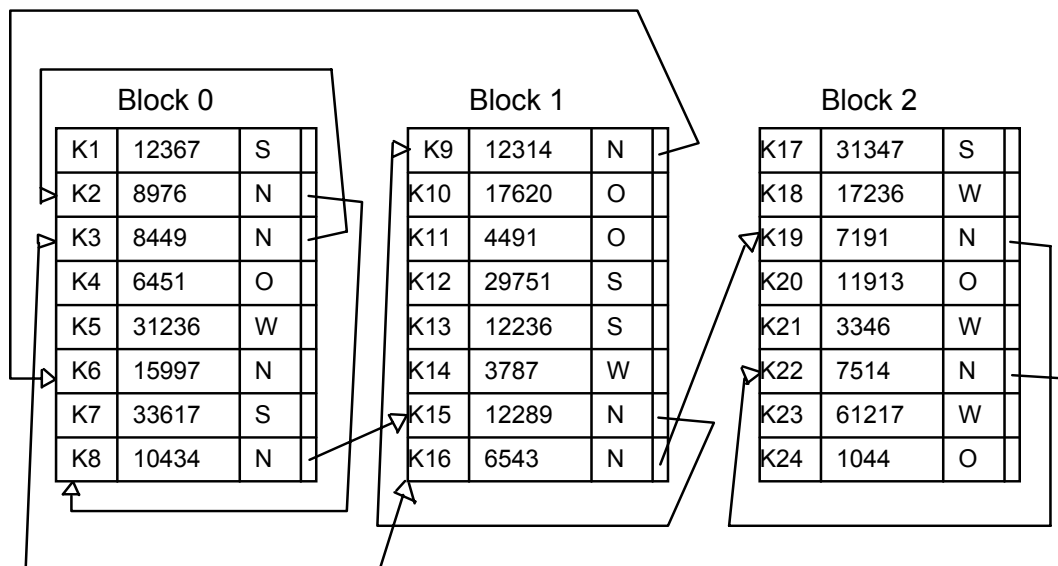


Abbildung 114: Sortierte Liste

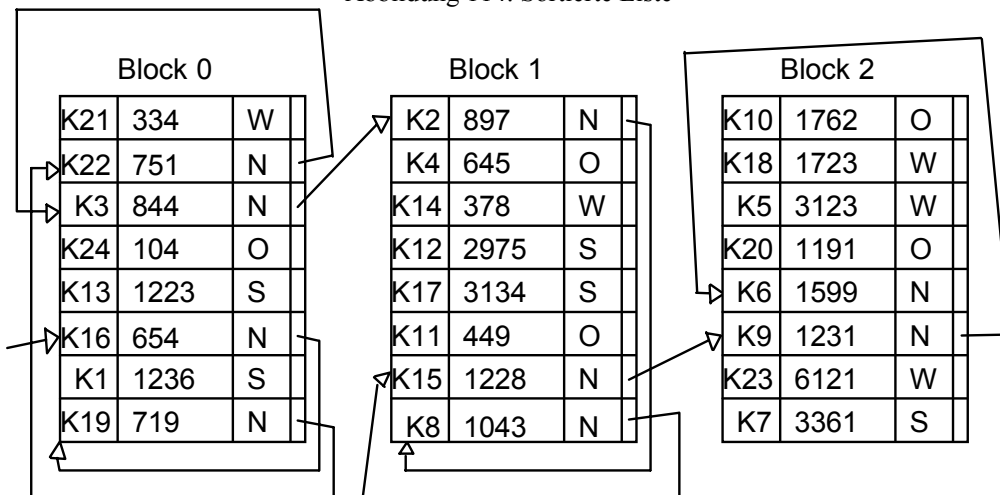


Abbildung 115: Sortierte Liste 2

Listenkopf

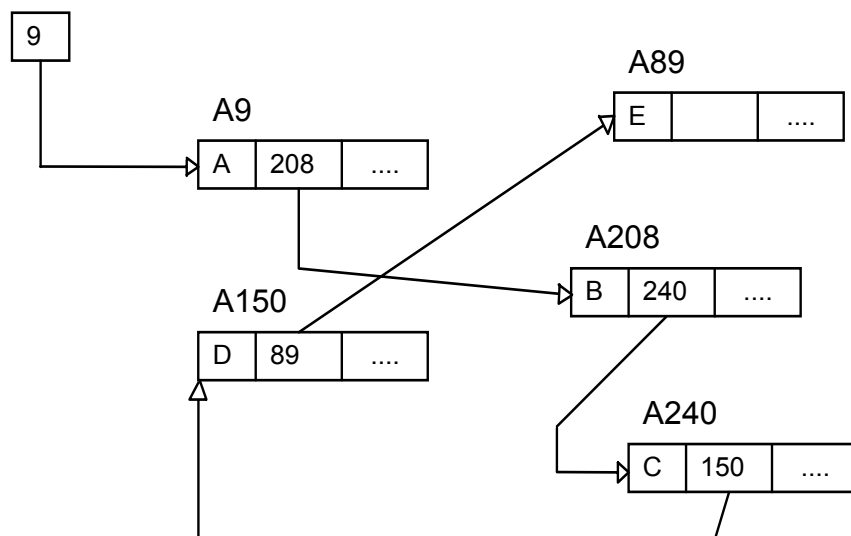


Abbildung 116: Physische Zeiger

Index

ID-Key	Adresse
A	9
B	208
C	240
D	150
E	89

Listenkopf

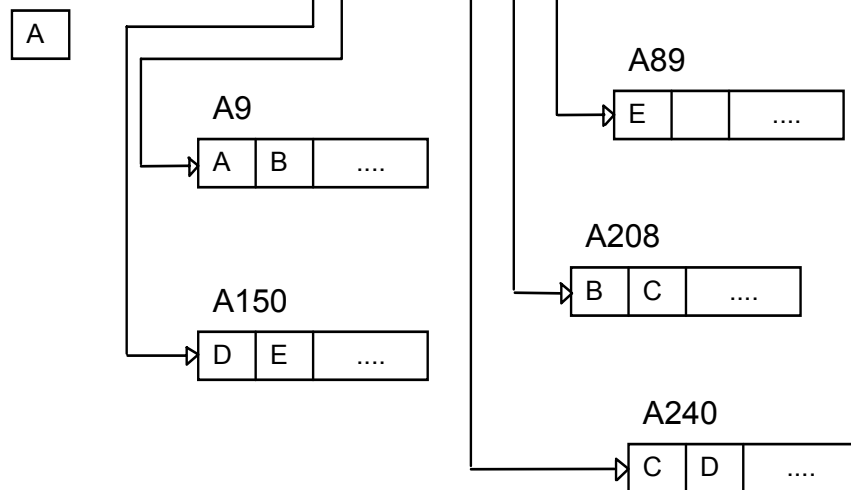


Abbildung 117: Logische Zeiger

Listenkopf

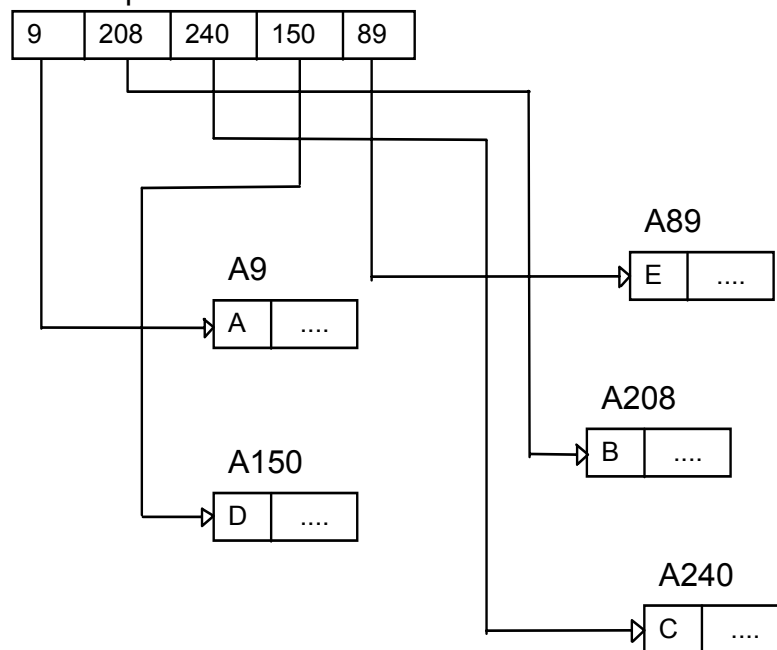
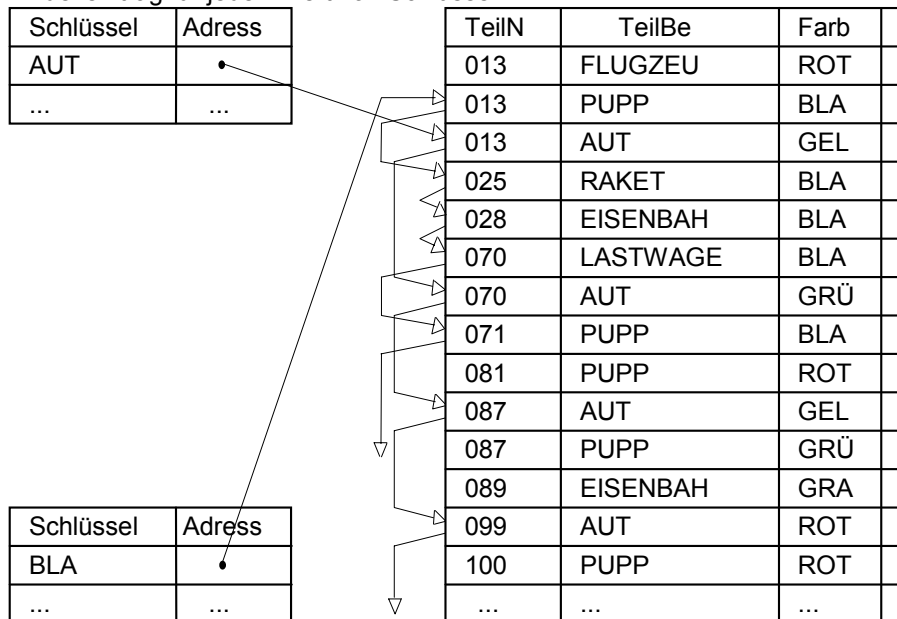


Abbildung 118: Invertierte Liste

Ort	Adr		P#	Name	Ort	Stufe
Linz	23	23	1	Franz	Linz	2
	53	38	2	Sepp	Wels	3
	83	53	3	Heidi	Linz	4
	113	68	4	Margo t	Marchtren k	3
Marchtren k	68	83	5	Theo	Linz	3
Wels	38	98	6	Kurt	Wels	2
	98	113	7	Sylvia	Linz	3

Abbildung 119: Übungsbeispiel invertierte Liste

1 Indexeintrag für jeden Wert von Schlüssel2



1 Indexeintrag für jeden Wert von Schlüssel3

Abbildung 120: Mehrere Sekundärschl.

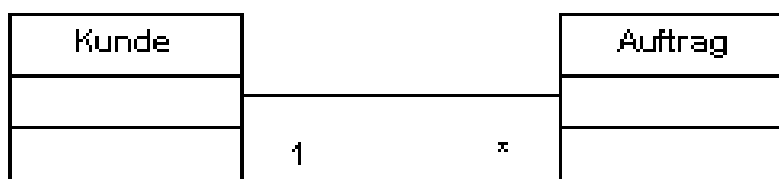


Abbildung 121: 1:n Teil 1

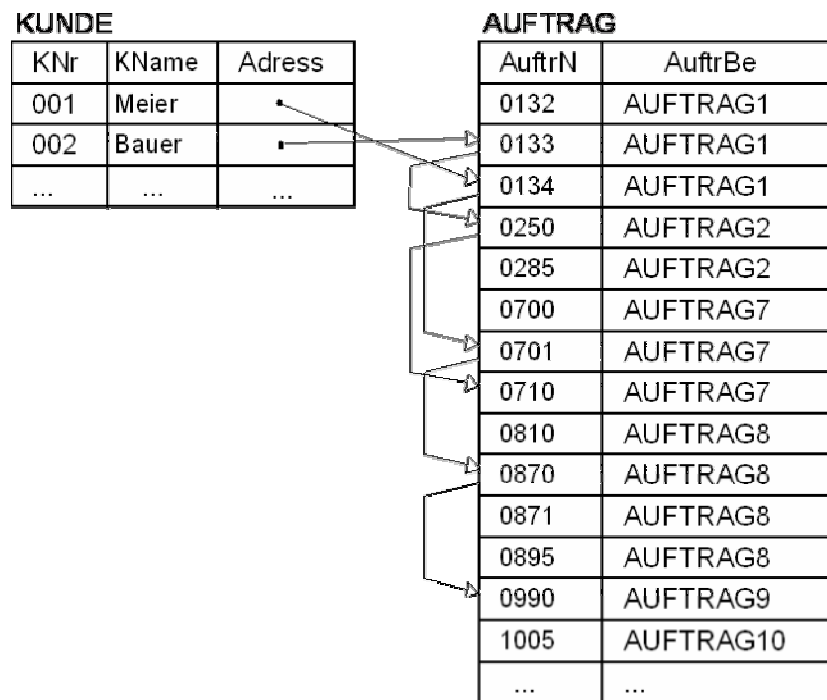


Abbildung 122: 1:n Teil 2

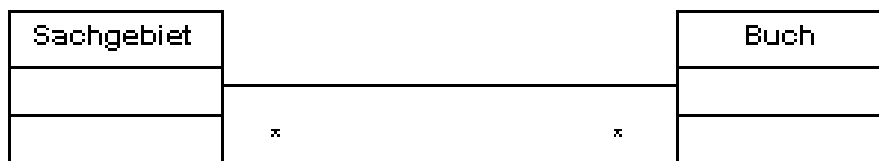


Abbildung 123: n:m Teil 1

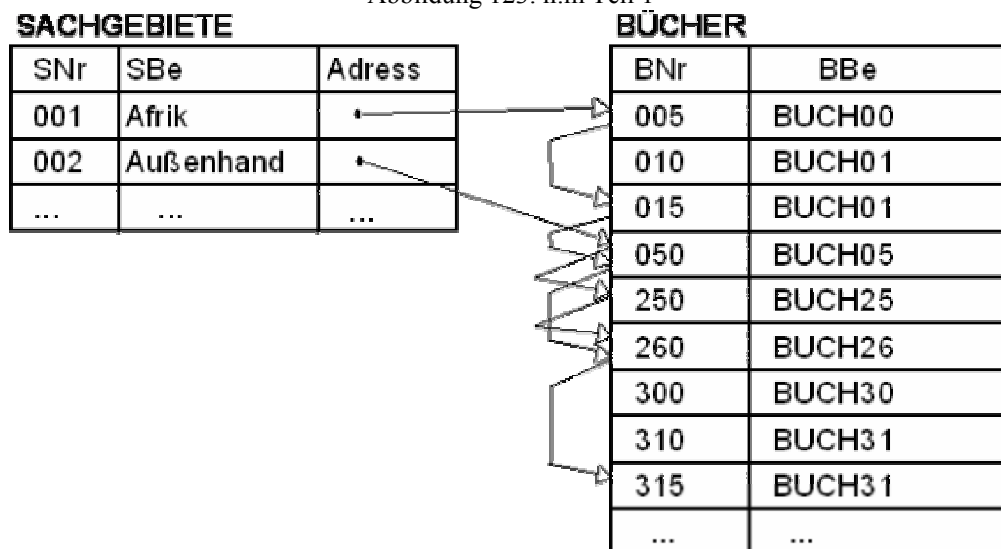


Abbildung 124: n:m Teil 2

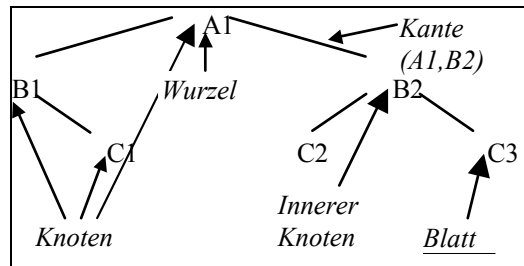


Abbildung 125: Elemente eines Baumes

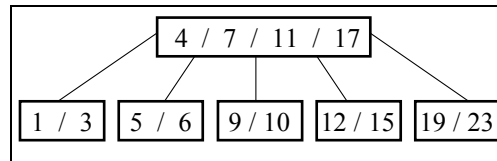


Abbildung 126: Suchbaum

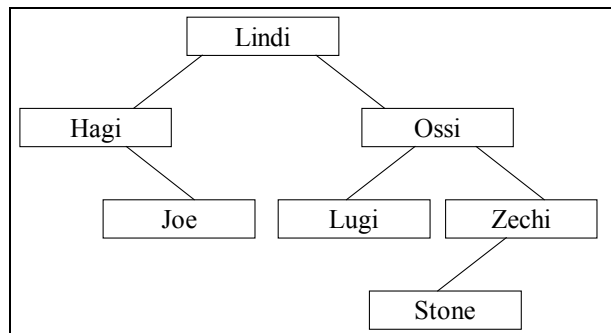


Abbildung 127: Binärer Suchbaum

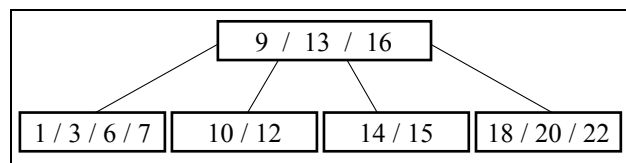


Abbildung 128: B(2,1)-Baum

```

Suche(KnotenAkt,X) returns KnotenX
  if X in KnotenAkt.Wert()
  then return(KnotenAkt)
  else begin
    if all KnotenAkt.Sohn() = nil
    then return(nil)

    case
      X < all KnotenAkt.Wert():
        return(Suche(KnotenAkt.Sohn(1),X))

      X > all KnotenAkt.Wert:
        return(Suche(KnotenAkt.Sohn(n+1),X))

    otherwise
      I := FindIntervall(KnotenAkt.Wert())
      return(Suche(KnotenAkt.Sohn(I),X))
    end other
  end case
end else
end Suche

```

Abbildung 129: Suche im B-Baum

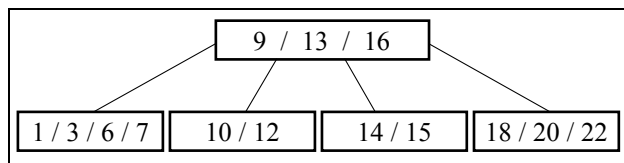


Abbildung 130: Beispiel – Einfügen in B-Baum

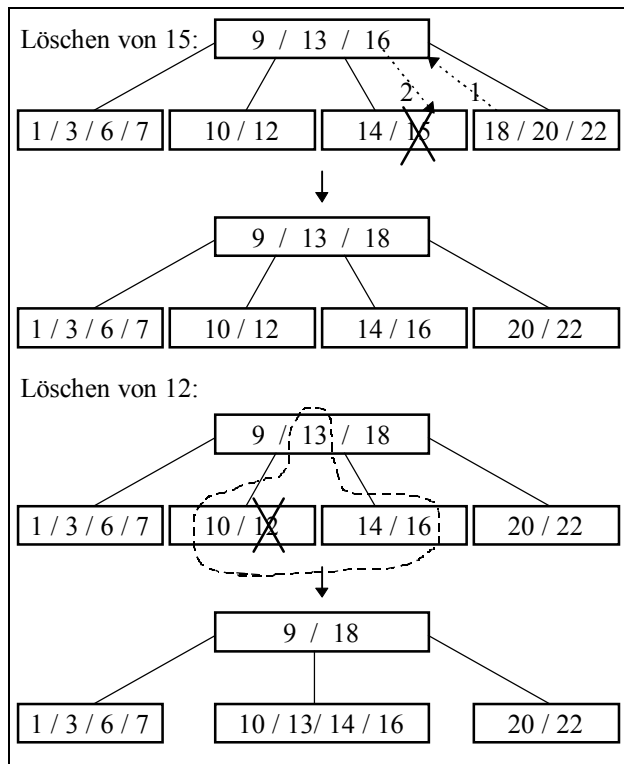


Abbildung 131: Beispiel – Löschen aus B-Baum