Datenstrukturierung in der Fertigung

Peter Loos

erschienen in Buchform:

R. Oldenbourg Verlag, München; Wien 1992, ISBN 3-486-22286-4

aus dem Vorwort:

Integrierte Informationssysteme, wie sie zunehmend als Bestandteil des Computer Integrated Manufacturing für den Fertigungsbereich eingesetzt werden, bedürfen neben einer durchgängigen Betrachtung der funktionalen Abläufe auch einer ganzheitlichen Darstellung der zugrundeliegenden Strukturen, die im Datenmodell des Anwendungssystems ihren Ausdruck findet. Diese Datenstrukturen müssen in einer der Problemstellung adäquaten Form dargestellt werden können, und bei der Umsetzung in ein Informationssystem mit Hilfe der zur Verfügung stehenden Datenbankmanagementsysteme realisert werden können.

Das Buch bietet einen umfassenden Datenmodellierungsansatz, der die Problemstellungen der industriellen Fertigung berücksichtigt. Dazu wird als Beschreibungssprache das Expanded Entity-Relationship-Modell (PERM) entwickelt, eine semantische Erweiterung des Entity-Relationship-Modells. Mit Hilfe der Modellierungsmethode wird ein Referenzdaten- modell für den Fertigungsbereich erstellt, in dem eingehend die Strukturen von Fertigungsgrunddaten, Arbeitsplandaten, Chargendaten, Lager- und Transportdaten sowie Auftrags- und Instandhaltungsdaten erläutert werden. Ein weiterer Schwerpunkt liegt auf der Repräsentation der Datenstrukturen in relationalen Datenbanksystemen, die mit Hilfe von SQL-Befehlen in Datenbankschemata Überführt werden.

Inhaltsverzeichnis

1. Einführung

1.1 Motivation

Neue Anforderungen wie wachsender Konkurrenzdruck und zunehmend kundenorientierte Produktion zwingen Industrieunternehmen zur Anpassung ihrer Produktspektren und ihrer Organisation, die in der integrierten Informationsverarbeitung des Computer Integrated Manufacturing (CIM) ihren Ausdruck findet. Diese Anpassungen stellen nicht nur einen wesentlichen Wettbewerbsvorteil dar, sondern werden in zunehmendem Maße zum kritischen Erfolgsfaktor. Ein wichtiges Element von CIM ist der Einsatz von Fertigungs-informationssystemen, deren Gestaltung und Implementierung ein adäquates Beschreibungsmodell der abzubildenden Aufbau- und Ablauforganisation voraussetzt. Diese Organisationsstrukturen lassen sich durch ein sogenanntes Datenmodell, einer Komponente des Informationsmodells, darstellen.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit besteht darin, einen umfassenden Datenmodellierungsansatz zu entwickeln, der die Problemstellungen der industriellen Fertigung berücksichtigt sowie darauf aufbauend ein Referenzdatenmodell des Fertigungsbereichs zu entwerfen.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in acht Kapitel. Nach der Einführung wird im **zweiten Kapitel** die Bedeutung der Datenmodellierung für die Gestaltung von Informationssystemen erläutert. Dafür wird das Informationsmodell als Gesamtbeschreibung für betriebliche Anwendungen in die Modellkomponenten Daten, Funktionen und Ablaufsteuerung unterteilt, und diese werden gegeneinander abgegrenzt. Diese sachlogische Unterteilung wird in ein Ebenenschema eingebettet, das die Strukturen eines Informationssystems auf verschiedenen Abstraktionsschichten wiedergibt.

Im **dritten Kapitel** werden die von einem Fertigungsinformationssystem abzudeckenden Funktionalbereiche definiert und die Anforderungen an diese Systeme aufgezeigt, die in den technologischen Weiterentwicklungen und geänderten Marktsituationen begründet sind. Daneben werden die sich aufgrund unterschiedlicher Fertigungsmerkmale ergebenden Einflußfaktoren auf die Produktion mit den Konzequenzen für die Gestaltung von Fertigungsinformationssystemen diskutiert. Daraus lassen sich allgemeine Anforderungen an das Datenmanagement für den Fertigungsbereich ableiten, die sich nach dem in Kapitel zwei definierten Ebenenschema differenzieren lassen.

Die wichtigsten bisher für die Formulierung von Datenmodellen entwickelten Methodiken werden in **Kapitel vier** vorgestellt, wobei nur graphische Beschreibungssprachen berücksichtigt werden, da die mathematisch-formalen Ansätze für die in der Arbeit verfolgte Zielsetzung als Kommunikationsinstrument zwischen den Anwendern und Entwicklern nicht geeignet sind. Die graphischen Beschreibungssprachen gehen vorwiegend auf das erstmals 1976 vorgestellte Entity-Relationship-Modell zurück. Die diskutierten Methodiken sind neben dem Ursprungsmodell die Erweiterten Entity-Relationship-Modelle, das Strukturierte Entity-Relationship-Modell, das Entity-Category-Relationship-Modell, die Nijssens Information Analysis Methode, die Entitäten Diagramme sowie das Sematic-Association-Modell.

In **Kapitel fünf** werden die Anforderungen an die Datenmodellierung aus Sicht der Fertigung definiert und den Merkmalen der vorgestellten Methodiken in Form einer Bewertung gegenübergestellt. Die Bewertungskriterien sind neben den meist verfügbaren Klassifizierungs-, Aggregations- und Gruppierungsoperatoren, die Möglichkeiten der Darstellung komplexer Integritätsbedingungen, die Generalisierung, die Versionsunterstützung sowie die Darstellung komplexer Objekte bzw. Cluster.

Aufgrund der bei der Bewertung aufgezeigten semantischen Darstellungsdefizite wird in **Kapitel sechs** eine der Problemstellung adäquate Beschreibungssprache, das Expanded Entity-Relationship-Modell, entwickelt, dessen Konstrukte und Operatoren sich möglichst an den bisher vorgestellten Methodiken orientieren. Nach der Definition der Beschreibungssprache wird diese vollständig mit ihren eigenen Sprachmitteln beschrieben, im sogenannten Metamodell.

Mit dieser Datenmodellierungsmethodik wird in **Kapitel sieben** ein Referenzdatenmodell für die in Kapitel drei abgegrenzten Funktionalbereiche eines Fertigungsinformationssystems entwickelt. Nach einer allgemeinen Darstellung der für den Fertigungsbereich notwendigen

Abbildungsmöglichkeit der zeitlichen Dimension werden die Strukturen der Grund-, Arbeitsplan-, Chargen-, Lager- und Transport-, Auftrags- und Instandhaltungsdaten entwickelt.

Kapitel acht zeigt die Möglichkeit der Abbildung der definierten Datenstrukturen in realen Datenbanksystemen. Dabei wird als Modell der Datenbanksysteme von dem Relationenmodell sowie von dem darauf aufbauenden SQL-Ansatz ausgegangen. Desweiteren werden dessen bereits teilweise vorhandenen Erweiterungen berücksichtigt. Nach einer allgemeinen Darstellung der Überführungsmöglichkeiten von Expanded Entity-Relationship-Strukturen werden zur Verdeutlichung ausgewählte Komponenten des Referenzmodells abgebildet.

Eine Zusammenfassung der Ergebnisse sowie ein thematischer Ausblick in **Kapitel neun** schließen die Arbeit.

2. Datenmodellierung als Gestaltungselement von Informationssystemen

Informationssysteme in der betrieblichen Anwendung dienen der Unterstützung von Tätigkeiten und Abläufen, die sich durch Ereignisse, Vorgänge und Zustände beschreiben lassen (Scheer 90d, S. 137). Der Anwender nutzt die im Informationssystem abgebildeten Funktionen und Daten, die durch eine Ablaufsteuerung miteinander in Beziehung stehen. Die Komponenten eines Informationssystems sind in Abbildung 1 dargestellt.

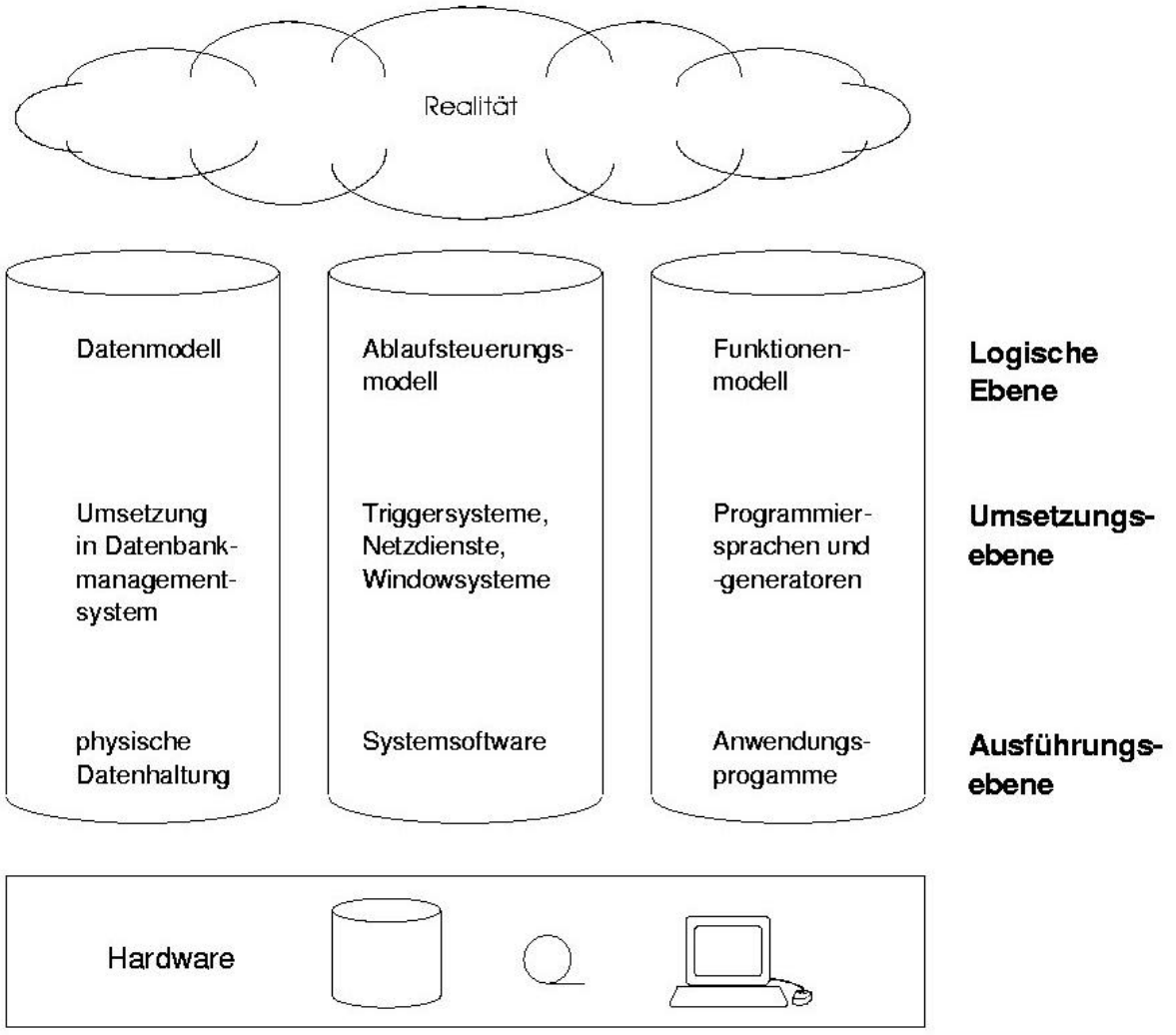


Abb.1: Architektur eines Informationssystems (in Anlehnung an: Scheer 90d, S. 144)

2.1 Die Architektur eines Informationssystems

Bei der Darstellung der einzelnen Komponenten können verschiedene Abstraktionsebenen unterschieden werden, die **Logische Ebene**, die **Umsetzungsebene** und die **Ausführungsebene**.

Die Aufgabe der Ausführungsebene ist die Kontrolle und Steuerung der **Anwendung**. Aus Sicht der Daten gehören dazu beispielsweise die physische Organisation der Plattenlaufwerke, die Zugriffsalgorithmen für das Datenretrieval oder die Synchronisationsmechanismen bei der Aktualisierung der Datenbestände. Die funktionale Ausprägung wird durch den ausführbaren Objektcode der Anwendungs- bzw. Applikationssoftware repräsentiert. Die Ablaufsteuerung eines in Betrieb befindlichen Informationssystems koordiniert die Zugriffe der Anwendungsprogramme auf die Daten sowie die Kommunikation mit dem Benutzer unter Nutzung von Systemsoftware wie Einheiten-Treiber, Terminalsteuerungen oder Netzen.

Die Umsetzungsebene repräsentiert die Implementierungsschicht, in der mit Hilfe verschiedener Tools die Anforderungen aus Sicht der betrieblichen Anwendung in einem DV-System realisiert werden (**Realisierung**). Scheer bezeichnet diese Ebene auch als Tool-Ebene (Scheer 90d, S. 146). Zur Umsetzung der Datensicht werden Datenbanksysteme eingesetzt, die die Implementierung mit Werkzeugen wie Data Dictionary, Reportgeneratoren oder benutzerfreundlichen Abfragesprachen unterstützen. Die Funktionen werden mit Hilfe von klassischen Programmiersprachen, Sprachen der vierten Generation (4GL), Programmgeneratoren oder bei Anwendungen aus dem Gebiet der Künstlichen Intelligenz mit Expertensystem-Shells realisiert. Die Implementierung der Ablaufsteuerung wird durch Triggersysteme, Electronic-Mail und Netzdienste sowie an der Schnittstelle zu dem Benutzer durch Maskengeneratoren und graphische Window-Systeme unterstützt.

Die logische Ebene eines Informationssystems wird durch ein Modell repräsentiert, dem die betriebliche Aufbau- und Ablauforganisation zugrunde liegen. Das Ziel dieses **Informationsmodells** liegt in der Darstellung des für den Anwender relevanten Ausschnitts der betrieblichen Realität. Entsprechend der Unterteilung der Ausführungs- und Umsetzungsebenen kann das Informationsmodell aufgrund des **Beschreibungsgegenstandes** in Einzelmodelle gegliedert werden. Diese Modelle der Informationsstrukturen können unabhängig von einer DV-gestützten Implementierung eines Informationssystems erfolgen. Erst durch die Umsetzungsebene wird das Informationsmodell in einem DV-System abgebildet.

2.2 Die Datenmodellierung innerhalb der Informationsmodellierung

Scheer definiert Informationsmodellierung als "Vorgang der Umsetzung der betriebswirtschaftlichen Konzeption in eine benutzernahe aber doch formalisierte Sprache" (Scheer 90c, S. 2).

Die Komponenten des Informationsmodells, das Datenmodell, das Funktionenmodell sowie das Ablaufsteuerungsmodell, stehen in einer engen, voneinander abhängigen Beziehung. Das **Datenmodell** behandelt die für die betriebliche Aufgabenstellung notwendigen Daten als eigenes Organisationselement und beschreibt die Strukturen von Objekten, deren Beziehungen zueinander und aller zugehörigen Integritätsbedingungen. Diese Informationen können sowohl ablauforganisatorische Daten wie Bestell- und Fertigungsaufträge als auch aufbauorganisatorische Daten wie Arbeitsplatzhierarchien und Kostenstellenzugehörigkeiten sein. Das **Funktionenmodell** beschreibt die Tätigkeiten des betrieblichen Ablaufs, wobei Tätigkeiten als Zusammenfassung von elementaren Funktionen verstanden werden. Funktionen sind dabei als Transformationen von Input- in Outputdaten definiert (Aue et al. 90, S. 3). Das **Ablaufsteuerungsmodell** legt die Beziehungen zwischen den Funktionen und Daten fest und regelt die Prozesse der betrieblichen Tätigkeiten. So muß bsw. definiert werden, welche Funktionen bei Eintreten eines bestimmten Ereignisses wie Auftragseingang, Rückmeldung eines Fertigungsauftrags oder Erreichung eines Zeitpunkts auszuführen sind. Für die Beschreibung der Modelle werden unterschiedliche Methoden oder Beschreibungssprachen, wie Entity-Relationship-Modell, Datenflußdiagramme, SADT, HIPO, Programmablaufpläne, Struktogramme oder Vorgangskettendiagramme eingesetzt (ein Überblick über die Methoden ist zu finden in: Scheer 90f, S. 14 - 73).

Die Datenstrukturen als Beschreibungsgegenstand des Datenmodells gehören zu den langlebigsten und stabilsten Bausteinen eines Informationsmodells (Scheer 90e, S. 92). Ziel der Beschreibung ist deshalb das Erreichen eines hohen Grades an Datenunabhängigkeit. Dadurch sollen die Strukturen zum einen implementierungsneutral (physische Datenunabhängigkeit), zum anderen aber auch unabhängig von den einzelnen Funktionen und Prozessen beschrieben werden (logische Datenunabhängigkeit). Schon frühzeitig wurde deshalb für die Implementierung von Datenbanksystemen (DBS) ein Ebenenschema vorgeschlagen, dessen Schichten sich im Datenteil des Architekturmodells eines Informationssystems widerspiegeln. Ein Datenbanksystem ist dabei als der Teil eines Informationssystems definiert, "der sich mit der Beschreibung der vorhandenen Daten, ihrer Verwaltung sowie dem Umgang mit und dem Zugriff zu ihnen befaßt" (Schlageter/Stucky 83, S. 13).

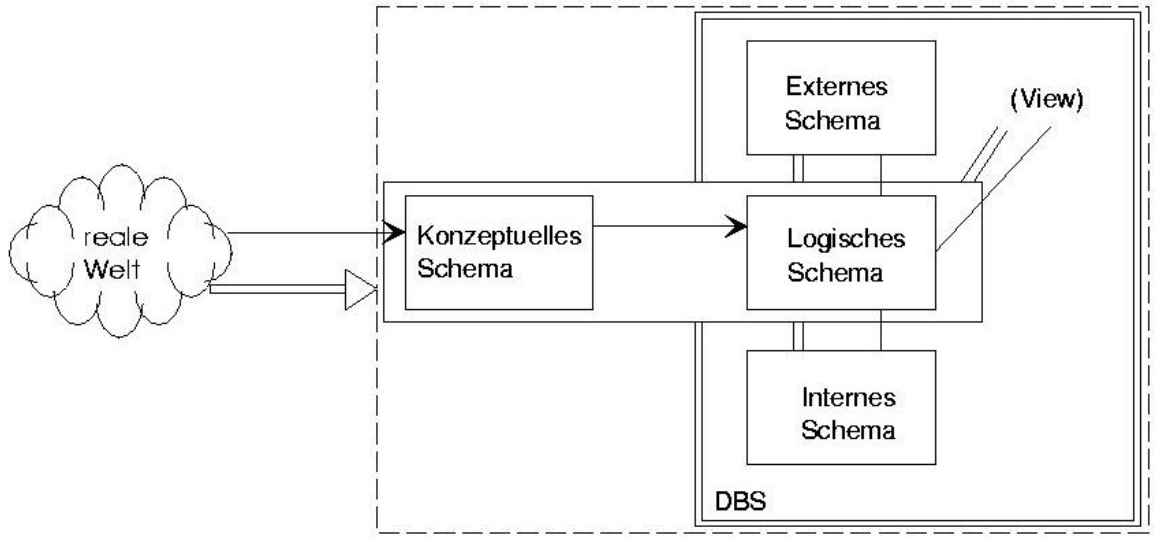


Abb. 2: Ebenenschema der Datenbanksysteme (aus: Schlageter/Stucky 83, S. 43)

In dem in Abbildung 2 gezeigten Ebenenschema entspricht das Interne Schema der datenbezogenen Ausführungsebene der Informationssystemarchitektur (s. S. 4). Das Logische Schema beinhaltet die Modelle der Datenbanksysteme (z. B. Netzwerkmodell oder Relationenmodell, vgl. hierzu Scheer 90f, S. 44 - 49) und repräsentiert damit die Umsetzungsebene der Informationssystemarchitektur. Das Konzeptuelle Schema als Gesamtheit der abzubildenden Datenstrukturen entspricht der Logischen Ebene der Informationssystemarchitektur. Das Externe Schema hat die Aufgabe, die für eine spezielle Anwendung notwendige Datensicht als Ausschnitt der Gesamtsicht bereitzustellen. Da die Repräsentation des Datenausschnitts aber der Aufgabenstellung angepaßt ist und somit auch einer Programmschnittstelle entsprechen kann, besitzt das Externe Schema sowohl Eigenschaften der Logischen Ebene als auch der Umsetzungsebene.

Die im Datenmodell definierten Strukturen müssen bei einer Implementierung, d. h. dem Übergang zur Umsetzungsebene, in dem Modell des für die Realisierung bestimmten Datenbanksystems (z. B. Relationenmodell) abgebildet und in die formale Data Description Language (z. B. SQL) überführt werden.

Es ist zu beachten, daß das dem Datenbanksystem zugrundeliegende Modell in der Literatur ebenfalls Datenmodell genannt wird (Luft 90). Dieses Modell als Bestandteil der Umsetzungsebene soll aber von der hier benutzten Definition des Datenmodells, als Komponente des Informationsmodells der Logischen Ebene, unterschieden werden.

2.3 Nutzen der Datenmodellierung

Die Beschreibung der Daten in einem eigenen Modell ist ein wichtiges Gestaltungselement für die Planung und Realisierung von Informationssystemen. Die wichtigsten Vorteile eines Datenmodells lassen sich wie folgt charakterisieren:

- Die Erarbeitung der Datenstrukturen in Rahmen der Datenmodellierung zeigt die **informationelle Verflechtung betrieblicher Aufgaben- und Abteilungsbereiche**, deren Etablierung oftmals mehrere Jahrzehnte zurückliegt. Die seit dem letzten Jahrhundert praktizierte Arbeitsteilung im Rahmen des Taylorismus führte zu einer Zerlegung von Prozeßabläufen mit der Konsequenz, daß zwar bei den einzelnen Teilaufgaben ein Produktivitätsgewinn zu verzeichnen ist, der Gesamtdurchlauf einer Vorgangskette allerdings bei jedem Abteilungswechsel durch eine große Warte- und Übergangszeit verzögert wird und für jede Teilaufgabe neue Einarbeitungszeiten anfallen können. Die moderne Informationstechnologie bietet dagegen die Möglichkeit, Teilaufgaben zu einem Gesamtablauf zusammenzufassen und damit die Durchlaufzeiten von Vorgangsketten wesentlich zu verkürzen. Diese Sichtweise ist eine wesentliche Voraussetzung für eine integrierte Informationsverarbeitung, die neben den Entwicklungen der Fertigungstechnologien der wichtigste Ansatzpunkt des Computer Integrated Manufacturing ist (vgl. hierzu Scheer 90 a).

- Mit der Darstellung der Datenstrukturen kann das Datenmodell funktionsunabhängig definiert werden. Dadurch können die sonst den Aufgaben und Teilbereichen zugeordneten Daten prozeßübergreifend in integrierter Form betrachtet werden (logische Datenunabhängigkeit). Mit Hilfe der Gesamtsicht können **Datenredundanzen** vermieden oder zumindest offengelegt werden, die bei isolierter Betrachtung zu inkonsistenten Daten und schließlich zu falschen Entscheidungsgrundlagen führen.

- Die rasche Entwicklung im Hardware- und systemnahen Software-Bereich, sowie die zunehmende Komplexität integrierter Anwendungssysteme führen dazu, daß Informationssysteme nicht genau auf eine DV-Anlage zugeschnitten sein können, da diese sonst nur durch Neuimplementierung an neue Rechnergenerationen angepaßt werden könnten. Die Abstraktion der Datenstrukturen von der physischen Realisierung der Ausführungsebene einerseits, sowie von den Datenbanksystem-Modellen der Umsetzungsebene anderseits, werden als physische Datenunabhängigkeit bezeichnet und gewährleisten eine weitgehende **Flexibilität** bei der Anpassung von Informationssystemen.

- Durch eine DV-unabhängige Form von Datenmodellen können diese als **Kommunikationsmedium** für die Verständigung zwischen den Anwendern von Informationssystemen wie Fachabteilung und Management einerseits und den DV-Spezialisten wie Systemanalytikern, Programmierern und Datenbankadministratoren anderseits eingesetzt werden.

- Die Modelle der Datenstrukturen dienen der Beschreibung der Aufbau- und Ablauforganisation eines Betriebes und bieten damit gleichzeitig eine **Dokumentation**. Man kann sie allgemein als Beschreibungsinstrument für informationelle Strukturen unabhängig von der Frage des DV-Einsatzes nutzen.

- Mit Hilfe eines Datenmodells können **Referenzmodelle** bereitgestellt werden, die als Ausgangsbasis für die Entwicklung von auf spezielle Problemstellungen zugeschnittenen Informationssystemen oder als Vergleichsmaßstab für die Beurteilung bestehender Software-Produkte genutzt werden können.

3. Charakterisierung von Informationssystemen für die Fertigung

Die Fertigung als ausführender Bereich innerhalb eines Industriebetriebs hat die Aufgabe, die durch Verkauf, Primärbedarfsplanung und Materialwirtschaft (im Rahmen der Produktionsplannung und -steuerung, PPS) vorgegebenen und durch die Konstruktion (Computer Aided Design, CAD) spezifizierten Bedarfe an Eigenfertigteilen zu realisieren. Im folgenden sollen Informationssysteme für die Fertigung anhand der abzudeckenden Funktionsbereiche unter Beachtung neuer Anforderungen und unter Berücksichtigung unterschiedlicher Einflußfaktoren charakterisiert werden.

3.1 Neue Anforderungen an die Gestaltung von Fertigungsinformations-systemen

Aufgrund geänderter Markterfordernisse einerseits, die sich im Wandel von einem Verkäufer- in einen Käufermarkt mit zunehmendem Konkurrenzdruck für die Anbieter widerspiegeln, sowie technologischer Entwicklungen andererseits, die durch starke Innovationsschübe in den Fertigungstechniken gekennzeichnet sind, werden Unternehmen gezwungen, ihre Produktion an die Anforderungen anzupassen, wobei sich die genannten Entwicklungen gegenseitig beeinflussen. Diese Anpassungen haben direkten Einfluß auf die Gestaltung von Informationssystemen für die Fertigung:

- Neue automatisierte **Fertigungstechnologien** wie Bearbeitungszentren (BAZ), Flexible Fertigungszellen (FFZ) und -systeme (FFS), DNC-Betrieb (DNC = Direct Numerical Control) oder fahreerlose Transportsysteme (FTS) benötigen DV-gestützte Infrastrukturen, die durch einen enormen Informationsaustausch zwischen den genannten CAM-Komponenten (Computer Aided Manufactoring) charakterisiert sind.

- Der zunehmende Konkurrenzdruck zwingt die Unternehmen, kurzfristig auf Kundenaufträge zu reagieren. Dies beinhaltet zum einen die Garantie **kurzer Lieferfristen** bei gleichzeitig hoher Liefertermintreue, zum anderen auch die Möglichkeit, flexibel auf Änderungswünsche durch den Kunden bezüglich der Produktspezifikation eingehen zu können.

- Durch die Anpassung an Kundenwünsche mit detaillierten Anforderungen seitens der Kunden steigt die **Variantenvielfalt** des Produktspektrums. Mit den Spezifikationen erden oftmals genaue **Qualitätsanforderungen** an die Produkte gestellt. Mit der Variantenvielfalt nimmt gleichzeitig auch die Dauer der **Produktlebenszyklen** ab.

- Die Reaktion auf die geänderte Marktsituation und die neuen Fertigungstechnologien führt zu Anpassungen von **Planungs- und Steuerungsphilosophien** bei der Organisation des Fertigungsablaufs (vgl. hierzu Scheer 86). Dies wird bsw. durch den zunehmenden Einsatz fertigungsnaher Informationssysteme wie Leitstände (vgl. hierzu Loos 89) oder durch neue Organisationsformen wie Fertigungsinseln (vgl. hierzu Ruffing 90) dokumentiert.

3.2 Funktionsbereiche von Fertigungsinformationssystemen

Im Rahmen des Computer Integrated Manufacturing sollen alle an der Produktion beteiligten Funktionsbereiche durch computergestützte Technologien unterstützt werden. Dies betrifft neben den betriebswirtschaftlichen Komponenten der Produktionsplanung und -steuerung auch die technischen Bereiche, die mit verschiedenen CAx-Begriffen bezeichnet werden (Scheer 90a, S. 3). Im folgenden sollen alle CIM-Komponenten zu dem Fertigungsbereich gezählt werden, die direkt in die Realisierungsphase der Produktion involviert sind. Dies sind die Fertigungssteuerung, die Betriebsdatenerfassung und die Betriebsdatenverarbeitung sowie das Computer Aided Manufacturing (CAM), d. h. der NC-, CNC-, DNC-Betrieb (NC = Numerical Control, CNC = Computerized Numerical Control) und die Montagesteuerung, die Transportsteuerung, die Lagersteuerung, die Instandhaltung und die Prüfausführung als Teil der Qualitätssicherung (CAQ). In Abbildung 3 sind diese CIM-Komponenten durch eine Schraffierung hervorgehoben (Loos 90a, S. 393, in Anlehnung an: Scheer 90a, S. 2).

Fertigungsaufträge

Fertigungsaufträge für die Produktion von Zwischen- und Endprodukten werden von der Material- und Kapazitätswirtschaft generiert und durch die Auftragsfreigabe der Fertigungssteuerung übergeben. Im Rahmen der Fertigungssteuerung werden diese so auf die vorhandenen Maschinen und Handarbeitsplätze verteilt, daß zum einen die Kapazitäten gleichmäßig ausgelastet sind, zum anderen die Termine der Aufträge eingehalten werden.

Desweiteren muß die Fertigungssteuerung dafür sorgen, daß neben den Maschinenkapazitäten auch andere notwendige Produktionsressourcen wie Rohstoffe, Werkzeuge, Vorrichtungen, NC-Programme und nicht zuletzt auch Personalkapazität bereitgestellt werden.

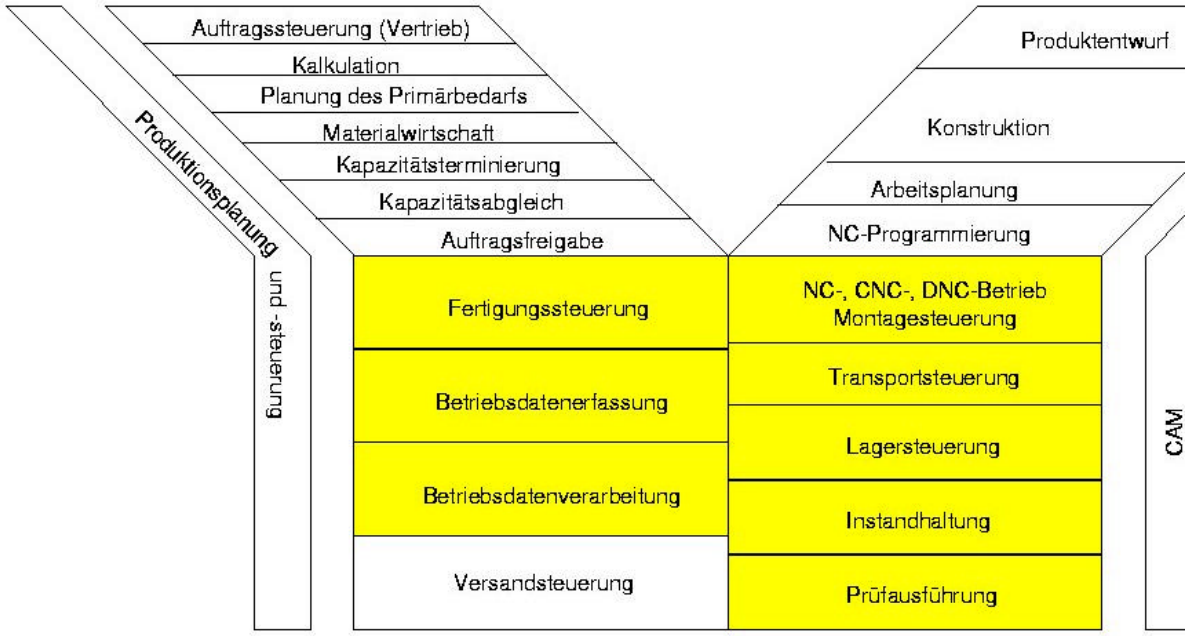


Abb. 3: Informationssystem des Computer Integrated Manufacturing (in Anlehnung an: Scheer 90a, S. 2)

Betriebdatenerfasssung

Betriebsdaten sind alle im Laufe des Produktionsprozesses anfallenden technischen und organisatorischen Informationen, insbesondere über das Verhalten und den Zustand des Betriebs (Loos et al. 89, S. 64). Die Betriebsdatenerfassung beinhaltet alle Maßnahmen, diese Informationen zu sammeln und für die Weiterverarbeitung zur Verfügung zu stellen. Neben den Auftragsdaten werden u. a. material-, maschinen-, werkzeug- und personalspezifische Informationen bereitgestellt.

Betriebsdatenverarbeitung

Die gesammelten Betriebsdaten werden im Rahmen der Betriebsdatenverarbeitung aufbereitet und an verschiedene Funktionsbereiche weitergeleitet. Insbesondere die Fertigungssteuerung mit ihren überwachenden Funktionen ist auf die Bereitstellung

aktueller Informationen angewiesen. Nur so kann der geplante Fertigungsablauf permanent mit dem Betriebsgeschehen abgeglichen werden, und es kann steuernd auf Betriebsstörungen reagiert werden.

Computer Aided Manufacturing

Unter Computer Aided Manufacturing wird die Steuerung von computergestützten Fertigungstechnologien verstanden (Scheer 90a, S. 47 ff). NC- und CNC-Maschinen werden durch Programme gesteuert, so daß keine manuelle Bedienung erforderlich ist. Beim DNC-Betrieb erfolgt eine automatische, zeitgerechte Übertragung der benötigten NC-Programme direkt in die Maschinensteuerung. Transportsysteme befördern die für die Produktion benötigten Materialien wie Rohstoffe und Werkzeuge zeitgerecht zwischen den verarbeitenden Maschinen und den Zwischenlagern. In automatisierten Transportsystemen erfolgt dies mit fahrerlosen Transportsystemen oder festinstallierten Transportstrecken. Automatisierte Lagersysteme verwalten die Stellplätze eines Lagers und steuern die Ein- und Auslagerung. Sie ermöglichen damit eine chaotische Lagerhaltung, bei der die Materialien einem beliebigen freien Lagerplatz zugeteilt werden (Helberg 87, S. 23). Bei Handhabungssystemen werden Funktionen der NC-Maschinen mit automatisiertem Transport und Lagerhaltung in einem System integriert (Helberg 87, S. 22). Diese Form ist insbesondere bei Flexiblen Fertigungszellen und Flexiblen Fertigungssystemen anzutreffen (Scheer 90a, S. 50). Die Aufgaben der Instandhaltung sind neben der Beseitigung von Betriebsstörungen die Planung und Durchführung vorbeugender Wartungsmaßnahmen. Die Prüfausführung überwacht die im Rahmen der Qualitätsplanung festgelegten Anforderungen an Qualitätsmerkmale der hergestellten Produkte.

3.3 Einflußfaktoren auf die Gestaltung von Fertigungsinformations-systemen

Die zu dem Funktionsbereich der Fertigung zählenden Aufgaben unterscheiden sich in ihrer Ausprägung aufgrund unterschiedlicher Einflußfaktoren auf die Fertigungsstrukturen. Dies führt zu unterschiedlichen Anforderungen an die in diesem Bereich eingesetzten Informationssysteme. Von den in der Literatur genannten Kriterien (z. B. Scheer 90f, S. 244) soll im folgenden auf diejenigen eingegangen werden, die starken Einfluß auf die Datenstrukturen besitzen.

Produktart

Aufgrund der Art des hergestellten Produktes läßt sich der Produktionsprozeß unterteilen in , stückgutorientierte Fertigung z. B. Maschinenbau, und in fließgutorientierte Fertigung, zu der u. a. die chemische Industrie zählt. Im Maschinenbau sind häufig große Fertigungstiefen anzutreffen, die zu einer Betonung der Materialwirtschaft mit ausgeprägten Stücklistenstrukturen führen (Scheer 90f, S. 249 - 251). Viele kommerziell verfügbaren Produktionsplanungs- und -steuerungssysteme bieten für diese Aufgabenstellung gute Unterstützung. Bei der Fließfertigung stehen häufig Produktionsprozesse im Vordergrund, die über Rezepturen aus einer beschränkten Anzahl von Rohprodukten eine Vielzahl von Enderzeugnissen herstellen. In der chemischen Industrie besteht schon länger die Notwendigkeit des Chargennachweises, der allerdings durch die Produkthaftungsgesetze auch zunehmend bei der Stückgutfertigung erforderlich wird.

Automatisierungsgrad

Der Automatisierungsgrad der Fertigung wird bestimmt durch den Einsatz unterschiedlicher CAM-Techniken. Der Betrieb entsprechender CAM-Komponenten erfordert allerdings nicht nur den Einsatz der dazu notwendigen Informationssysteme, sondern hat auch Einfluß auf die Funktionen und Datenstrukturen der betriebswirtschaftlichen Funktionsbereiche. So muß bsw. bei Einsatz von NC-Maschinen im Rahmen der Fertigungssteuerung das Vorhandensein der benötigten NC-Programme für die Zuteilung eines Auftrags an eine Maschine berücksichtigt werden. Ein Transportsystem benötigt Informationen aus der Betriebsdatenerfassung über den Abarbeitungszustand eines Auftrags, um nach der Fertigstellung eines Transportloses automatisch einen Transportvorgang auslösen zu können.

Wiederholungsgrad

Hinsichtlich der Wiederholung der Produktion eines Teils kann in Einzel-, Serien- und Massenfertigung unterschieden werden. Bei der **Einzelfertigung** wird ein Endprodukt meistens speziell für einen Kunden hergestellt. Da die Reaktionszeit auf Kundenanforderungen reduziert werden soll, führt dies zu einer Materialdispositions- und Lagerpolitik, bei der die Kundenauftragsbindung möglichst in einer späten Produktionsstufe erfolgt (vgl. hierzu Becker 87). Desweiteren sollte die aktuelle Kundenzuordnung von Fertigungsaufträgen jederzeit ersichtlich sein, um gezielter auf Produktionsstörungen reagieren zu können. Auch kann es notwendig sein, daß mit der Produktion schon vor der endgültigen Spezifikation des Endprodukts begonnen wird, so daß häufig keine Arbeitspläne

vorhanden sind. Bei **Serienfertigung** werden Baugruppen und Endprodukte in Losen gefertigt, deren Größe von den Kriterien Lagerkosten und Rüstkosten abhängig gemacht wird (vgl. hierzu Scheer 90f, S. 130 - 134). Die Fertigungssteuerung hat hierbei das Problem der optimalen Bearbeitungsreihenfolgen unter den konkurrierenden Zielsetzungen Durchlaufzeitreduzierung, Kapazitätsauslastung und Termintreue zu lösen. Bei der **Massefertigung** sind häufig keine Fertigungsaufträge und Arbeitspläne bekannt, sondern die Fertigung wird in Einheiten wie Ausbringung pro Periode gesteuert.

3.4 Anforderungen an das Datenmanagement

Unter Datenmanagement soll die Handhabung der Daten über alle Architekturebenen von Informationssystemen, also die Datenmodellierung, die Umsetzung in Datenbanksysteme sowie die Unterstützung des laufenden Betriebs eines Informationssystems (s. Abbildung 1, S. 4), verstanden werden. Charakteristisch für die Anforderungen an das Datenmanagement aus Sicht der Fertigung sind, bedingt durch die Funktions- und Datenintegration von CIM (s. Abbildung 3, S. 12), sowohl die Problemstellung konventioneller Datenbanksysteme für betriebswirtschaftliche Anwendungen, als auch Probleme, die durch Begriffe wie technische Datenbanken oder Non-Standard-Datenbank verdeutlicht werden. Auf der logischen Ebene der Datenmodellierung lassen sich u. a. folgende Problemstellungen nennen:

- Neben einfachen Strukturen wie beispielsweise einem Arbeitsplatz, der durch skalare Merkmale beschrieben werden kann, müssen **komplexe Strukturen** abbildbar sein. Diese sind u. a. notwendig, um die Strukturbeziehungen eines Schichtmodells oder eine Fertigungszeichnung darstellen zu können.

- Die Beziehungen der Strukturen untereinander gehen häufig über einfache Existenzabhängigkeiten hinaus. Solche **komplizierten Integritätsbedingungen** sind Bestandteil der Datenstrukturen und sollten deshalb auch durch das Datenmodell ausgedrückt werden. So ist es beispielsweise notwendig, bei der Abbildung der Materialzuführung zu Bearbeitungs- und Montageprozessen sicherzustellen, daß diese Komponenten auch den Stücklistenstrukturen entsprechen.

- Obwohl ein Datenmodell statische Strukturen eines Informationssystems beschreibt, ist die Modellierung der **zeitlichen Dimension** notwendig. Die Zeitdimension ist beispielsweise zur Abbildung von Maschinenbelegungsplänen notwendig. Auch der Zugriff auf alte, zu einem früheren Zeitpunkt gültige Datenzustände (Historie) kann erforderlich sein.

Neben den Problemen der Datenstrukturierung sind aus Sicht der Umsetzungs- und der Ausführungsebene folgende Fragen zu berücksichtigen (Loos 90a, S. 412):

- Verteilung von Daten über Rechnernetze,
- Durchführung zeitkritischer Operationen (Echtzeitbetrieb),
- sowohl kurzzeitige als auch langlaufende Transaktionen bei gleichzeitigem Mehrbenutzerbetrieb,
- Ausfallsicherheit und Fehlertoleranz mit schnellen Wiederanlaufzeiten sowie der Betrieb im 24-Stunden-Einsatz.

4. Beschreibungssprachen für die Datenmodellierung

Ziel einer Beschreibungssprache ist die Darstellung von Datenstrukturen auf logischer Ebene, d. h. die Formulierung des Datenmodells. Die Beschreibung der Daten erfolgt dabei außer in einer implementierungsunabhängigen auch in einer funktions- und anwendungsneutralen Form. Da die Strukturierung einerseits Wissen über den zu modellierenden Realitätsausschnitt verlangt und andererseits die Beschreibungssprache als Kommunikationsmittel zwischen der DV-Abteilung und den Fachabteilungen dient, sollte die Repräsentation in einer einfachen, benutzernahen Form erfolgen. Hierfür eignet sich besonders die graphische Darstellungsform. Gleichzeitig soll die Beschreibungssprache das gesamte semantische Wissen über die Daten wiedergeben. Dazu gehören insbesondere:

- Einteilung der Daten in Klassen oder Teilklassen,
- Beziehungen der Klassen und Teilklassen zueinander sowie deren Mächtigkeit,
- Spezifizierung der Klassen und Bezeichnung über Attribute,
- bedingungsabhängige Beschränkungen von Klassenausprägungen und
- Abhängigkeiten und Einschränkungen von Beziehungen zwischen unterschiedlichen Klassen.

Als Beschreibungssprache für die Datenmodellierung hat sich im Allgemeinen die Methode des Entity-Relationship-Modells durchgesetzt. Das Entity-Relationship-Modell ist im Laufe der Zeit um zusätzliche semantische Komponenten erweitert worden bzw. bildete das Grundgerüst für darauf aufbauende Beschreibungsmodelle. Im folgenden sollen diese Modelle und deren Strukturelemente erörtert werden.

4.1 Entity-Relationship-Modell

Das Grundmodell des Entity-Relationship-Modells (ERM) geht zurück auf Chen (Chen 76). Er unterscheidet vier Stufen von Datensichten:

1. umgangssprachliche Informationen über Objekte und Beziehungen,

2. Informationsstrukturen über Objekte und Beziehungen, repräsentiert durch Daten,

3. zugriffspfadunabhängige Datenstrukturen,

4. zugriffspfadabhängige Datenstrukturen.

Ziel des ERM ist die Darstellung der beiden ersten Stufen der Datensicht. Die für die Strukturierung notwendigen Komponenten des ERM sind:

1. Entities und Entitytypen:
   Entities sind konkrete oder abstrakte Objekte der realen Welt, wie beispielsweise eine Maschine, ein Lagerplatz oder ein Qualitätsmerkmal. Sie können durch Eigenschaften und Merkmale beschrieben werden. Lassen sich mehrere Entities durch die gleiche Art (nicht Ausprägung) von Eigenschaften beschreiben, so werden diese zu einem Enitytyp zusammengefaßt, z. B. alle Maschinen zu den Entitytyp Maschine.

2. Beziehungen und Beziehungstypen:
   Beziehungen (relationships) sind Verknüpfungen zwischen zwei oder mehreren Entities. Dabei müssen die verknüpften Entities nicht notwendigerweise zu unterschiedlichen Entitytypen gehören. Beziehungen lassen sich analog zu den Entities zu Beziehungstypen zusammenfassen. Zwischen den gleichen Entitytypen können verschiedene Beziehungstypen existieren, z. B. zwischen den Entitytypen *Mitarbeiter* und *Abteilung* die Beziehungstypen *arbeitet in* und *leitet*.

3. Attribute, Werte und Wertebereiche:
   Die Eigenschaften, die die Entities oder Beziehungen beschreiben, werden Attribute genannt. Ein Attribut einer Maschine ist beispielsweise der Kostensatz. Dem Attribut ist ein bestimmter Wertebereich (Domäne) zugeordnet, z. B. ein DM-Betrag zwischen 10 und 1.000. Für ein Entity "Maschine A567" des Entitytyps *Maschine* ist der Wert des Attributes *Kostensatz* beispielsweise DM 240.

In der graphischen Repräsentation der Datenstrukturen als Entity-Relationship-Modell-Diagramm werden Entitytypen als Rechtecke und Beziehungstypen als Rauten dargestellt (vgl. Abbildung 4).

Die Beziehungen werden nach ihrer Beziehungsart unterschieden in 1:1-, 1:N- und N:M-Beziehungen. Die unterschiedlichen Beziehungsarten sind als Mengendarstellung in Abbildung 5 gezeigt.

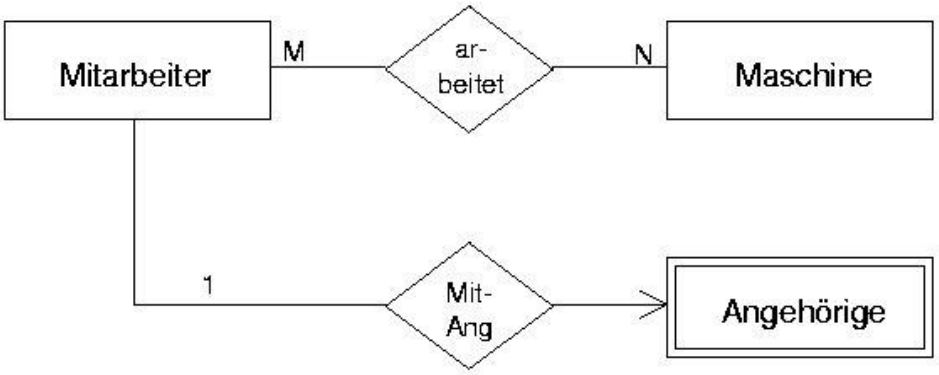


Abb. 4: Entity-Relationship-Modell-Darstellung des Grundmodells

Bei einer 1:1 Beziehung ist jedem Element der Menge A höchstens ein Element der Menge B zugeordnet, und jedem Element der Menge B höchstens ein Element der Menge A. Bei der 1:N-Beziehung können jedem Element der Menge A mehrere Elemente der Menge B zugeordnet werden, aber einem Element der Menge B höchstens ein Element der Menge A. Können dagegen jedem Element der Menge A beliebig viele Elemente der Menge B und jedem Element aus B beliebig viele Elemente aus A zugeordnet werden, so spricht man von einer M:N-Beziehung. Diese Beziehungsarten werden im Entity-Relationship-Modell-Diagramm an den Verbindungslinien zwischen Entitytyp und Beziehungstyp dargestellt (s. Abbildung 4, Beziehung *arbeitet*).

Weiterhin zeigt das ERM-Diagramm die Existenzabhängigkeit eines Entitytyps von einem anderen Entitytyp. Entities des Typs *Angehörige* können nur existieren, wenn sie genau einem Entity des Typs *Mitarbeiter* zugeordnet sind. Graphisch wird dies durch ein doppelumrahmtes Rechteck des abhängigen Entitytyps (Weak Entityset oder schwacher Entitytyp) dargestellt.

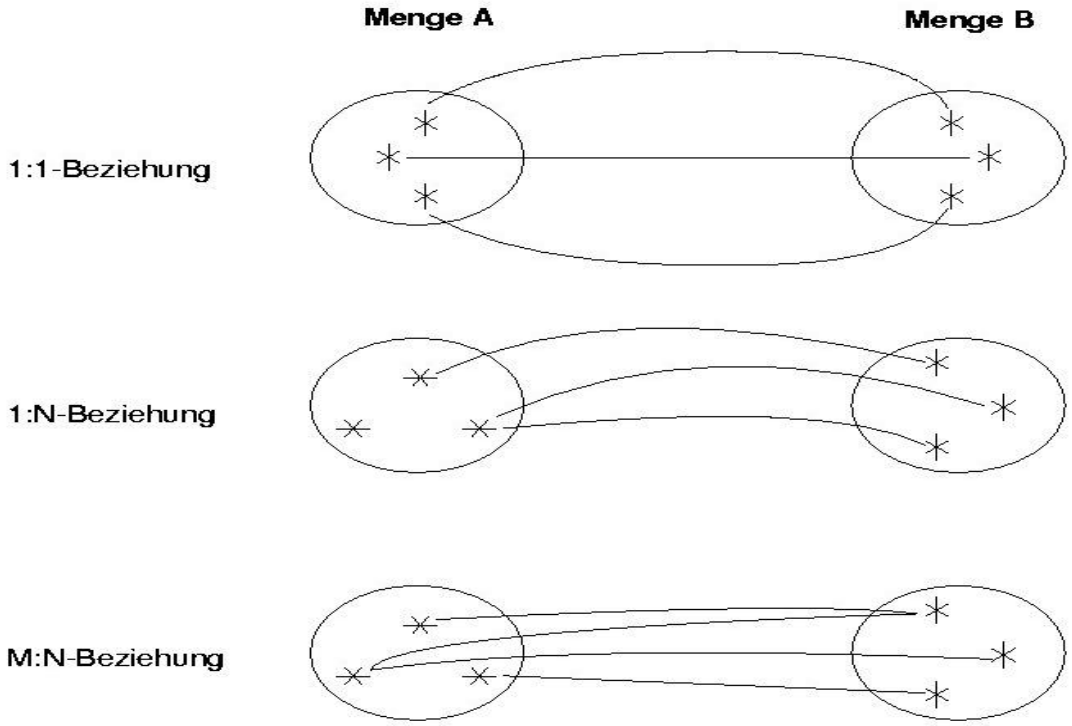


Abb. 5: Mengendarstellung der Beziehungsarten

4.2 Allgemeine Erweiterungen des Entity-Relationship-Modells

Das Entity-Relationship-Modell hat seit seiner Veröffentlichung im Jahr 1976 zahlreiche Erweiterungen und Verfeinerungen durch verschiedene Autoren erfahren. Deshalb muß man, wenn man von dem Erweiterten Entity-Relationship-Modell oder Expented Entity-Relationship-Modell (EERM) spricht, die Weiterentwicklungen im Einzelnen betrachten. Die wichtigsten Ergänzungen, die die semantische Ausdruckskraft des Entity-Relationship-Modells vergrößern und allgemein gebräuchlich sind, sind die Darstellung der Attribute, die Generalisierung, die Uminterpretation von Beziehungstypen sowie der Komplexitätsgrad.

Die bereits bei Chen 1976 betrachteten **Attribute** wurden in anschließenden Veröffentlichungen (z. B. Chen 83) in die graphische Darstellung der Entity-Relationship-Modell-Diagramme aufgenommen. Die Attribute werden dabei in der Regel als Kreise oder Ovale den Entitytypen und Beziehungstypen zugeordnet, wobei Schlüsselattribute unterstrichen werden (s. Abbildung 6).

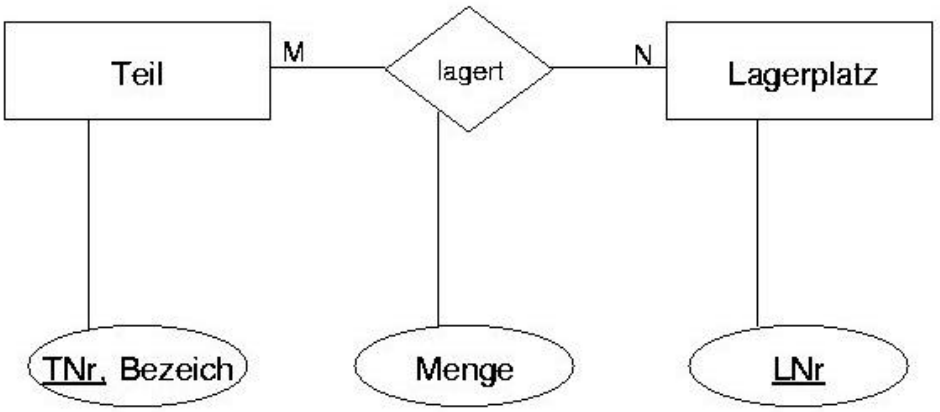


Abb. 6: Attributdarstellung im Entity-Relationship-Modell-Diagramm

Bei der **Generalisierung** werden Mengen unterschiedlicher Klassen, die sich teilweise durch gleiche Attribute beschreiben lassen, zu Obermengen zusammengefaßt. Smith/Smith definieren die Generalisierung wie folgt: "A generalization is an abstraction which enables a class of individual objects to be thought of generically as a single named object" (Smith/Smith 77b, S. 107).
Durch die Bildung generischer Objekte können

- gemeinsame Attribute,
- gemeinsame Beziehungen und
- gemeinsame Operatoren

definiert werden.

Smith/Smith zeigen, daß Generalisierungen in zwei charakteristischen Merkmalen variieren können. Erstens sind nicht alle Subtypen einer generischen Hierarchie disjunkt. Als Beispiel seien alle *Engpaßmaschinen* und alle *NC-Maschinen* zu dem generischen Objekt *Maschine* generalisiert. *Einzelmaschinen* können durchaus sowohl *NC*- als auch *Engpaßmaschinen* sein. Zweitens sind nicht alle Hierarchien echte Baumstrukturen. So kann die *NC-Maschine*, die Element der generischen Struktur *Maschine* ist, gleichzeitig zur generischen Struktur *Automat* gehören (vgl. Abbildung 7).

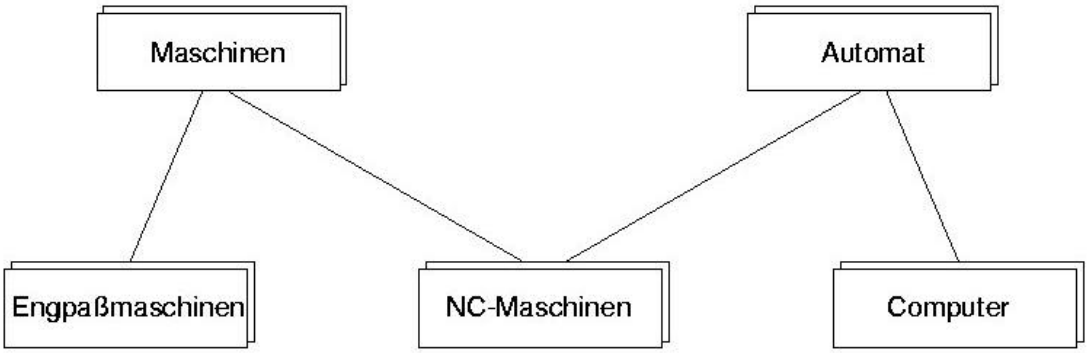


Abb. 7: Beispiele für Generalisierung

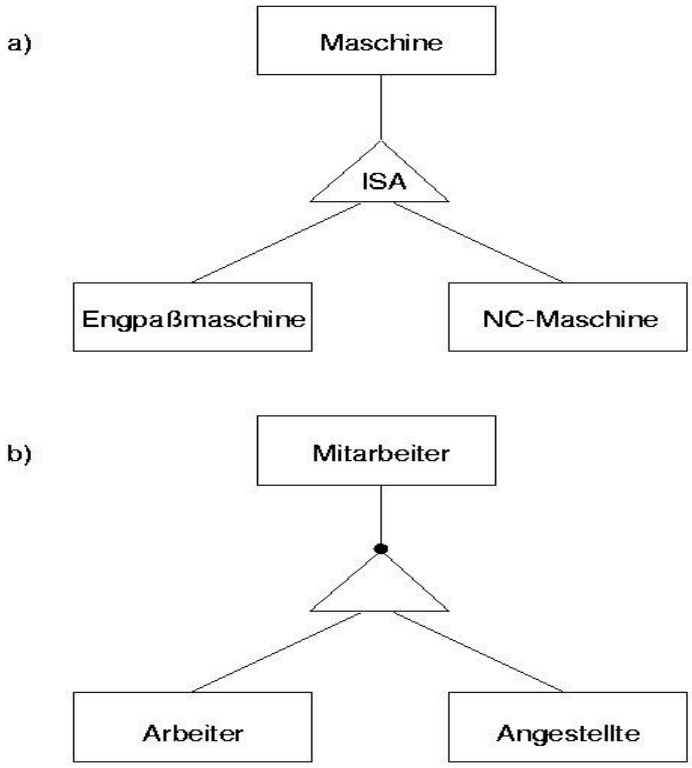


Abb. 8: Generalisierung im ERM

In die Darstellung des Entity-Relationship-Modells wurde die Generalisierung in der Regel als Dreiecksymbol aufgenommen (Dogac/Chen 83). Abbildung 8 a zeigt den generischen Entitytyp *Maschine*. Caldiera und Quitadamo haben die Semantik der Darstellung um disjunkte, vollständige Teilmengen erweitert (Caldiera/Quitadamo 83; vgl. auch Wagner 88). Abbildung 8 b zeigt den generischen Entitytyp *Mitarbeiter*, der aus den Subtypen *Arbeiter* und *Angestellte* zusammengesetzt ist. Ein *Mitarbeiter* muß genau zu einem Subtyp gehören.

Betrachtet man die Abstraktion in umgekehrter Richtung von dem generischen Entitytyp aus zu den Subtypen, so spricht man von der **Spezialisierung** (Scheer 90f, S. 27).

Hohenstein et al. verallgemeinern den Begriff der Generalisierung (Hohenstein et al. 87; Hohenstein/Gayolla 88). An die Stelle der Subtypen und des generischen Typs treten Input- und Outputtypen. Die Generalisierungsoperation muß zwei Bedingungen genügen:

- Die Menge der Entities aller Inputtypen I muß die Menge der Entities aller Outputtypen O_m enthalten. Es gilt:

  $$I_1 \cup I_2 \cup ...I_n \supseteq O_1 \cup O_2 \cup ...O_m$$

- Ein Entity eines Outputtyps O_i darf zu keinem anderen Outputtyp gehören, d. h. die Outputtypen müssen paarweise disjunkt sein. Es gilt:

  $$O_i \cap O_j = \varnothing \quad \forall \; i,j \in \{1,...,m\} \wedge i \neq j$$

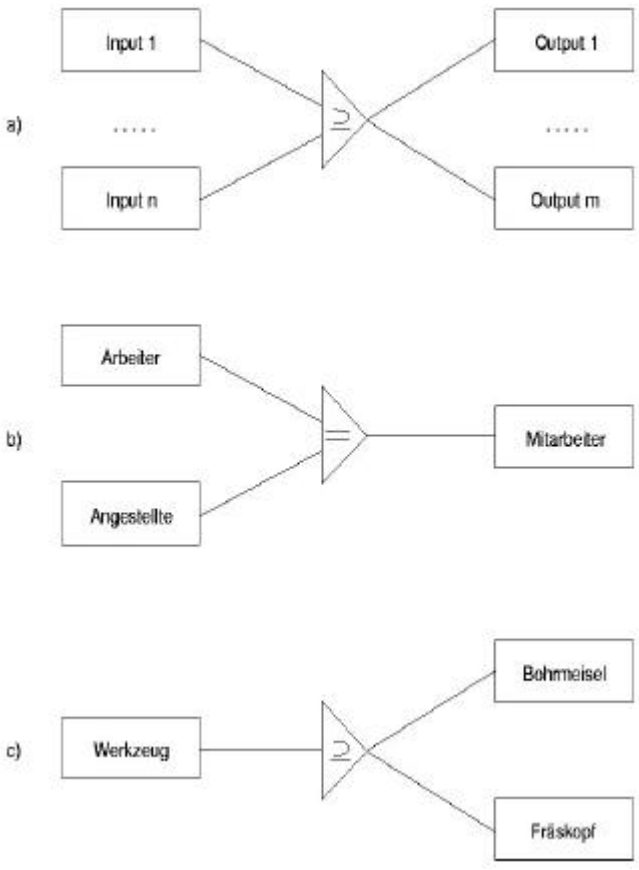


Abb. 9: Generalisierung/Spezialisierung mit Input- und Outputtypen

Abbildung 9 a zeigt die allgemeine Form der Generalisierung. Abbildung 9 b zeigt das bekannte Beispiel aus Abbildung 8 b. Teil c der Abbildung 9 zeigt eine Spezialisierung des Entitytyps *Werkzeug*. *Bohrmeisel* und *Fräskopf* sind jeweils disjunkt, stellen aber nicht alle denkbaren Werkzeuge dar.

Beziehungstypen entstehen als Aggregation zweier oder mehrerer Entitytypen. Die Aggregation kann ihrerseits in einem folgenden Konstruktionsprozeß wieder Ausgangspunkt weiterer Beziehungstypen sein. Um die Möglichkeit zu schaffen, daß Beziehungstypen selbst weitere Beziehungen eingehen können, wurde die **Uminterpretation eines Beziehungstyps** in einen Entitytyp eingeführt (Schlageter/Stucky 83) und als rechteckig umrahmte Raute in das ERM-Diagramm aufgenommen (Webre 83). Scheer weist darauf hin, daß es notwendig ist, den Konstruktionsprozeß des uminterpretierten Beziehungstyps zu erkennen (Scheer 90f, S. 34). Dafür sollen die Verbindungskn der an der Aggregation beteiligten Entitytypen graphisch bis zur Raute geführt werden, die Kanten der Beziehungstypen des uminterpretierten Beziehungstyps jedoch nur bis zum Rechteck (vgl. Abbildung 10).

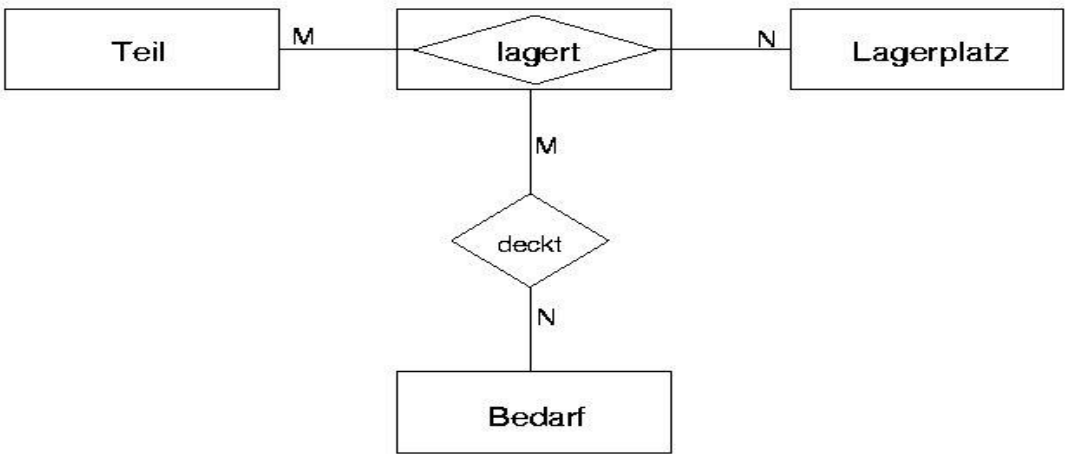


Abb. 10: Uminterpretation eines Beziehungstyps zum Entitytyp

Der gleiche Sachverhalt wird auch dargestellt (vgl. Abbildung 11), indem die Aggregation einschließlich der beteiligten Entitytypen in den neuen Beziehungstyp eingehen (vgl. auch Briand et al. 88, S. 290; Put 88, S. 285).

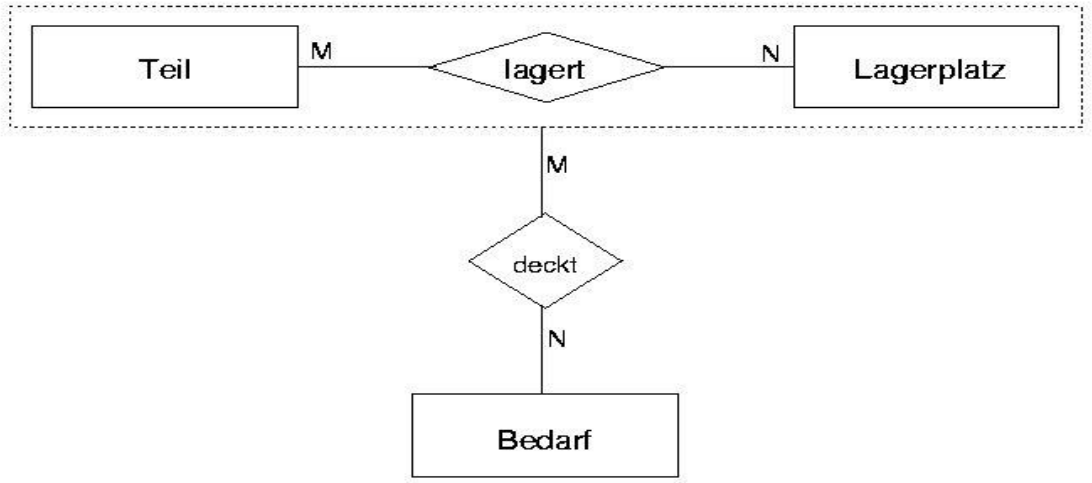


Abb. 11: Aggregation als Pseudo-Entitytyp

Die von Chen eingeführte Beschreibung der Beziehungsarten 1:1, 1:N und M:N stellen Obergrenzen für die Anzahl der Beziehungen eines Entities des Typs A zu Entities des Typs B dar. Bei einem schwachen Entitytyp wird neben der Obergrenze von eins auch gleichzeitig eine Untergrenze von eins impliziert, da ein schwaches Entity ohne eine Beziehung definitionsgemäß nicht existieren kann. Die Darstellung hat die Nachteile, daß die Definition der Untergrenze bei mehreren Beziehungstypen des schwachen Entitytyps mehrdeutig wird, eine gegenseitige Abhängigkeit von Entitytypen nicht vorgesehen ist und daß bei mehreren Beziehungstypen zwischen zwei Entitytypen die Eindeutigkeit verloren geht, wenn ein Beziehungstyp existentiell ist (Webre 83, S. 175 - 176).

Abbildung 12 a zeigt die existentielle Abhängigkeit des Entitytyps E2 von dem Typ E1 über den Beziehungstyp B (es gilt E1 … E2). Beispielsweise sei E1 der Typ *Kostenstellen* und E2 der Typ *Maschine*. Eine Maschine muß genau einer Kostenstelle zugeordnet werden, während einer Kostenstelle keine oder bis maximal 15 Maschinen zugeordnet werden können. Der **Komplexitätsgrad** kgrad der Beziehung ist damit kgrad (E1,B) = (0,15) und kgrad (E2,B) = (1,1). Der Komplexitätsgrad einer Aggregation wird auch Kardinalität genannt.

Die Ungenauigkeiten dieser Darstellung werden durch die (min,max)-Notation beseitigt (Schlageter/Stucky 83). Dabei wird der Komplexitätsgrad der Beziehung an die Kanten zwischen Entitytyp und Beziehungstyp geschrieben (vgl. Abbildung 12 b). Auch bei der (min,max)-Notation kann für eine beliebige Obergrenze der Wert "n" angegeben werden.

Lenzerini und Santucci erweiterten in ihrem Ansatz die Untergrenze des Komplexitätsgrads um die optionale Beziehung (Lenzerini/Santucci 83). Beispielsweise bedeutet ein Komplexitätsgrad opt(5,10), daß ein Entity entweder keine Beziehungen oder mindestens fünf bis maximal zehn Beziehungen eingehen kann.

Eine Verallgemeinerung stellt die 1,c,m Notation dar (wobei "c" für choice und "m" für multiple steht). Damit können vier Komplexitätsgrade ausgedrückt werden (vgl. auch Abbildung 12 c):

- 1 für (1,1),
- c für (0,1),
- m für (1,n) und
- cm für (0,n).

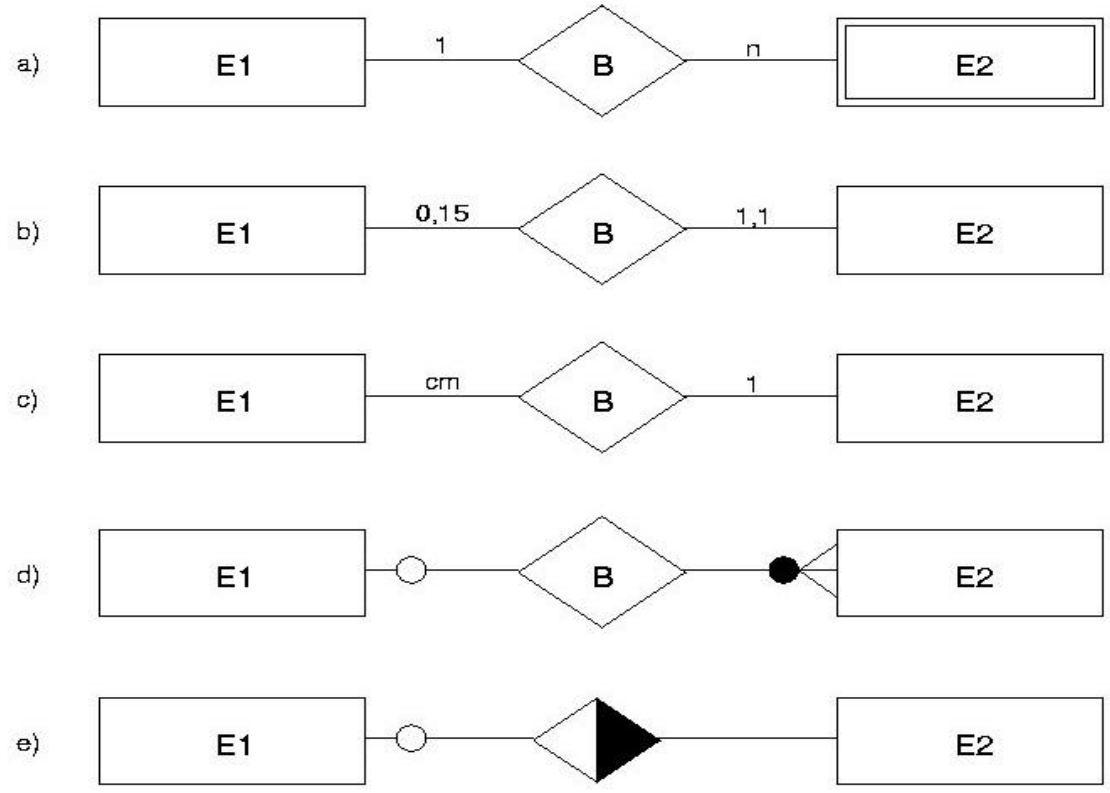


Abb. 12: Beispiele für Erweiterungen der graphischen Darstellung bzgl. des Komplexitätsgrades

Die 1,c,m-Notation wird häufig auch direkt in graphische Symbole umgesetzt. Dabei werden Komplexitätsgrade von "n" als "Krähenfüße" (z. B. bei Dampney 86) oder als schattierte Rauten (z. B. bei Teorey et al. 86) und optionale Beziehungen als offene Punkte dargestellt (vgl. Abbildung 12 d und 12 e). Zu beachten ist, daß die Seiten, an denen der Komplexitätsgrad angegeben wird, bei den einzelnen Darstellungsarten nicht einheitlich sind (vgl. hierzu Schlageter/Stucky 83; Scheer 90f, S. 36 - 39).

4.3 Strukturiertes Entity-Relationship-Modell

Aufbauend auf dem Entity-Relationship-Modell entwickelte Sinz das Strukturierte Entity-Relationship-Modell **SERM** (Sinz 87). Ausgangspunkte für die Weiterentwicklung des Modells waren u. a. die bessere Visualisierbarkeit der Existenzabhängigkeiten von Entitytypen und die Verhinderung versehentlich modellierter bei ein- oder wechselseitigen Abhängigkeiten (Sinz 88). Um dies zu erreichen, wird das Konzept quasi-hierarchischer Graphen in das Entity-Relationship-Modell übernommen.

Die Objekttypen des Strukturierten ERM sind der Entitytyp (E-Typ), der Relationship-Typ (R-Typ) und der Entity-Relationship-Typ(ER-Typ), der dem zum Entitytyp uminterpretierten Beziehungstyp des Erweiterten ERM entspricht. Die Beziehungen (Kanten) zwischen den Objekttypen werden in einer graphischen (1,c,m)-Notation dargestellt. Zusätzlich enthält das SERM den Konstruktionsoperator Generalisierung. Sinz unterscheidet die IS-A-Hierarchie und die Subtyp-Hierarchie . Unter einer IS-A-Hierarchie wird dabei die Zusammenfassung von paarweise disjunkten Teilmengen zu einer Oberklasse verstanden, wenn jedes Entity der Obermenge auch Element einer Teilmenge ist (Vollständigkeit). Bei einer Subtyp-Hierarchie wird die Vollständigkeit nicht gefordert. Abbildung 13 zeigt die im SERM verwendeten Symbole.

Da das Diagramm des SERM ein gerichteter Graph ist, gelten für die Modellierung folgende Regeln:

- Kanten sind immer von links nach rechts gerichtet, d. h. links ist der Startknoten, rechts der Zielknoten angeordnet (R1).

- Zwischen zwei Knoten sind mehrere Kanten zulässig (R2).

- Ein R-Typ kann nur Zielknoten mindestens zweier Kanten sein (R3).

- Ein E-Typ kann nur Startknoten sein (R4).

Das SERM-Diagramm hat eine eindeutige Referenzrichtung. Alle links angeordneten E-Typen stellen Entitytypen des ERM dar, die unabhängig von Beziehungen existieren können. Alle ER- und R-Typen sind in ihrer Existenz von den links angeordneten E- oder ER-Typen abhängig, mit denen sie über Kanten verbunden sind.

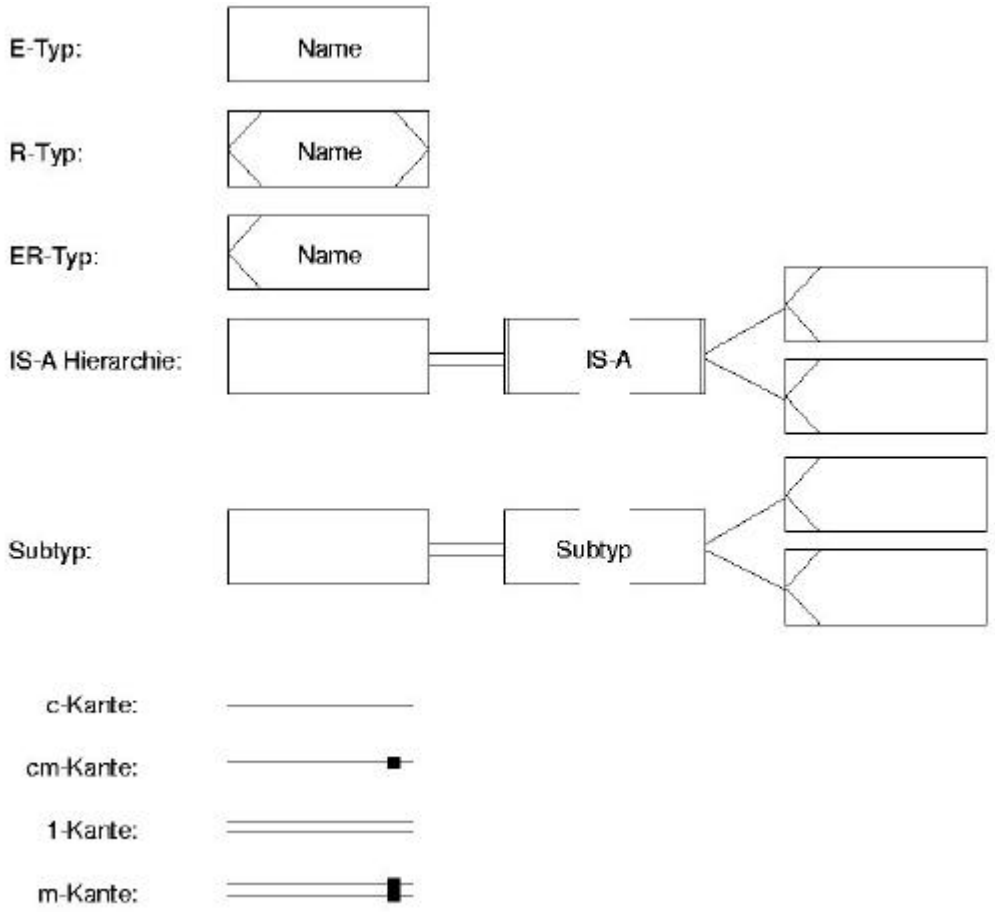


Abb. 13: Symbole des SERM-Diagramms

Abbildung 14 zeigt zwei SERM-Diagramme, wobei Teil a den Sachverhalt aus Abbildung 4 und Teil b aus Abbildung 10 wiedergibt. Der schwache Entitytyp *Angehörige* wird im SERM

in einen ER-Typ überführt und ist damit vom E-Typ *Mitarbeiter* abhängig. Der zum Entitytyp uminterpretierte Beziehungstyp *lage*r*t* wird im SERM als ER-Typ dargestellt und ist damit selbst existentielle Voraussetzung für den R-Typ *deckt*.

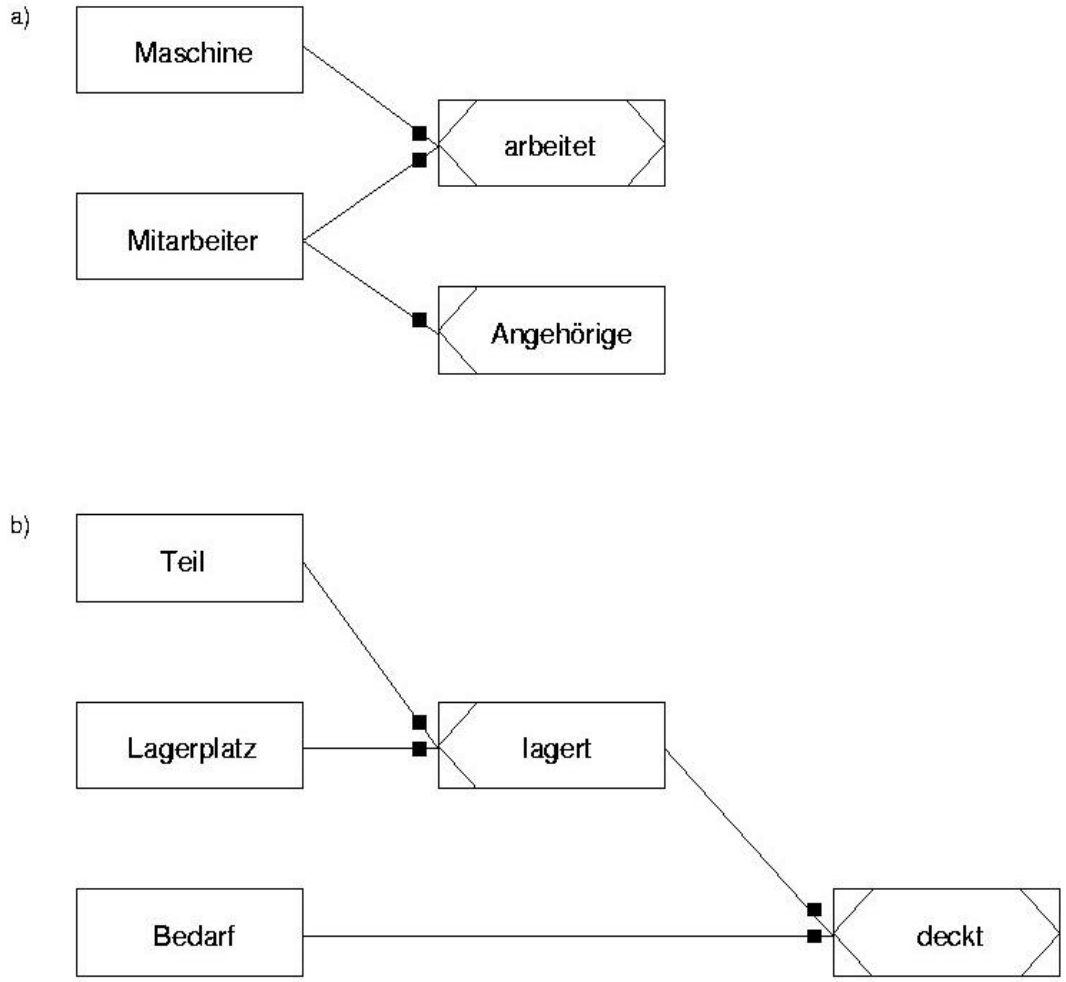


Abb. 14: Beispiele für SERM-Diagramme

4.4 Entity-Category-relationship-Modell

Auch das Entity-Category-Relationship-Modell (**ECRM**, auch Entity-Category-Relationship-Datenmodell) stellt eine Weiterentwicklung des ERM dar. Das Hauptziel des ECRM ist die Erweiterung des ERM um die Generalisierung sowie um die Möglichkeit, verschiedene Entitytypen in der gleichen Kantenrolle in einen Beziehungstyp eingehen zu lassen (Elmasri et al. 85). Dazu wird das Konzept der Kategorie(Category) eingeführt. Außerdem wird die Mächtigkeit der Attributdefinition um und Funktionen erweitert.

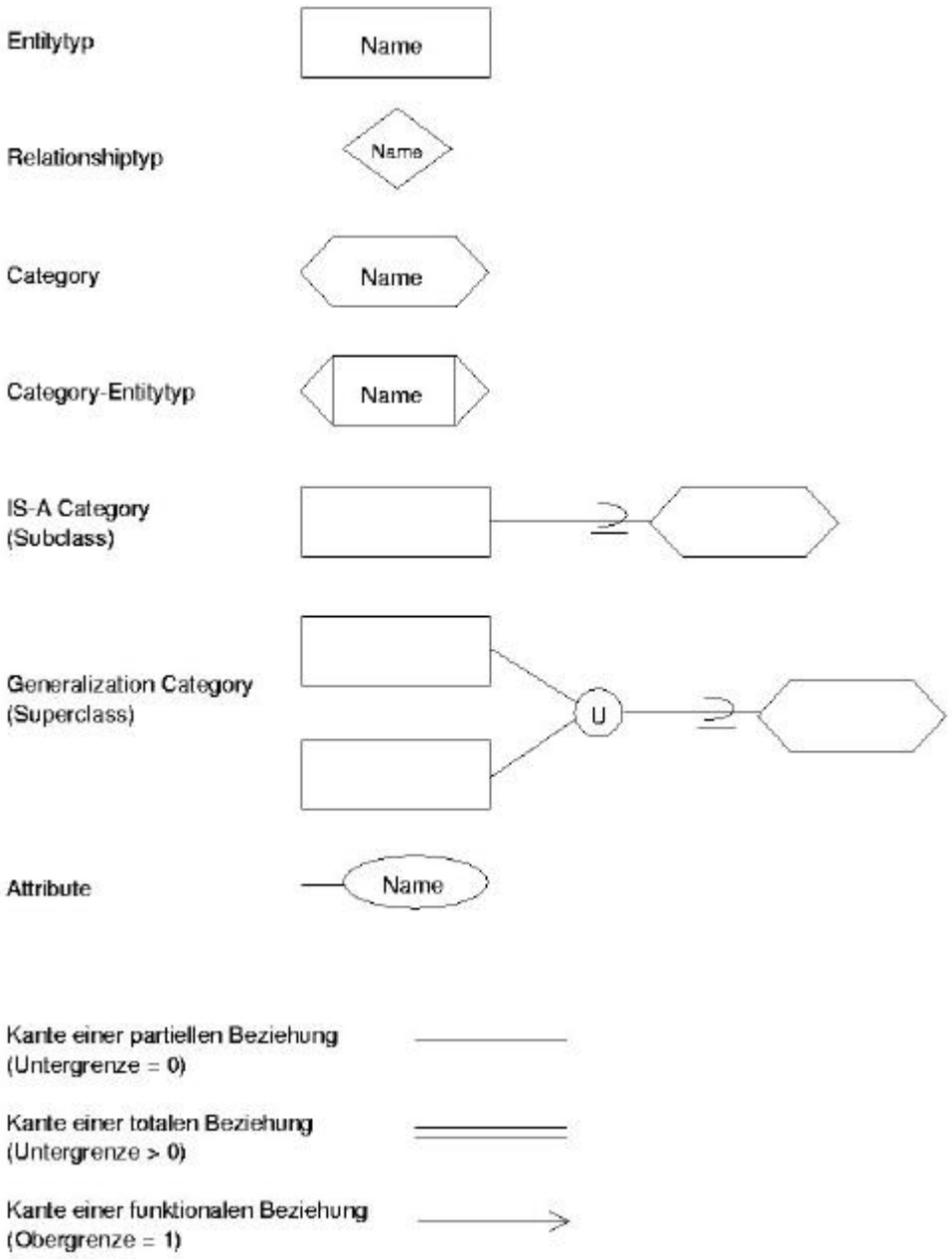


Abb. 15: Symbole des ECRM-Diagramms

Entities mit gleichen Attributen werden analog zum ERM in Entitytypen zusammengefaßt. Entities, die die gleiche Rolle in Verbindungen zu Beziehungstypen einnehmen, werden zu einer Category zusammengefaßt. Categories können wie Entitytypen Attribute besitzen und bestehen wie diese aus einer Menge von Entities. Ein Entity gehört damit zu einem Entitytyp und zu einer von der Anzahl der verschiedenen eingegangenen Beziehungstypen abhängigen Menge von Categories. Dabei erbt das Entity sowohl die Attribute des Entitytyps als auch die der Categories. Gehen die Entities eines Entitytyps alle die gleichen Arten von

Beziehungen ein, so ist der Entitytyp gleichzeitig eine Category. Abbildung 15 zeigt die im ECRM-Diagramm verwendeten Symbole.

Abbildung 16 zeigt ein ECRM-Diagramm, das den Sachverhalt aus Abbildung 4 wiedergibt. *Mitarbeiter*, *Maschine* und *Angehörige* sind Entitytypen, da sie eine Zusammenfassung von Entities mit gleicher Attributbeschreibung sind. Gleichzeitig stellen sie auch Categories dar, da alle Entities je Entitytyp die gleichen Beziehungen eingehen. Die Kantenbeschriftung ist nach der (min,max)-Notation vorgenommen worden, wie sie von Navathe et al. benutzt wurde (Navathe et al. 86).

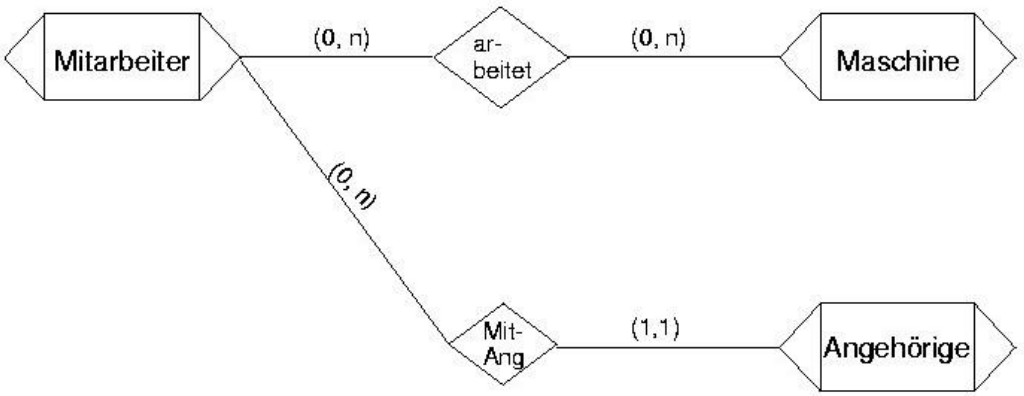


Abb. 16: ECRM-Diagramm mit Category-Entitytyp

Superclass- und Subclass-Beziehungen, die die Generalisierungsoperation ausdrücken, werden im ECRM ebenfalls mit Hilfe der Category ausgedrückt. In Abbildung 17 ist je ein Beispiel für eine Generalization Category(a) sowie eine ISA Category (b) dargestellt. Bei der Generalization Category werden die beiden Entitytypen *Maschine* und *Werkzeug*, die jeweils durch eigene Attribute beschrieben werden, zu der Category *Ressource* zusammengefaßt, da sie beide die gleiche Beziehung zu dem Category-Entitytyp *Auftrag* eingehen.

In der ISA Category geht ein Teil der Entities des Typs *Maschine* eine Beziehung zum Category-Entitytyp *NC-Programm* ein, und zwar genau die Maschinen, die gleichzeitig auch *NC-Maschine*n sind. Eine andere Teilmenge der Maschinen bildet die ISA Category *Engpass*, für die eigene Attribute vergeben werden, die für sonstige Maschinen bedeutungslos sind.

Für die Beschreibung weiterer Integritätsbedingungen sowie als Abfragesprache wurde, aufbauend auf dem Entity-Category-Relationship-Modell, die Sprache GORDAS(Graph Oriented Data Selection) entwickelt (Elmasri 85, S. 99 ff). Als mathematisch-formale

Sprache ist sie allerdings nicht integraler Bestandteil des ECRM-Diagramms und soll deshalb an dieser Stelle nicht weiter behandelt werden.

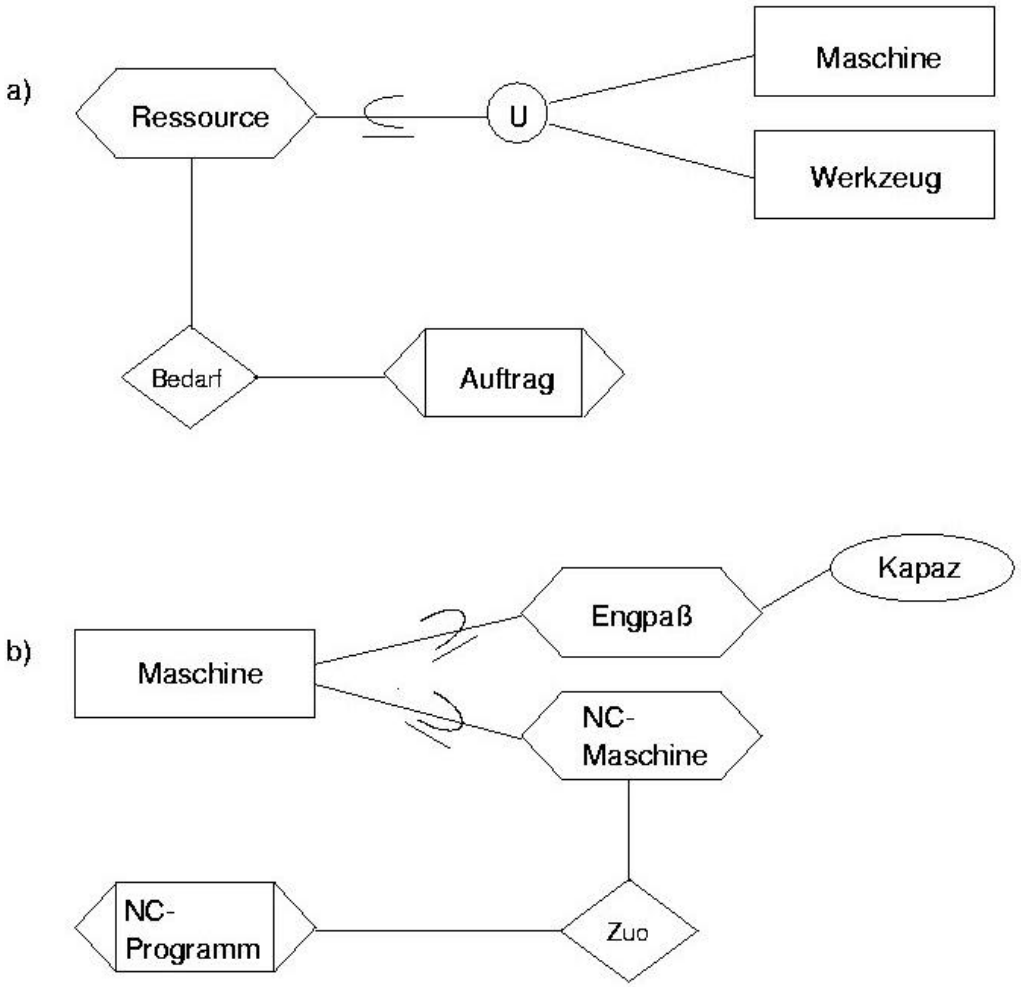


Abb. 17: ECRM-Diagramm mit Generalization Category(a) und ISA Category (b)

4.5 Binäre Relationship-Modelle

Binäre Relationship-Modelle lassen im Gegensatz zu dem klassischen Entity-Relationship-Modell nur Beziehungen zwischen zwei Objekten zu (Chen 83) und zeichnen sich durch die Modellierungsmöglichkeiten vielfältiger Integritätsbedingungen aus (DeTroyer 89). Graphisch lassen sich Binäre Relationship-Modelle durch **NIAM**-Diagrammee (Nijssens Information Analysis Method) darstellen (Verheijen/VanBekkum 82).

Die NIAM-Methode unterscheidet generell zwei Arten von Objekttypen sowie eine Mischform. Der erste Typ ist der sogenannte Non-lexical Object Type(NOLOT). In dieser Klasse werden gleiche, nicht-lexikalische Objekte zusammengefaßt, z. B. Maschinen. NOLOT sind den Entitytypen des ERM ähnlich. Die zweite Typenklasse ist der Lexical Object Type(LOT), der vergleichbar ist mit den Attributen des ERM, z. B. Maschinenbezeichnung. Die Mischform LOT-NOLOT wird dann eingesetzt, wenn die Unterscheidung zwischen lexikalischem und nicht-lexikalischem Objekt für den darzustellenden Sachverhalt nicht relevant ist.

Zwischen genau zwei Objekten kann eine Beziehung hergestellt werden (daher binäres Relationship-Modell), die als bezeichnet wird. Ein LOT muß allerdings immer in genau ein Fact eingehen, und zwar mit einem NOLOT-Objekt. Knappe et al. bezeichnen die Facts zwischen NOLOTs als "Idee", Facts zwischen einem NOLOT und einem LOT als "Brücke" (Knappe/Suer 88). Die Beziehungsart und der Komplexitätsgrad von Facts werden durch Rules beschrieben.

Zwischen zwei NOLOT oder zwei LOT kann eine Subtyp-Beziehung hergestellt werden, mit der die Generalisierung modelliert werden kann. Abbildung 18 zeigt die Symbole der NIAM-Notation (Knappe/Suer 88; Lin et al. 88).

Integritätsbedingungen können zwischen zwei oder mehreren Rules unterschiedlicher Facts, zwischen zwei oder mehreren Facts oder zwischen zwei oder mehreren Subtypen einer Objektklasse bestehen. Bei Bedingungen zwischen Rules unterschiedlicher Facts gilt (vgl. Bedingung Y1 in Abbildung 18):

(1) Bei einer Exklusion (Typ X) darf jedes Element a_i der Objektklasse A1 nur in einer der Rules r_1 oder r_3 vorkommen. Es gilt:
$r_1(A1) \cap r_3(A1) = \varnothing$

(2) Bei der Vollständigkeit (Typ C = Cover) müssen alle Elemente a_i der Objektklasse A1 eine Beziehung r_1 oder r_3 vorkommen. Es gilt:
$r_1(A1) \cup r_3(A1) = A1$

(3) Bei der n Bedingung (Typ T) müssen die Bedingung der Exklusion und die der Vollständigkeit erfüllt sein, d. h. jedes a_i der Klasse A1 muß genau in einer Rule r_1 oder r_3 enthalten sein.

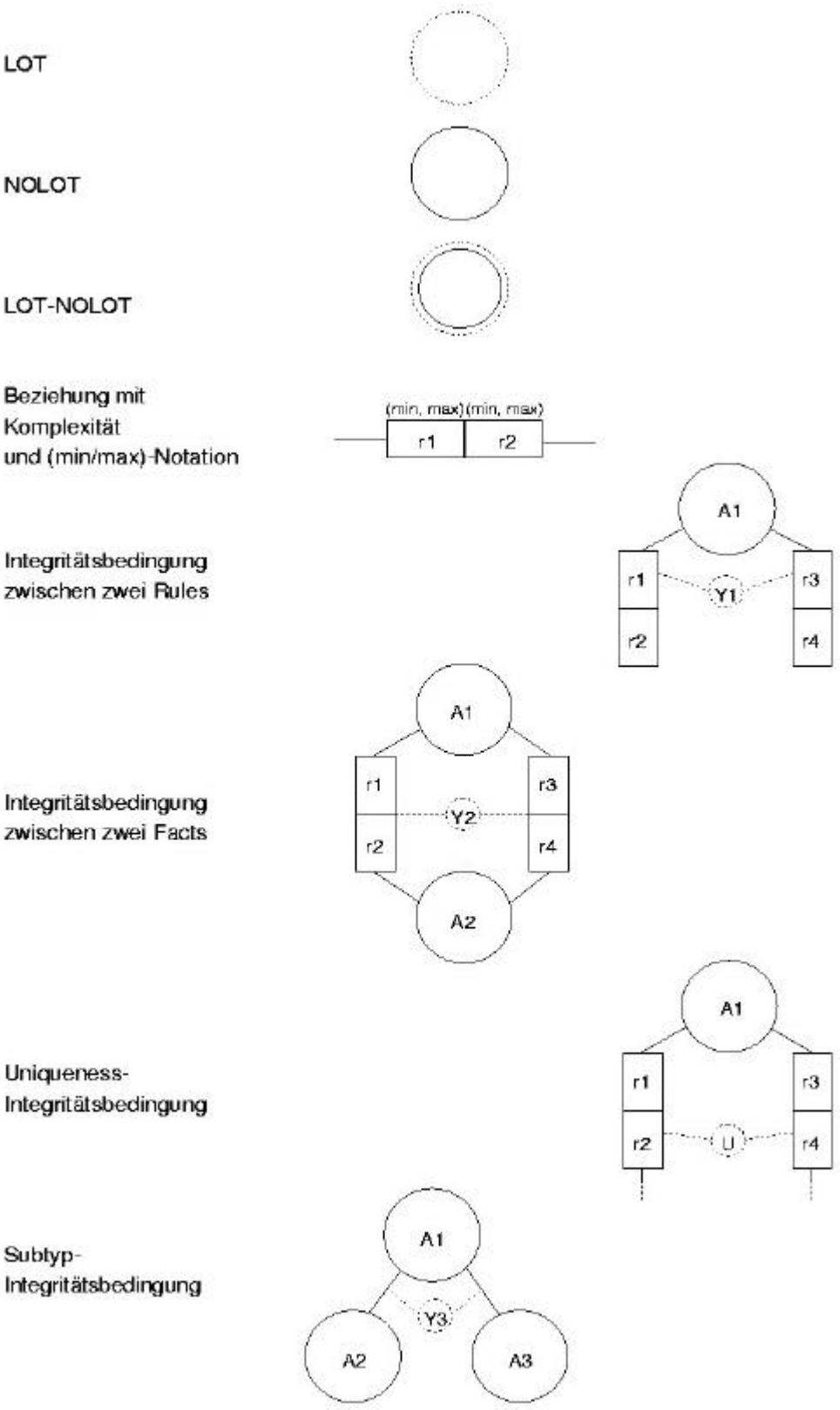


Abb. 18: Symbole der NIAM-Notation

(4) Bei den Subset-Bedingungen (Typ $\subseteq$) müssen alle a_i der Klasse A1, die in Rule r_1 vorkommen, auch in Rule r_3 enthalten sein. Es gilt:
$r_1(A1) \subseteq r_3(A1)$
Die Richtung des Subsets ist zu beachten.

(5) Bei den Gleichheitsbedingungen (Typ E = Equality) müssen alle a_i der Klasse A1 in r_1 auch in r_3 enthalten sein und umgekehrt. Es gilt:
$r_1(A1) = r_3(A1)$

Bei den Integritätsbedingungen zwischen Facts könne die gleichen Arten

- Exklusion,
- Vollständigkeit,
- totale Bedingung,
- Subset-Bedingung und
- Gleichheitsbedingung

wie bei den Rule-Integritätsbedingungen unterschieden werden (vgl. Bedingung Y2 in Abbildung 18). Dabei gilt allerdings, daß beide Rules der beteiligten Facts berücksichtigt werden. Am Beispiel der Exklusion bedeutet dies, daß ein Element a_1 aus der Klasse A1 und ein Element a_2 aus der Klasse A2 entweder über den Fact mit den Rules r_1 und r_2 oder über den Fact mit den Rules r_3 und r_4 in Beziehung stehen. Es gilt:
$\{(a_1,a_2) \mid a_1 \in r_1(A1) \wedge a_2 \in r_2(A2)\} \cap$
$\{(a_1,a_2) \mid a_1 \in r_3(A1) \wedge a_2 \in r_4(A2)\} = \varnothing$

Bei den Subtyp-Integritätsbedingungen können 3 Arten unterschieden werden (vgl. Bedingung Y3 in Abbildung 18):

(1) Bei der Exklusion darf ein Objekt a nur in der Subtypmenge A2 oder A3 vorkommen. Es gilt:
$A2 \cap A3 = \varnothing$

(2) Bei der Vollständigkeit müssen alle Objekte der Klasse A1 entweder in den Subtypklassen A2 oder A3 enthalten sein. Es gilt:
$A2 \cup A3 = A1$

(3) Bei der totalen Bedingung gelten sowohl die Vollständigkeit als auch die Exklusion, d. h. jedes Objekt der Klasse A1 muß in genau einer Subtypklasse vertreten sein.

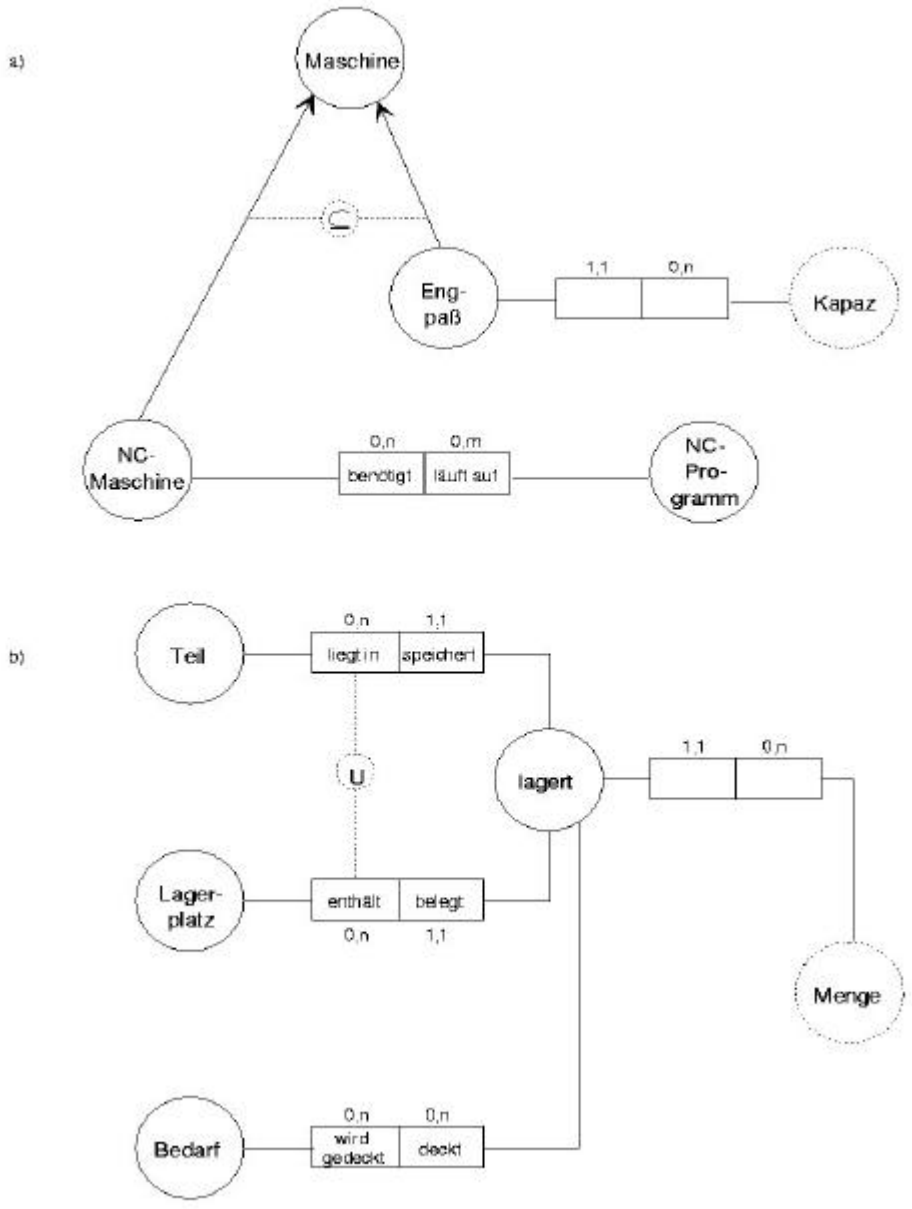


Abb. 19: Beispiele für NIAM-Diagramme

Abbildung 19 a zeigt das leicht abgewandelte Beispiel aus Abbildung 17. Das Attribut *Kapaz* ist als LOT dem NOLOT *Engpass* zugeordnet. Jedem Objekt aus *Engpass* wird genau ein Objekt aus *Kapaz* zugeordnet. Der Fact zwischen *NC-Maschine* und *NC-Programm* weist eine Komplexität von (0,n) zu (0,m) auf. Die Subtypen sind so definiert, daß eine *NC-Maschine* gleichzeitig auch immer eine *Engpaß*maschine ist, *Engpaß*maschinen aber auch andere Objekte enthalten können. In Abbildung 19 b ist das Beispiel aus Abbildung 10 dargestellt. Der NOLOT *lagert* kann nur existieren, wenn er genau eine Verbindung zu den NOLOTs *Teil* und *Lagerplatz* eingeht sowie zu dem LOT *Menge*. Gleichzeitig wird über die die Uniqueness- Integritätsbedingung die Primärschlüsseleigenschaft für das NOLOT *lagert* definiert.

Shoval/Even-Chaime setzen NIAM-Diagramme ohne Unterscheidung von LOT und NOLOT ein (Shoval/Even-Chaime 89). Es existiert nur eine Klasse von Objekttypen (OT). Damit bildet das NIAM ein binäres ERM ohne Attribute (vgl. Chen 83). Gleichzeitig kann ein OT aber auch ein Fact sein. Dies entspricht der Uminterpretation eines Beziehungstyps zum Entitytyp im Erweiterten Entity-Relationship-Modell.

Abbildung 20 zeigt das Beispiel aus Abbildung 19 b. Im Gegensatz zu Abbildung 19 b sind *Teil* und *Menge* von der gleichen Klasse OT. *Lagert* ist gleichzeitig ein Fact und ein OT.

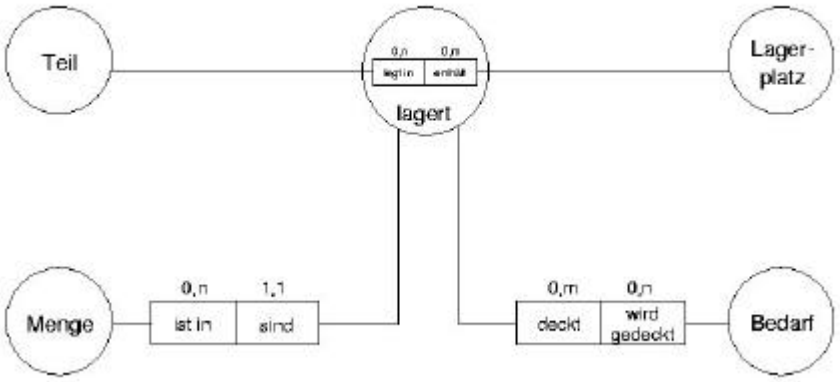


Abb. 20: NIAM-Darstellung ohne Attribute

4.6 Entitäten-Diagramme

Thurnherr und Zehnder erweiterten das Relationenmodell (Codd 70) um seine semantische Aussagekraft (Thurnherr 80; Zehnder 83). Obwohl es als Beschreibungssprache abhängig vom Relationenmodell als Modell eines Datenbanksystems ist, soll es aufgrund seiner graphischen Darstellungsmittel, den Entitäten-Diagrammen (**ED**), und den semantischen Erweiterungen in diesem Ansatz diskutiert werden (im Gegensatz zu anderen Erweiterungen wie z. B. dem RM/T (Codd 79) oder dem SDM (Hammer/McLeod 81), die über keinegraphischen Darstellungsmittel verfügen).

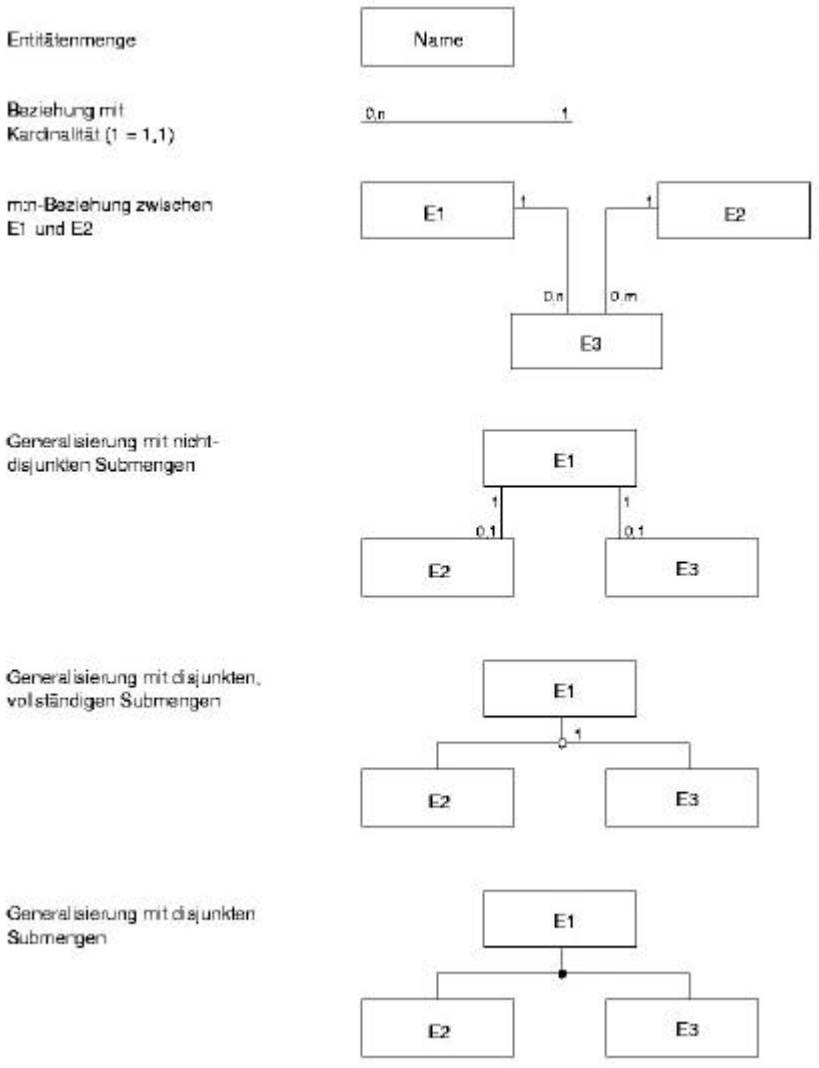


Abb. 21: Symbole des Entitäten-Diagramms

Entitätsmengen entsprechen den im Entity-Relationship-Modell vorgestellten Entitytypen (bzw. den Relationen im Relationenmodell). Sie sind eine Klasse von Entitäten, die durch gleiche Attribute beschrieben werden. Zur Identifizierung einer Entität innerhalb einer Entitätenmenge dient ein Attribut oder eine Attributskombination, deren Wert sich während der Existenz der Entität nicht ändern darf. Zwischen Entitätsmengen können Beziehungen bestehen. Nichthierarchie-Beziehungen, d. h. (m:n)-Beziehungen werden durch Einführung einer zusätzlichen Entitätsmenge in zwei (1:n)-Beziehungen aufgelöst (vgl. Abbildung 21). Rekursive Beziehungen werden prinzipiell aufgelöst, so daß eine Ordnung entsteht. Die Komplexität der Beziehungen wird mit Unter- und Obergrenzen angegeben.

Entitätsmengen können zu einer neuen Entitätsmenge generalisiert werden. Bei der Generalisierung wird zum einen unterschieden, ob die Submengen disjunkt oder überlappt sind und zum anderen, ob die Generalisierung vollständig in den Submengen enthalten sein muß.

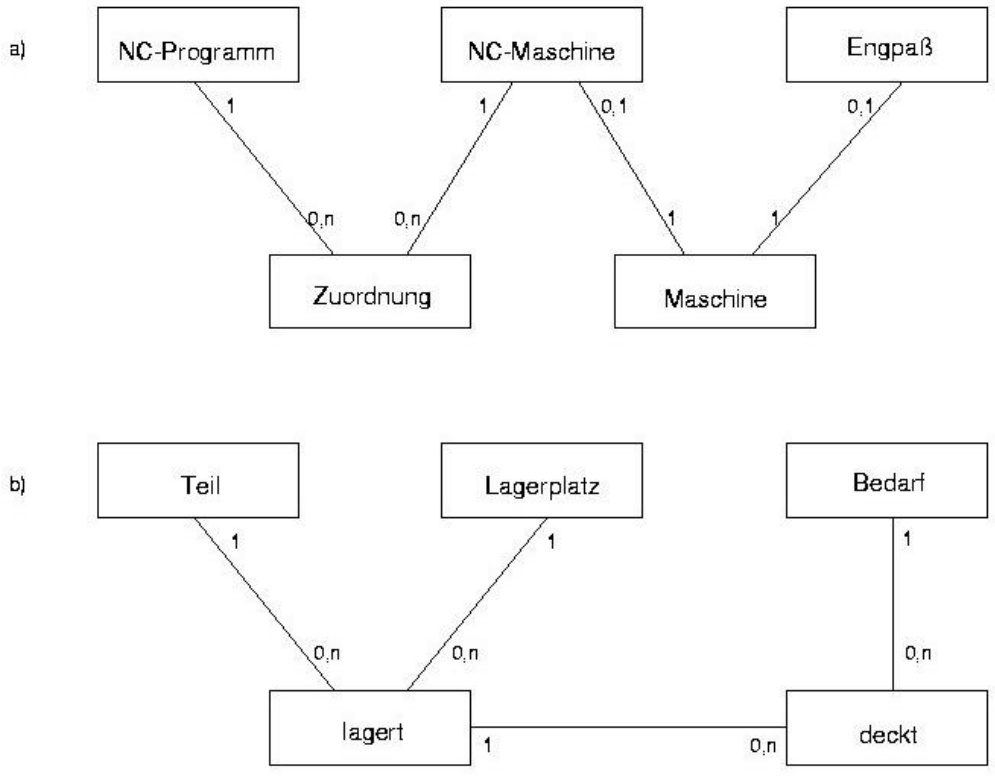


Abb. 22: Beispiele für Entitäten-Diagramme

Weiterhin wird das Konzept von globalen und lokalen Attributen eingeführt, um die Redundanzen durch die funktionalen Abhängigkeiten der Attribute unterschiedlicher

Entitätsmengen, die im Normalisierungsprozeß (Codd 70; Date 81; Kent 83) nicht berücksichtigt werden, besser kontrollieren zu können. Da dieser Aspekt bereits Gegenstand der Implementierungsebene ist, soll er an dieser Stelle nicht weiter verfolgt werden.

Abbildung 21 zeigt die im Entitäten-Diagramm (ED) verwendeten Symbole. Abbildung 22 zeigt die Beispiele aus Abbildung 19. Für die Beziehung zwischen den Entitätsmengen *NC-Programm* und *NC-Maschine* ist eine neue Entitätsmenge *Zuordnung* eingeführt worden, die aber existentiell von den beiden anderen Entitätsmengen abhängig ist. Das gleiche gilt für die Mengen *lagert* und *deckt*, wobei *lagert* als eine aus einer Beziehung resultierende Entitätsmenge selbst wieder existentielle Voraussetzung für die Menge *deckt* ist. Die Attribute *Kapaz* und *Menge* werden auf dieser Ebene nicht dargestellt.

4.7 Semantic-Association-Modell

Das Semantic-Association-Modell (**SAM***) geht zurück auf Arbeiten von Su und Lo (Su/Lo 80). Das Modell stellt verschiedene Modellierungskonstrukte und ein semantisches Netzdiagramm zur Verfügung, um Anforderungen wie komplexe Datentypen, temporale Beziehungen, rekursive Objektdefinitionen, komplexe Datenobjekte, Generalisierung, Attributvererbungen, Partitionierung und Replikationen sowie Versionsverarbeitung darstellen zu können (Su 85).

In SAM* werden zwei Typen von Begriffen, atomare und nicht-atomare, unterschieden. Atomare Begriffe sind nicht weiter zerlegbare Informationselemente wie z. B. eine Teilenummer oder eine Bezeichnung, aber auch komplexere Gebilde wie Vektoren für die Koordinatenbestimmung. Nicht-atomare Begriffe sind aus atomaren oder nicht-atomaren Begriffen zusammengesetzte Objekte, die als Associations bezeichnet werden. Abhängig von den abzubildenden Struktureigenschaften und semantischen Bedingungen können sieben Typen von Associations unterschieden werden, die als Modellierungskonstrukte die Knoten in einem Semantischen Netz repräsentieren.

Associations werden als Kreise dargestellt, die einen Buchstaben als Kennzeichnung des Typs enthalten und durch einen Namen bezeichnet werden. Diese Kreise oder Knoten werden über gerichtete Kanten, die durch Komplexitäts- oder Typangaben ergänzt sind, verbunden. Abbildung 23 zeigt die Symbole des Semantic-Association-Modells.

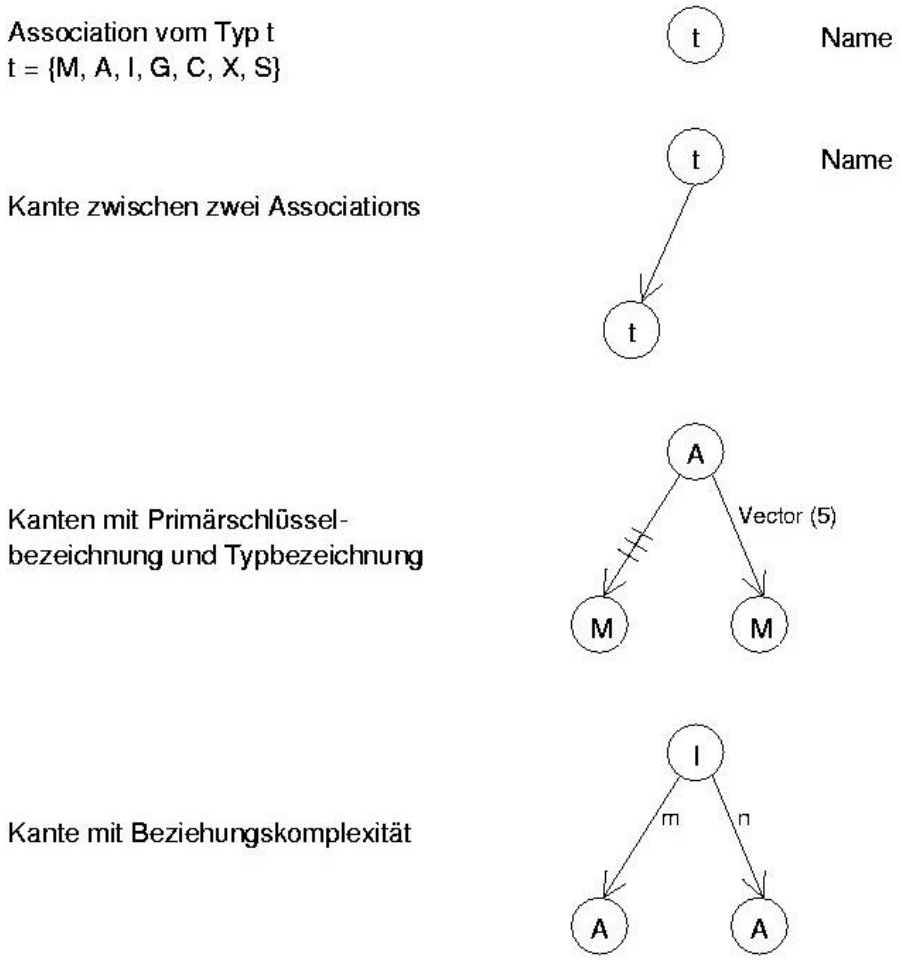


Abb. 23: Symbole des SAM*

Die **Membership Association** (Typ M) ist eine Gruppierung von atomaren Begriffen des gleichen Datentyps und bilden damit eine Domäne. Neben einfachen Datentypen wie Integer, Character, Date, Time oder Dollar können komplexe Datentypen wie Vektor, Matrix, String, Zeitreihe, Liste, geordnete Liste oder Duplikatliste und Anweisungen wie Compute oder Rule auftreten.

Unter einer **Aggregation Association** (Typ A) versteht man einen Begriff, der durch eine Menge von Attributen und Charakteristiken beschrieben wird. Damit entspricht eine Aggregation Association den Entitytypen des ERM. Abbildung 24 zeigt die Aggregation des Typs *Teil*, die durch eine *TId* als Schlüssel, eine *Bezeichnung* und eine *Größe*nangabe als Vektor beschrieben wird.

Beziehungen zwischen Aggregation Associations können durch **Interaction Aggregation** (Typ I) ausgedrückt werden, die durch eine Komplexitätsangabe ergänzt werden. In Abbildung 24 sind die beiden Typen *Teile* und *Kunde* über die Interaction Association mit dem Komplexitätsgrad (n:m) verknüpft.

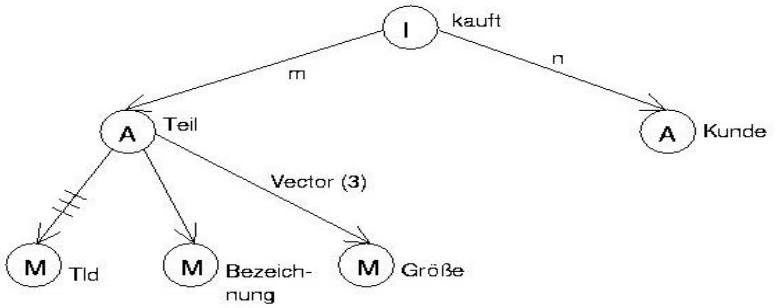


Abb. 24: Aggregation und Interaction Association im SAM*

Mit der **Generalization Association** (Typ G) können Generalisierungen mit Integritätsbedingung formuliert werden. Diese vier Bedingungen sind die Exklusion von Teilmengen (SX = Set Exclusion), die Gleichheit von Teilmengen (SE = Set Equality), das Überlappen von Teilmengen (SI = Set Intersection) sowie die Untermenge einer Teilmenge zu einer anderen Teilmenge (ST-SS = Set-Subset).

Die **Composition Assoziation** (Type C) ist eine Zusammenfassung beliebiger Associations, bei denen im Gegensatz zu der Generalization Association die beteiligten Associations keine Schlüsselattribute (Member Association) der gleichen Domäne besitzen müssen. Abbildung 25 zeigt am Beispiel einer vertikalen Verteilung eines Datenbestandes die Anwendung der Generalization und Composition Association. Die Arbeitspläne werden aufgeteilt in einen lokalen Bestand, in dem alle *Vorgabezeiten*, und in einen anderen Bestand, in dem alle *Losgröße*n definiert sind. Die beiden Bestände bilden jeweils Teil-Arbeitspläne, deren *APL-Nr* als Primärschlüssel dem gleichen Wertebereich angehören müssen. Beide Teil-Arbeitspläne werden zum Typ *APL* generalisiert, wobei jedes Element aus Teilarbeitsplan 1 auch in der Menge Teilarbeitsplan 2 und umgekehrt vorkommen muß. Teilarbeitsplan 2 bildet zusammen mit den *Maschine*n und *Werkzeug*en sowie deren Beziehung zueinander eine lokale Datenbank.

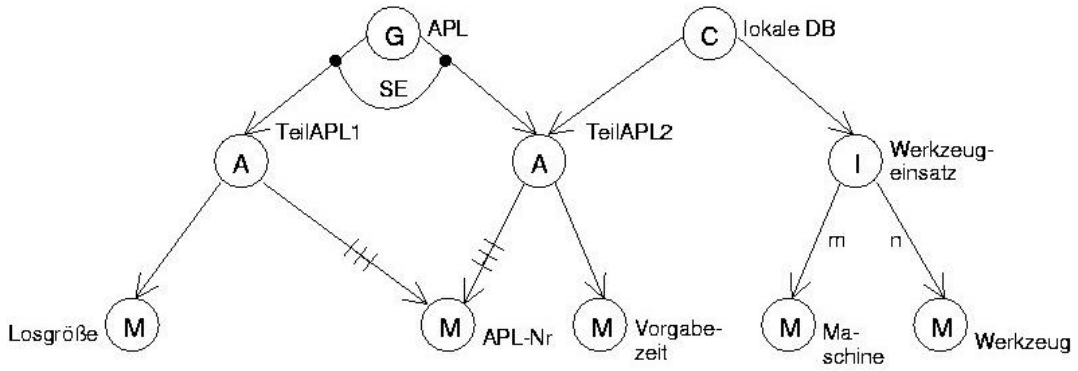


Abb. 25: Generalization und Composition Association in SAM*

Abbildung 26 zeigt die Beispiele aus Abbildung 19. Die Association *Engpaß* als Teilmenge der Association *Maschinen* ist eine Untermenge der Teilmenge *NC-Maschinen*. Beide werden über Maschinennummern identifiziert. Die Interaction Association *lagert-Id* als Beziehung zwischen *Teil* und *Lagerplatz* bildet zusammen mit dem Member Association *Menge* den Typ *lagert*, der seinerseits eine Beziehung mit dem Typ *Bedarf* eingeht.

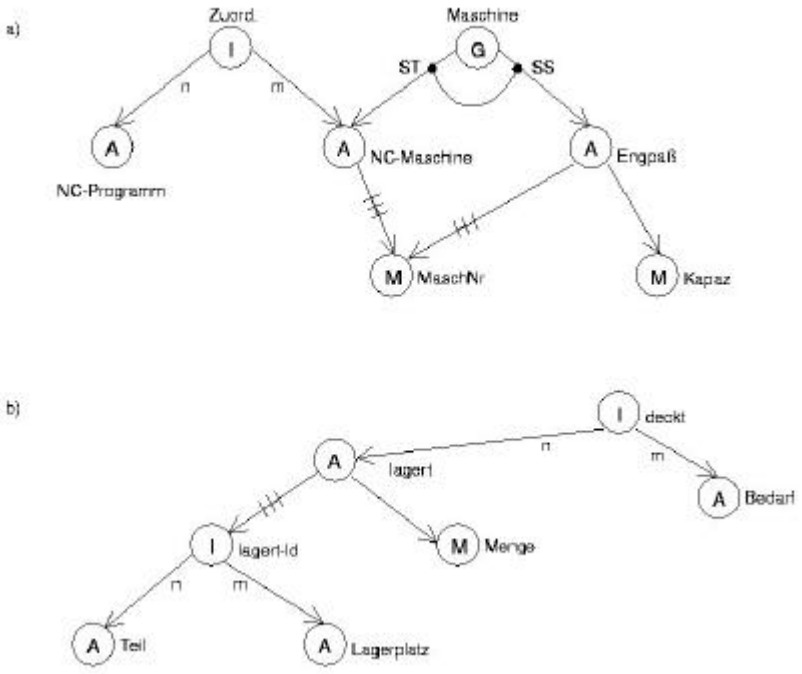


Abb. 26: Beispiel für SAM*-Diagramme

5. Anforderungen an Beschreibungssprachen aus Sicht der Fertigung

Die vorgestellten Beschreibungssprachen sollen anhand von ausgewählten Kriterien untersucht und bewertet werden. Als Kriterien dienen **Konstruktionsoperatoren**, wie sie Ortner als allgemeingültigen Ansatz formuliert (Ortner 85). Abbildung 27 zeigt, wie die Konstruktionsoperatoren abgeleitet sind. Besteht eine äquivalente Beziehung zwischen Objekten, so kann zwischen der Identität, wie der Identifikation eines Objektes über die Ausprägung von Schlüsselattributen, und der Parität unterschieden werden. Werden gleiche Objekte unter einem Begriff subsumiert, so spricht man von der Klassifikation. Besteht die Gleichheit von Objekten dagegen in der Unterordnung von einem Objekt unter ein Oberobjekt, so spricht man von Generalisierung. Bei abhängigen Beziehungen können interdependente und inhärente Beziehungen unterschieden werden. Interdependenzen können zwischen verschiedenen gleichwertigen Objektbegriffen bestehen und werden als Aggregation bezeichnet.

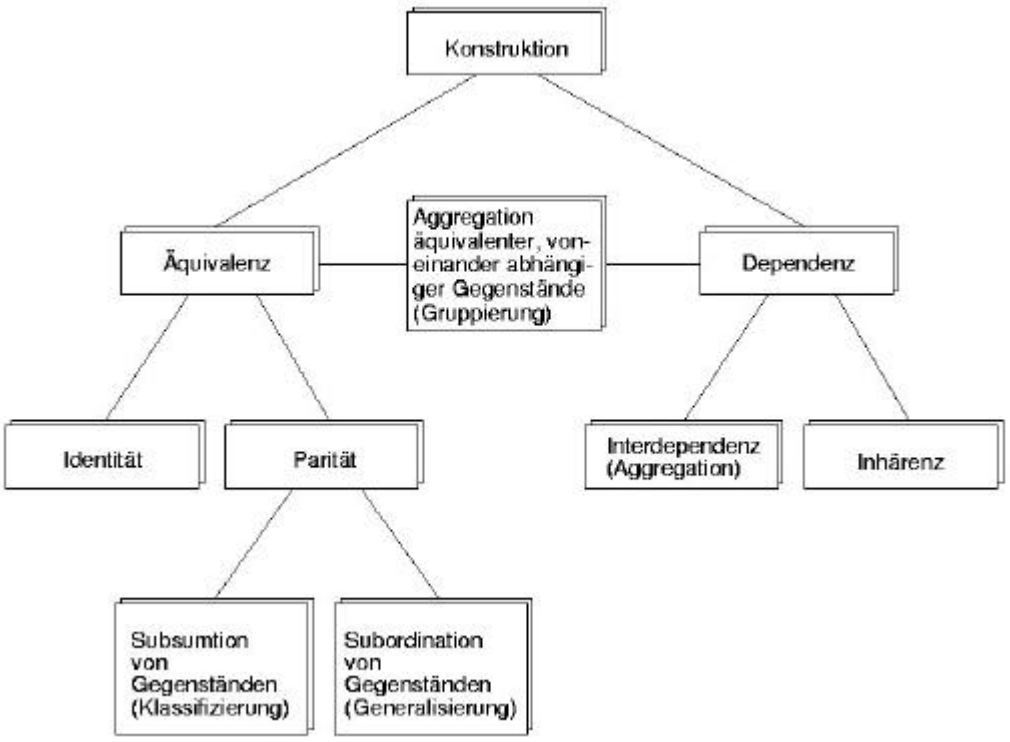


Abb: 27: Ableitung von Konstruktionsoperationen (in Anlehnung an: Ortner 85)

Inhärente Beziehungen bestehen innerhalb eines Begriffes, wie beispielsweise die Zugehörigkeit eines Attributes. Beziehungen, die gleiche, untereinander abhängige Objekte aggregieren, sollen als Gruppierung bezeichnet werden.

Ferner sollen zusätzliche Aspekte als Kriterium herangezogen werden, die die **semantische Ausdruckskraft** der Beschreibungssprache vergrößern und für die Abbildung der im Fertigungsbereich auftretenden Strukturen und Problemstellungen notwendig sind. Dies sind die Modellierung alternativer Beziehungen, die Modellierungsmöglichkeiten von Versionsverwaltung, die Zusammenfassung von Strukturteilbereichen zu Cluster sowie die Modellierung zusätzlicher Integritätsbedingungen. In der Literatur finden sich weitere Kriterien und Modellvergleiche (u. a. bei Brodie 83; Chen 83; Brodie 84; Hagelstein/Rifaut 88).

Im folgenden werden die Kriterien im einzelnen diskutiert, die für die Modellierung von fertigungsrelevanten Datenstrukturen notwendig sind, um anschließend die vorgestellten Modelle beurteilen zu können. Die Notwendigkeit wird durch Beispiele aus dem Anwendungsbereich dokumentiert. Die bei den Erläuterungen verwendete Darstellung der Kriterien lehnt sich an das Entity-Relationship-Modell und dessen Erweiterungen an.

5.1 Klassifizierung

Werden Objekte zu einem Objekttyp zusammengefaßt, so wird dies als Klassifizierung bezeichnet. Ortner verwendet auch den Begriff Prädikation (Ortner 85). Die einzelnen Objekte müssen bezüglich des Objekttyps gleich sein. Dies wird gewährleistet, wenn sich die Objekte durch gleiche Attribute beschreiben lassen. Beispielsweise lassen sich alle Bearbeitungsmaschinen der Fertigung zu dem Typ Maschine zusammenfassen, der durch Attribute wie Bezeichnung, Kostensatz, Leistungsgrad oder Anschaffungsdatum beschreibbar ist.

5.2 Aggregation

Aggregationen (Smith/Smith 77a) oder Konnexionen (Ortner 85) als Beziehungen zwischen Objekten lassen sich durch die Eigenschaften der

- Anzahl der Objekte, sowie der

- Anzahl der Objekttypen, die an der Aggregation beteiligt sind, und der

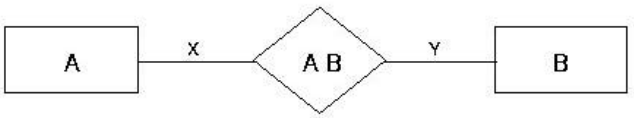
- Komplexität der Aggregation bezüglich der beteiligten Objekte charakterisieren.

An einer Aggregation müssen mindestens zwei Objekte beteiligt sein. Da ein Objekt auch eine Beziehung mit einem Objekt des gleichen Typs eingehen kann, ist die Untergrenze für die Anzahl der Objekttypen gleich eins. Zur Unterscheidung der beteiligten Objekte können Rollennamen vergeben werden. Die Anzahl der beteiligten Objekte soll als Grad der Beziehung bezeichnet werden (im Gegensatz zu (Schlageter/Stucky 83), bei denen die Anzahl der Objekttypen den Grad der Beziehung bestimmt). In der angelsächsischen Literatur sind die Begriffe uniary relationship für die Beziehung zwischen zwei Objekten eines Objekttyps, binary relationship für die Beziehung zwischen zwei Objekten unterschiedlicher Objekttypen und tenary relationship für Beziehungen zwischen drei Objekten unterschiedlicher Objekttypen eingeführt (z. B. Teory et al. 86). Es sei noch einmal darauf hingewiesen, daß Beziehungen gleich welchen Grades nur bestehen können, wenn jede Rolle von einem vorhandenen Objekt besetzt ist, d. h. eine Beziehung ist von einer Anzahl Objekte existentiell abhängig, die dem Grad der Beziehung entspricht (Wedekind 85).

Im folgenden soll untersucht werden, welche Arten von Aggregationen abhängig vom Grad der Beziehung unterschieden werden können. Die Unterscheidung der verschiedenen Aggregationsarten wird mit Hilfe der Definition der Komplexitätsgrade vorgenommen, wie sie bereits in dem Kapitel "Allgemeine Erweiterungen des Entity-Relationship-Modells" (s. S. 20) vorgestellt worden ist.

Binäre Aggregation

Bei binären Aggregationen zwischen **zwei Objekttypen** können, wenn bei dem Komplexitätsgrad zwischen einem (min)-Wert von null oder eins sowie einem (max)-Wert von eins oder "n" differenziert wird, 10 Arten von Beziehungstypen unterschieden werden. Sie werden in Abbildung 28 dargestellt. Diese Darstellungsmöglichkeiten entsprechen der 1,c,m-Notation.



	X	Y
(1)	0,1	0,1
(2)	0,1	1,1
(3)	1,1	1,1
(4)	0,n	0,1
(5)	0,n	1,1
(6)	1,n	0,1
(7)	1,n	1,1
(8)	0,n	0,n
(9)	1,n	0,n
(10)	1,n	1,n

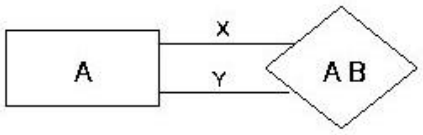
Abb 28: Typen binärer Aggregationen zwischen zwei Objekttypen

Rekusive, binäre Beziehungen innerhalb **eines Objekttyps** treten dann auf, wenn ein Objekt Beziehungen zu anderen Objekten eingehen kann, die der gleichen Klasse angehören. Dies läßt sich am Beispiel von Stücklistenstrukturen verdeutlichen. Die Stücklisten stellen eine Beziehung zwischen zwei Objekten des Typs *Teil* dar, wobei ein Teil die Rolle des übergeordneten Teils (Oberteil) und ein Teil die Rolle des untergeordneten Teils (Unterteil) übernimmt. Dieser Sachverhalt wird in allgemeiner Form in Abbildung 29 dargestellt, wobei der Objekttyp A das Teil repräsentiert, und die Aggregation AB die Stücklistenstruktur wiedergibt.

Die Komplexitätsgrade der Kanten *x* (Oberteil) und *y* (Unterteil) sind wie folgt zu interpretieren:

(11) Jedes Teil kann maximal einmal als Oberteil und maximal einmal als Unterteil auftreten.

(12) Jedes Teil ist genau einmal Unterteil und einmal Oberteil. Bei dieser Darstellung bildet jede Stückliste einen geschlossenen Ring.



	X	Y
(11)	(0,1)	(0,1)
(12)	(1,1)	(1,1)
(13)	(0,1)	(0,n)
(14)	(1,1)	(0,n)
(15)	(0,n)	(0,n)
(16)	(0,n)	(1,n)
(17)	(1,n)	(1,n)

Abb. 29: Typen rekursiver, binärer Aggregationen innerhalb eines Objekttyps

(13) Jedes Teil kann maximal einmal als Unterteil auftreten, d. h. es hat höchstens ein übergeordnetes Teil, und jedes Teil kann mehrmals als Oberteil auftreten, d. h. es kann mehrere untergeordnete Teile besitzen. Mit diesen Komplexitätsgraden können hierarchische Strukturen abgebildet werden.

(14) Jedes Teil tritt genau einmal als Unterteil auf, d. h. jedes Teil hat immer genau einen Vater. Weiterhin kann jedes Teil mehrmals als Oberteil auftreten.

(15) Jedes Teil kann mehrmals als Unterteil und mehrmals als Oberteil auftreten. Diese Komplexitätsgrade sind der Normalfall für Teilestücklisten, die als Gozintograph abgebildet werden.

(16) Jedes Teil kann mehrmals als Unterteil und muß mindestens einmal als Oberteil auftreten.

(17) Jedes Teil muß mindestens einmal als Unterteil und einmal als Oberteil auftreten.

Abbildung 30 zeigt die beschriebenen Möglichkeiten auf Objektebene. Eine Beziehung mit den Komplexitätsgraden (1,1) und (0,1) würde bsw. bedeuten, daß jedes Objekt genau einen Vater, verschiedene Objekte aber keinen Sohn besitzen. Dies führt aber bei einer endlichen Anzahl von Objekten dazu, daß jedes Objekt genau einen Sohn haben muß und ist deshalb mit dem Fall (12) identisch. Beziehungen mit den Komplexitätsgraden (0,1) und (1,n) sowie mit (1,1) und (1,n) sind nicht abbildbar, da auch hier bei einer endlich Anzahl von Objekten nicht jedes Objekt ein oder mehrere Söhne besitzen kann.

Es ist festzuhalten, daß rekursive, binäre Aggregationen mit einer Kante, die eine Untergrenze von eins aufweist, immer **Rekursiven auf der Objektebene** in beliebiger Tiefe aufweisen.

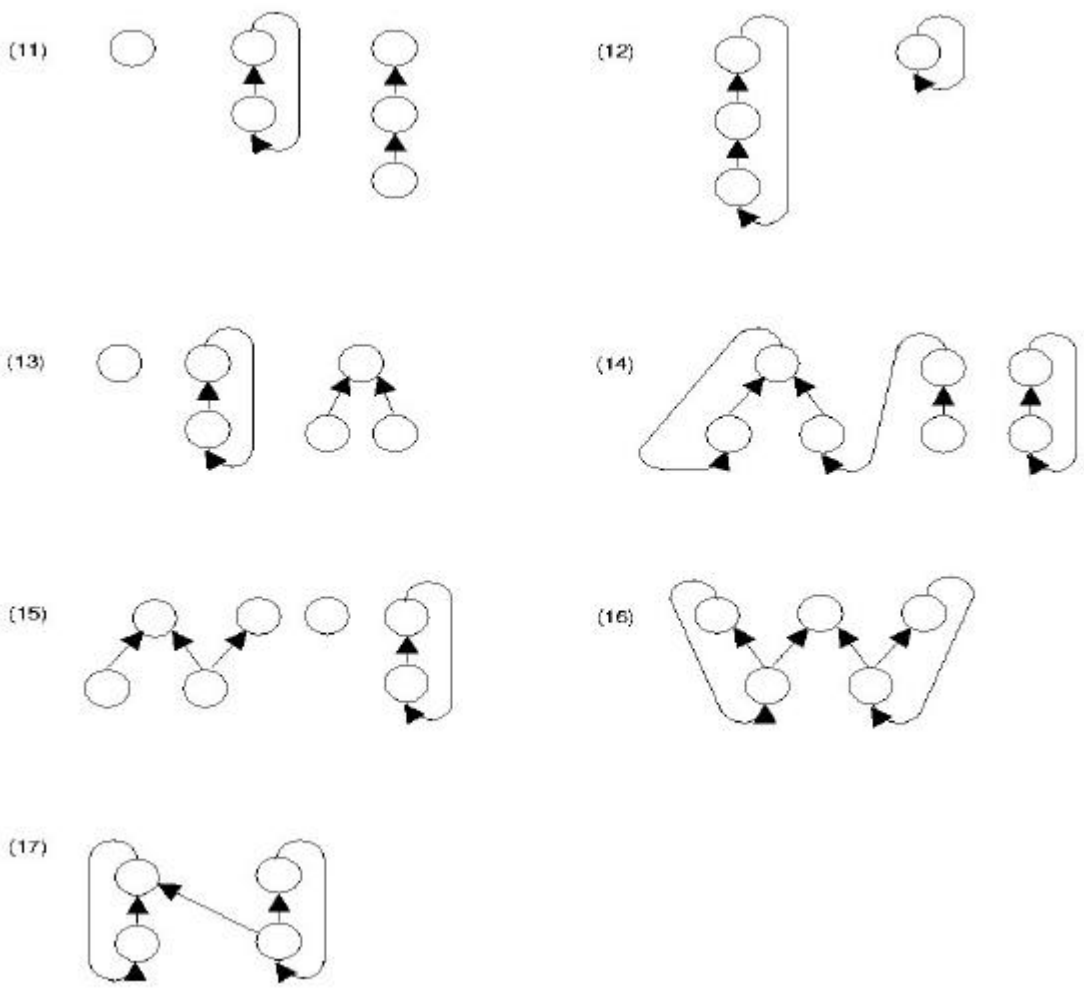


Abb. 30: Binäre, rekursive Beziehungen auf Objektebene

Mehrfachaggregation

Bei Mehrfachaggregationen ist die Notation der Komplexitätsgrade nicht hinreichend genau. Dies soll an einem Beispiel mit einer Dreierbeziehung erläutert werden:

Die Aggregation *arbeitet* sei eine Beziehung zwischen den Objekttypen *Mitarbeiter*, *Abteilung* und *Projekt*. Ein Mitarbeiter kann in mehreren Abteilungen an unterschiedlichen Projekten arbeiten. Andererseits kann ein Mitarbeiter an einem Projekt in unterschiedlichen Abteilungen arbeiten. In einer Abteilung arbeiten mehrere Mitarbeiter an unterschiedlichen Projekten. Ein Projekt kann in mehreren Abteilungen von verschiedenen Mitarbeitern bearbeitet werden. Abbildung 31 zeigt den Sachverhalt als ERM mit der (min,max)-Notation für die Komplexitätsdarstellung.

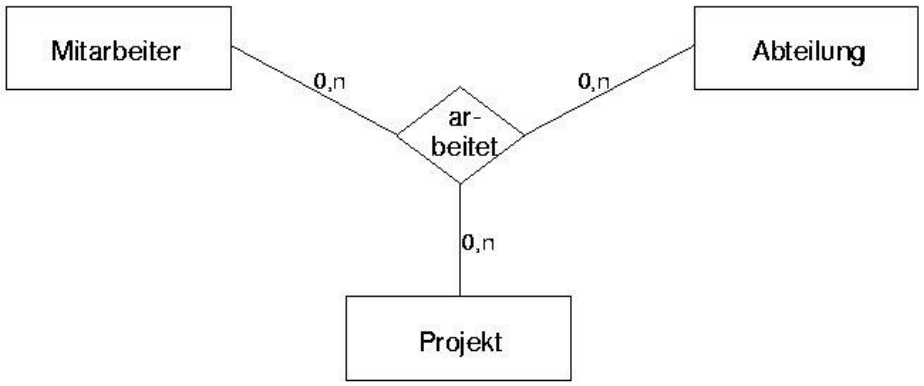


Abb. 31: Komplexitätsangaben bei Mehrfachaggregation

Keine Objekte der Typen *Mitarbeiter*, *Abteilung* und *Projekt* müssen zwingend eine Beziehung eingehen, deshalb haben alle Objekte eine Untergrenze von null. Andererseits kann jedes einzelne Objekt aller drei Typen mehrmals eine Beziehung eingehen, so daß jeweils ein "n" als Obergrenze anzugeben ist.

Das Beispiel wird dahingehend geändert, daß ein Mitarbeiter zwar in mehreren Abteilungen, aber pro Abteilung nur in genau einem Projekt arbeiten kann. Die übrigen Aussagen bleiben bestehen. Für die Bestimmung der Komplexitätsangaben gilt, daß ein Mitarbeiter in unterschiedlichen Kombinationen von Abteilungen und Projekten arbeiten kann. Die Aussage trifft analog auch für die beiden anderen Objekttypen zu. Es zeigt sich, daß unterschiedliche Ausgangssituationen von Dreierbeziehungen zu gleichen Komplexitätsangaben führen, d. h. die Angaben sind nicht eindeutig.

Die unterschiedlichen Möglichkeiten einer Beziehung von Grad drei sollen genauer mit Hilfe der funktionalen Abhängigkeiten (Date 81; Vetter 89) und einer Auflösung der Beziehung in mehrere Beziehungen von Grad zwei mit eindeutiger (min,max)-Notation untersucht werden.

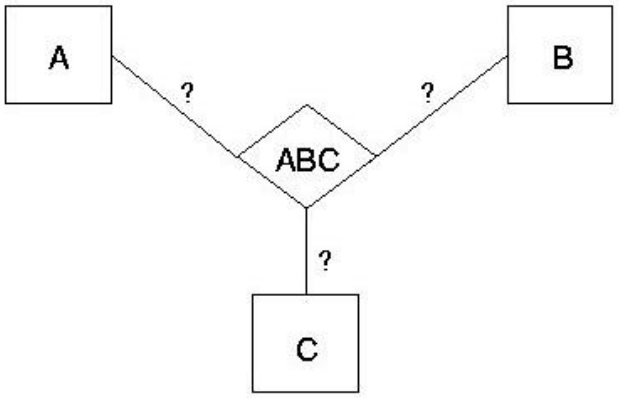


Abb. 32: Ausgangssituation einer Dreifachaggregation

Ausgangssituation ist die Beziehung ABC zwischen den drei Objekttypen A, B und C aus Abbildung 32. Weiterhin soll gelten:

- a sei Primärschlüssel für A, b sei Primärschlüssel für B, und c sei Primärschlüssel für C,

- dom(a) sei der Wertebereich oder die Domäne von a und damit die maximale Anzahl a_i in A; analog für dom(b) und dom(c),

- $a \rightarrow b$ bedeutet, daß b von a funktional abhängig ist.

(1) Zwischen den Attributen a, b und c in der Aggregation existiert keine funktionale Abhängigkeit. Abbildung 33 (1) a zeigt die Dreierbeziehung mit (min,max)-Notation, Teil b der Abbildung eine mögliche Auflösung in binäre Beziehungen. AB muß eine (1,n)-Beziehung zu AB-C eingehen. Es gilt:

$a, b, c \rightarrow \varnothing$
$dom(ABC) = dom(a) \times dom(b) \times dom(c)$

Beispiel:
Dies entspricht der Ausgangssituation des Beispiels aus Abbildung 31 mit A = *Mitarbeiter*, B = *Abteilung* und C = *Projekt*.

(2) Zwischen den Attributen der Aggregation besteht eine funktionale Abhängigkeit, so daß ein Attribut von einer Kombination der beiden anderen Attribute abhängig ist. In Teil b der Abbildung 33 (2) geht dementsprechend eine AB-Beziehung ihrerseits genau eine Beziehung mit C ein, während in Teil a der Abbildung sich gegenüber Fall (1) nichts geändert hat. Es gilt:

$a, b \rightarrow c$
$dom(ABC) = dom(a) \times dom(b)$

Beispiel:
Dies entspricht dem modifizierten Beispiel aus Abbildung 31.

(3) Zu der einen Abhängigkeit in (2) kommt eine zweite Abhängigkeit, d. h. je ein Attribut ist von der Kombination der restlichen zwei Attribute abhängig. In der (min,max)-Notation hat sich in der Dreierbeziehung nichts geändert (Abbildung 33 (3) a). In der binären Darstellung gehen jeweils die beiden Objekttypen der bestimmenden Attribute der funktionalen Abhängigkeiten eine Beziehung ein, die ihrerseits über eine Beziehung mit der Komplexität (1,1) verbunden sind, d. h. sie bedingen sich gegenseitig. Es gilt:

$a, b \rightarrow c$
$a, c \rightarrow b$
$dom(ABC) = min((dom(a) \times dom(b)), (dom(a) \times dom(c)))$

Beispiel:
Ein Mitarbeiter kann in einer Abteilung nur in einem Projekt und an einem Projekt nur in einer Abteilung arbeiten.

(4) Es existieren insgesamt drei funktionale Abhängigkeiten. Ein Attribut ist jeweils von den beiden anderen Attributen abhängig. Auch in dieser Situation bleibt die (min,max)-Notation unverändert. Bei der Auflösung in Zweierbeziehungen gehen die

Objekttypen paarweise binäre Beziehungen ein, die ihrerseits jeweils zu dem dritten Objekttyp eine Beziehung eingehen (Abbildung 33(4) b). Bezüglich der Komplexitätsangaben hat sich nichts gegenüber der Ausgangssituation geändert, da jedes Objekt aller drei Typen mehrere Beziehungen eingehen kann. In der binären Darstellung geht jede Zweierbeziehung zwischen A, B und C mit den anderen Zweierbeziehungen eine Beziehung mit der Komplexität von (1,1) ein, d. h. sie setzen sich jeweils gegenseitig voraus. Es gilt:

$a, b \rightarrow c$
$a, c \rightarrow b$
$b, c \rightarrow a$
$dom(ABC) = \min(dom(a) \times dom(b), dom(a) \times dom(c), dom(b) \times dom(c))$

Beispiel:
Pro Abteilung kann nur ein Mitarbeiter an einem bestimmten Projekt arbeiten, an verschiedenen Projekten aber unterschiedliche Mitarbeiter. Ein Mitarbeiter kann in einer Abteilung nur in einem Projekt und an einem Projekt nur in einer Abteilung arbeiten. Ein Projekt kann in mehreren Abteilungen bearbeitet werden, aber in jeder Abteilung nur von einem Mitarbeiter.

(5) Innerhalb der Aggregation sind zwei Attribute funktional von dem dritten Attribut abhängig. Dies bedeutet, daß die Aggregation von der Ausprägung eines einzigen Objekttyps bestimmt wird (Abbildung 33 (5) a). Jedes Objekt des Typs A kann nur in einer einzigen Beziehung ABC existieren und erhält damit die Komplexitätsangabe (0,1). In der binären Darstellung geht deshalb der Objekttyp A eine Beziehung A-BC mit dem Komplexitätsgrad (0,1) zu der Aggregation BC ein, während der Komplexitätsgrad aus Sicht der Aggregation (1,n) ist. Es gilt:

$a \rightarrow b, c$
$dom(ABC) = dom(a)$

Beispiel:
Wenn ein Mitarbeitet arbeitet, dann genau in einer Abteilung in einem Projekt. In einer Abteilung können an einem Projekt mehrere Mitarbeiter arbeiten, auch kann in einer Abteilung an mehreren Projekten gearbeitet werden. Ein Projekt kann in mehreren Abteilungen bearbeitet werden.

(6) Die Aggregation weist zwei funktionale Abhängigkeiten auf, wobei in beiden Fällen jeweils zwei Attribute von dem dritten Attribut abhängig sind. Die beiden bestimmenden Objekttypen besitzen jeweils einen Komplexitätsgrad von (0,1) (Abbildung 33 (6) a). In der binären Darstellung werden alle Objekttypen jeweils untereinander verbunden, wobei die Komplexitätsgrade aus Sicht der die funktionalen Abhängigkeiten bestimmenden Objekttypen (A und B) jeweils (0,1) beträgt. Die Beziehung zwischen diesen beiden Objekttypen (AB) bedarf notwendigerweise der Existenz der anderen Beziehungen, was zu den neuen Beziehungen AB-AC und AB-BC mit den Komplexitätsgraden (1,1) führt. Es gilt:

$a \rightarrow b, c$
$b \rightarrow a, c$
$dom(ABC) = min(dom(a), dom(b))$

Beispiel:
Im Gegensatz zu dem Beispiel zu Typ (5) kann in einer Abteilung nur maximal ein Mitarbeiter an nur einem Projekt arbeiten.

(7) Innerhalb der Aggregation existieren drei funktionale Abhängigkeiten, bei denen jeweils zwei Attribute von dem dritten Attribut abhängig sind. Dies führt zu einem Komplexitätsgrad von (0,1) für jeden Objekttyp. In der binären Darstellung wird jeder Objekttyp mit jedem anderen über Beziehungen mit dem Komplexitätsgrad (0,1) verbunden. Diese Beziehungen bedingen sich jeweils gegenseitig, so daß diese wiederum über Beziehungen mit dem Komplexitätsgrad von (1,1) verbunden sind. (Obwohl eine der drei Beziehungen AB-AC, AB-BC und AC-BC aufgrund der gegenseitig existierenden Abhängigkeiten redundant ist, sollen hier der Vollständigkeit wegen alle aufgeführt werden). Es gilt:

$a \rightarrow b, c$
$b \rightarrow a, c$
$c \rightarrow a, b$
$dom(ABC) = min(dom(a), dom(b), dom(c))$

Beispiel:
Ein Projekt kann maximal nur in einer Abteilung und von einem Mitarbeiter bearbeitet werden. In einer Abteilung kann maximal nur ein Mitarbeiter arbeiten an

genau einem Projekt. Ein Mitarbeiter kann maximal nur an einem Projekt arbeiten in genau einer Abteilung.

(8) Die Aggregation besitzt zwei funktionale Abhängigkeiten. Zum einen ist ein Attribut von der Kombination der beiden anderen Attribute abhängig, zum anderen gilt die Abhängigkeit auch umgekehrt. Damit stellt der Typ (8) eine Kombination der Typen (2) und (5) dar. Der Komplexitätsgrad ist identisch mit dem des Typs (5), d. h. das Attribut des Typs, das die erste Abhängigkeit bestimmt (A), erhält die Angabe (0,1), die anderen (0,n). Die binäre Darstellung ist der des Typs (5) ähnlich. Damit geht der die erste Abhängigkeit bestimmende Objekttyp (A) eine Beziehung mit der Beziehung zwischen den anderen Objekttypen ein (BC). Aus Sicht der Beziehung ist der Komplexitätsgrad allerdings (1,1), da aufgrund der zweiten Abhängigkeit auch die Attribute der beiden anderen Objekttypen bestimmend sind. Es gilt:

$a \rightarrow bc$
$b, c \rightarrow a$
$dom(ABC) = \min(dom(a), (dom(b) \times dom(c)))$

Beispiel:
Ein Mitarbeiter kann maximal nur in einer Abteilung und auch nur an einem Projekt arbeiten. Andererseits kann ein Projekt aber in unterschiedlichen Abteilungen bearbeitet werden, und eine Abteilung kann mehrere Projekte bearbeiten. In einer Abteilung kann an einem Projekt jedoch nur ein Mitarbeiter arbeiten.

Neben den gezeigten Typen von Dreifachbeziehungen gibt es noch die Möglichkeit, daß zu den aufgeführten Abhängigkeiten zusätzlich noch solche zwischen genau zwei von drei Attributen kommen (Springsteel/Chuang 88). Abbildung 34 zeigt den Fall (2) aus Abbildung 33 mit einer zusätzlichen funktionalen Abhängigkeit **zweier** Attribute. Es gilt:

$a, b \rightarrow c$
$a \rightarrow c$
$dom(ABC) = dom(a) \times dom(b)$

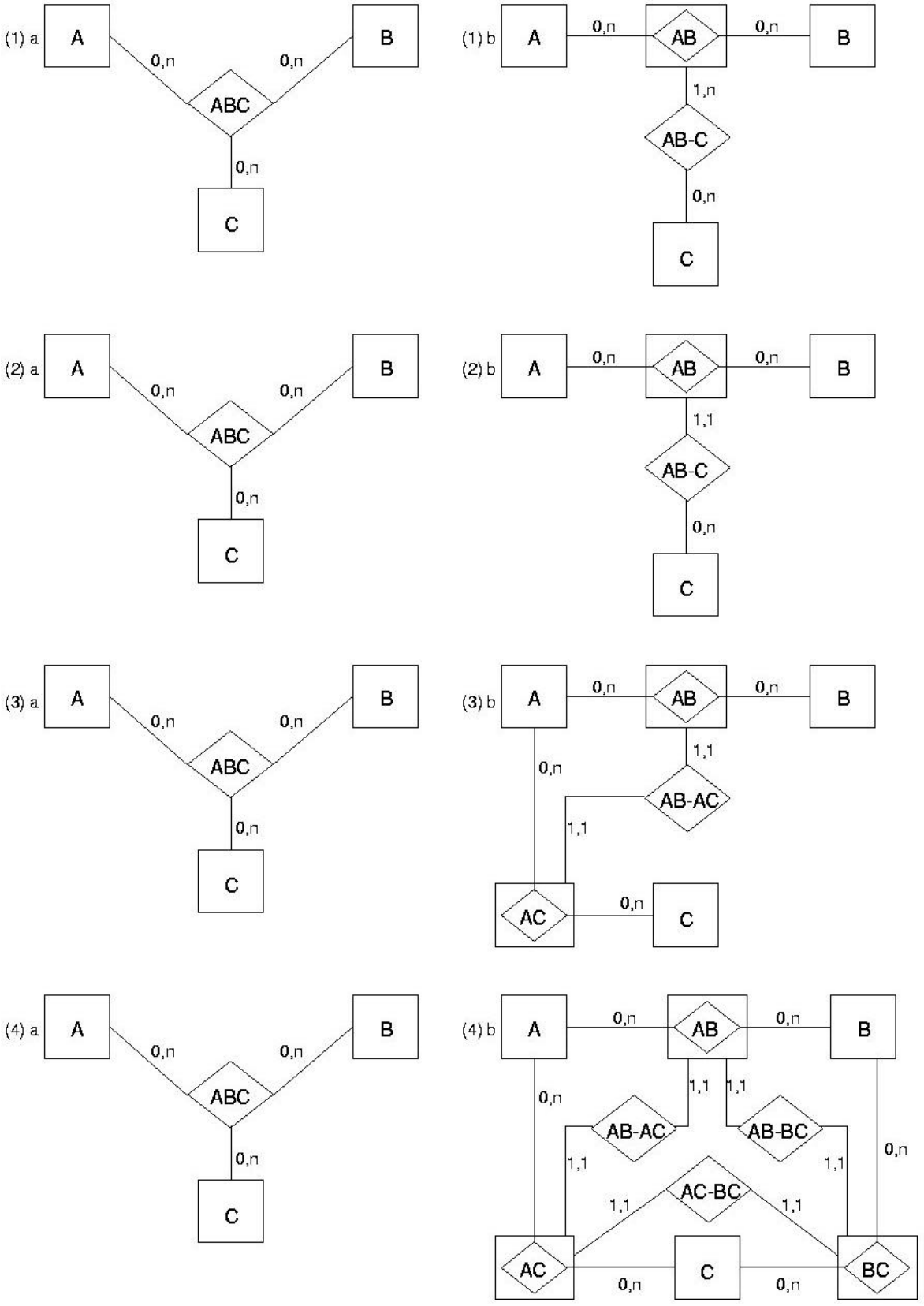


Abb. 33-1: Typen von Dreifachaggregationen (Typ 1 bis 4)

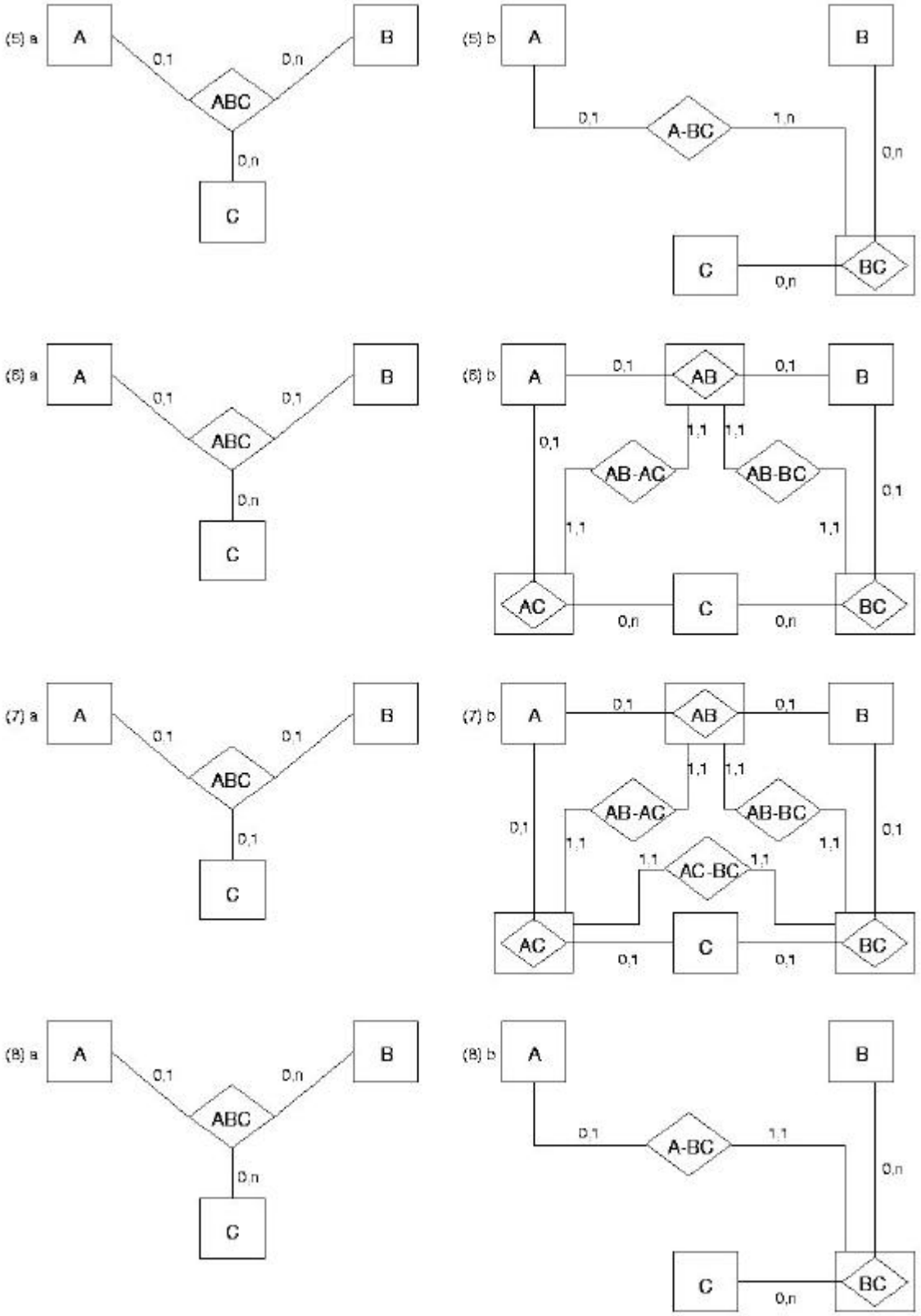


Abb. 33-2: Typen von Dreifachaggregationen (Typ 5 bis 8)

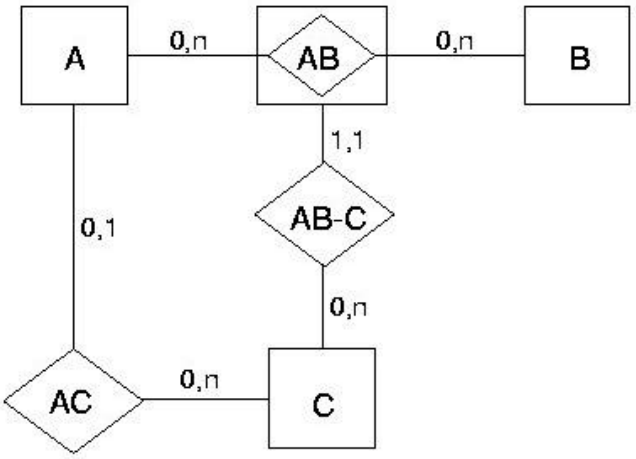


Abb. 34: Dreifachaggregation mit binären Abhängigkeiten

Lenzerini und Santucci geben als Komplexitätsnotation für solche Fälle für alle Abhängigkeiten eine Minimum- und Maximum-Bedingung an (Lenzerini/Santucci 83). Für das gezeigte Beispiel gilt:

$\max[ABC(a,b/c)] = 1 \wedge \max[ABC(a/c)] = 1$

Beziehungen solchen Typs sollen als unechte Dreifachbeziehungen bezeichnet werden, da

- neben Abhängigkeiten zwischen allen beteiligten Entities auch Abhängigkeiten unter Ausschluß eines Entities und damit binäre Beziehungen definiert werden und

- die der Aggregation entsprechende Relation nicht der zweiten Normalform (Codd 70; Date 81; Kent 83) genügt, da ein Nichtschlüsselattribut (c) funktional von einem Schlüsselteil und nicht vom Gesamtschlüssel abhängig ist.

Die gezeigte Nichteindeutigkeit der (min,max)-Notation für Beziehungen von Grad drei gilt auch für Beziehungen von Grad größer drei. So gibt es allein für Beziehungen von Grad vier 24 verschiedene Möglichkeiten funktionaler Abhängigkeiten (vgl. Abbildung 35).

(1) $a,b,c,d \rightarrow \varnothing$

(2) $a,b,c \rightarrow d$

(3) $a,b,c \rightarrow d \ \wedge \ a,b,d \rightarrow c$

(4) $a,b,c \rightarrow d \ \wedge \ a,b,d \rightarrow c \ \wedge \ a,c,d \rightarrow b$

(5) $a,b,c \rightarrow d \ \wedge \ a,b,d \rightarrow c \ \wedge \ a,c,d \rightarrow b \ \wedge \ b,c,d \rightarrow a$

(6) $a,b \rightarrow c,d$

(7) $a,b \rightarrow c,d \ \wedge \ c,d \rightarrow a,b$

(8) $a,b \rightarrow c,d \ \wedge \ a,c \rightarrow b,d$

(9) $a,b \rightarrow c,d \ \wedge \ c,d \rightarrow a,b \ \wedge \ a,c \rightarrow b,d$

(10) $a,b \rightarrow c,d \ \wedge \ a,c \rightarrow b,d \ \wedge \ b,c \rightarrow a,d$

(11) $a,b \rightarrow c,d \ \wedge \ c,d \rightarrow a,b \ \wedge \ a,c \rightarrow b,d \ \wedge \ b,d \rightarrow a,c$

(12) $a \rightarrow b,c,d$

(13) $a \rightarrow b,c,d \ \wedge \ b \rightarrow a,c,d$

(14) $a \rightarrow b,c,d \ \wedge \ b \rightarrow a,c,d \ \wedge \ c \rightarrow a,b,d$

(15) $a \rightarrow b,c,d \ \wedge \ b \rightarrow a,c,d \ \wedge \ c \rightarrow a,b,d \ \wedge \ d \rightarrow a,b,c$

(16) $a \rightarrow b,c,d \ \wedge \ b,c \rightarrow a,d$

(17) $a \rightarrow b,c,d \ \wedge \ b,c \rightarrow a,d \ \wedge \ c,d \rightarrow a,b$

(18) $a \rightarrow b,c,d \ \wedge \ b,c \rightarrow a,d \ \wedge \ c,d \rightarrow a,b \ \wedge \ b,d \rightarrow a,c$

(19) $a \rightarrow b,c,d \ \wedge \ b,c,d \rightarrow a$

(20) $a,b \rightarrow c,d \ \wedge \ b,c,d \rightarrow a$

(21) $a,b \rightarrow c,d \ \wedge \ b,c,d \rightarrow a \ \wedge \ a,c,d \rightarrow c$

(22) $a,b \rightarrow c,d \ \wedge \ a,c \rightarrow b,d \ \wedge \ b,c,d \rightarrow a$

(23) $a,b \rightarrow c,d \ \wedge \ a,c \rightarrow b,d \ \wedge \ a,d \rightarrow b,c \ \wedge \ b,c,d \rightarrow a$

(24) $a \rightarrow b,c,d \ \wedge \ b \rightarrow c,d,a \ \wedge \ c,d \rightarrow a,b$

Abb. 35: Typen von Vierfachaggregationen

5.3 Gruppierung

Eine Gruppierung ist die Zusammenfassung von einzelnen Elementen einer Menge zu neuen Teilbegriffen (Scheer 90f, S. 29). Ortner hat gezeigt, daß zwischen den Elementen einer Gruppierung sowohl partielle Gleichheit besteht als auch eine Abhängigkeit vom gebildeten Gruppenbegriff (Ortner 85). Die Gruppierung läßt sich als Aggregation mit existentieller Abhängigkeit darstellen (binäre Aggregation von Typ (5), s. S. 47). Abbildung 36 a zeigt ein Beispiel, bei dem *Maschine*n zu *Kostenstelle*n gruppiert werden. Dabei sind die Maschinen von genau einer Zuordnung zu einer Kostenstelle abhängig, hingegen kann eine Kostenstelle ohne Maschinen existieren. Dieser Sachverhalt wird teilweise mit einem eigenen Symbol wie in Abbildung 36 b dargestellt (Webre 83, S. 193; Scheer 90f, S. 39). Abbildung 356 c zeigt ein Beispiel, bei dem in wechselseitigen Abhängigkeiten auch die Gruppierung *Maschinengruppe* von dem zu gruppierenden Objekttyp *Maschine* abhängig ist. In Abbildung 36 d ist die Gruppierung als Klassifizierung dargestellt. Dies ist dann möglich, wenn als Attribute auch nicht-skalare Wiederholungsgruppen wie im Semantic-Association-Modell (vgl. Kapitel "Semantic-Association-Modell", S. 40) erlaubt sind.

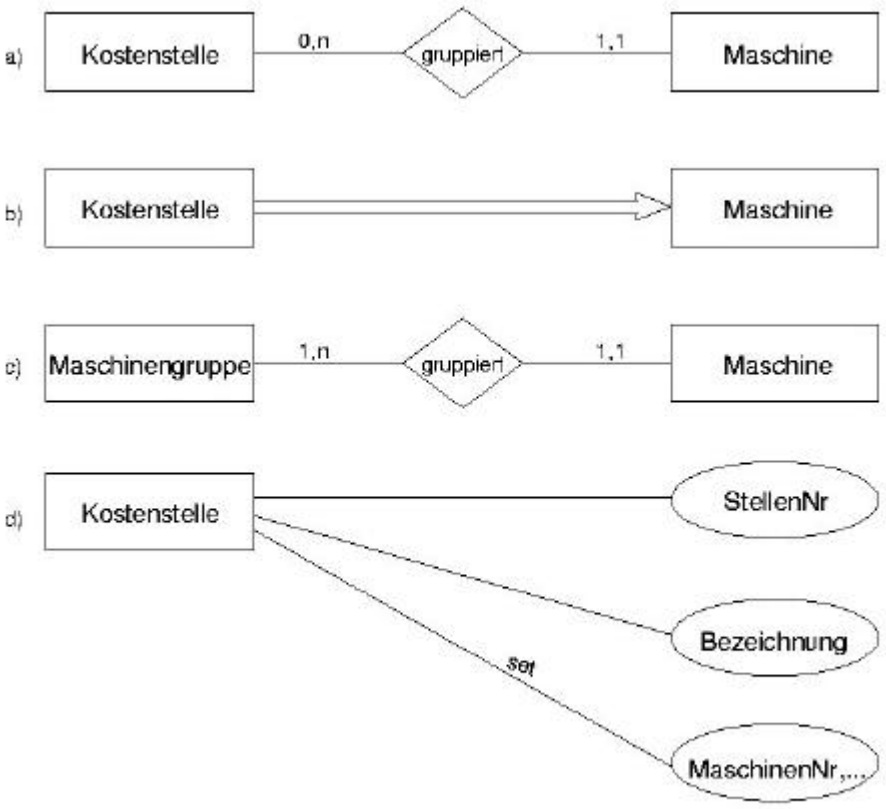


Abb. 36: Beispiele für Gruppierungen

5.4 Generalisierung

Bereits Smith und Smith hatten darauf hingewiesen, daß die Generalisierung verschiedene Eigenschaften annehmen kann (s. Kapitel "Allgemeine Erweiterungen des Entity-Relationship-Modells", S. 20). Neben der Unterscheidung in disjunkte und nicht-disjunkte Teilmengen können die nicht-disjunkten Teilmengen ihrerseits weitere Charakteristiken aufweisen. Abbildung 37 zeigt mögliche Charakteristiken, wie sie im folgenden als sinnvoll erachtet werden.

Charakteristiken Der Generalisierng	Disjunkt $T_i \cap T_j = \varnothing$ $\forall$ i,j $\in$ {1,2...n} $\wedge$ i $\neq$ j	nicht disjunkt $T_i \cap T_j \neq \varnothing \exists$ i,j $\in$ {1,2...n} $\wedge$ i $\neq$ j	
		Untermenge $T_i \subset T_j$	Schnittmenge $T_i \setminus T_j \neq \wedge$ $T_j \setminus T_i \neq \varnothing$
vollständig $G=T_1 \cup T_2 \cup .... \cup T_n$	(1)	(3)	(5)
nicht- vollständig $G \supset T_1 \cup T_2 \cup ... \cup T_n$	(2)	(4)	(6)

Abb. 37: Charakteristiken der Generalisierung

Bei disjunkten Teilmengen einer Generalisierung müssen alle "n" Teilmengen paarweise disjunkt sein. Ist nur ein Paar nicht disjunkt, so kann eine Teilmenge eine Untermenge zur anderen Teilmenge sein, oder die Teilmengen sind teilweise überlappt. Gleiche Teilmengen sind nicht sinnvoll, da dann kein Grund für eine Spezialisierung besteht. Falls eine Teilmenge eine Untermenge einer anderen Teilmenge ist, kann dies auch als Generalisierung in der Generalisierung verstanden werden (vgl. Abbildung 38). Durch diese geschachtelte Generalisierung kann die ursprüngliche Generalisierung von der Eigenschaft nicht-disjunkt in paarweise disjunkt überführt werden.

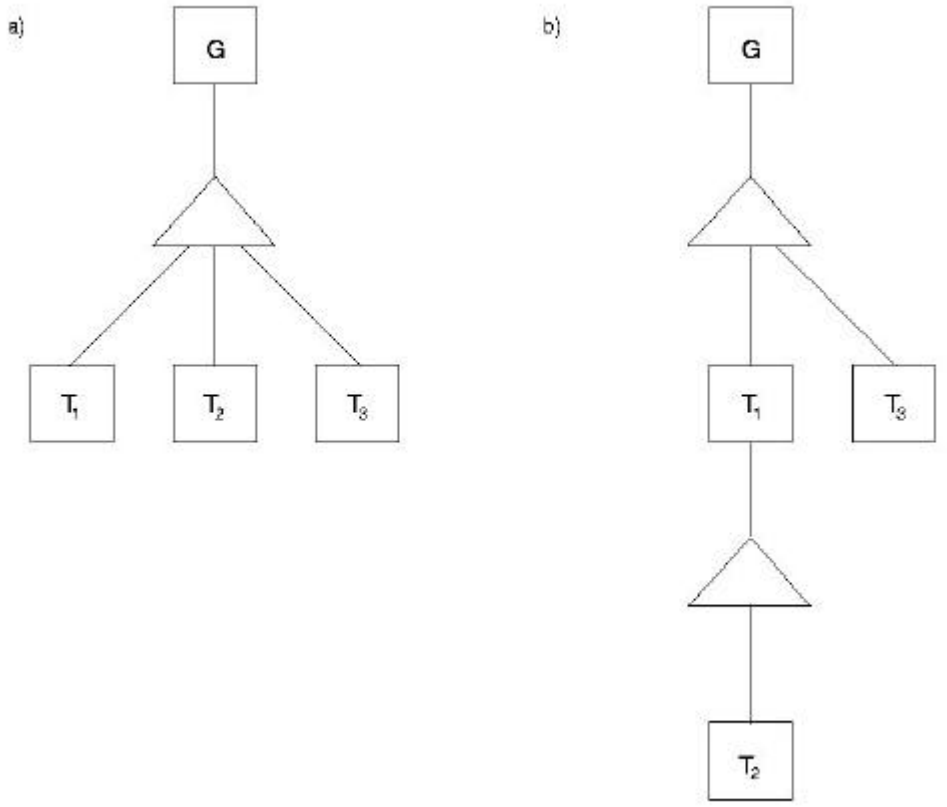


Abb. 38: Auflösung von Unterteilmengen (a) in geschachtelte Generalisierungen (b)

5.5 Aggregation alternativer Objekttypen

Objekttypen können zu beliebig vielen anderen Objekttypen unabhängige Beziehungen eingehen. Dies kann durch eine entsprechende Anzahl von Aggregationen ausgedrückt werden. Andererseits können Beziehungen auf Objektebene aber durchaus voneinander abhängig sein.

Abbildung 39 zeigt zwei Beispiele. In Beispiel a soll ein *Mitarbeiter* entweder in *Projekt*en oder in der Produktion in Fertigungsbereichen (*FertigBereich*) eingesetzt werden. Beziehungen eines Mitarbeiters zu *Projekt*en schließen Beziehungen zu dem Typ *FertigBereich* aus und umgekehrt. In jedem Beziehungstyp sollen unterschiedliche Informationen, z. B. über Entlohnung, gespeichert werden. In Beispiel b soll der *Teilebedarf* entweder durch einen *Lagerbestand* oder durch einen Beschaffungsauftrag

(*BeschaffAuftrag*) gedeckt werden. Beide Beziehungen enthalten außer den referenzierten Objekttypen die gleichen Informationen.

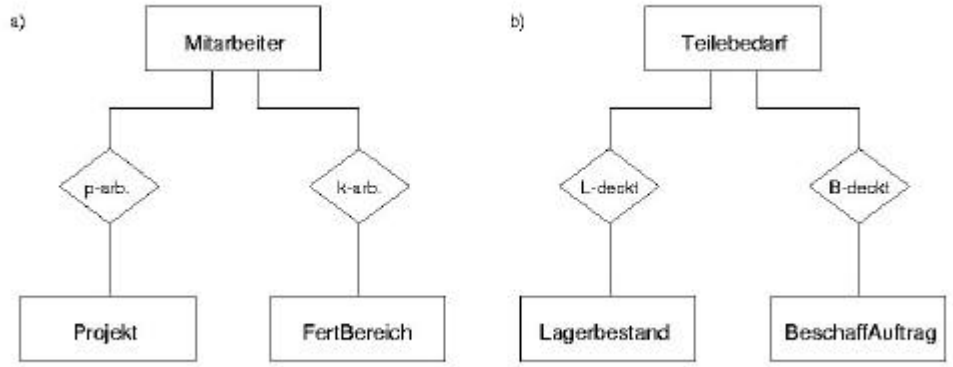


Abb. 39: Beispiel zu alternativen Beziehungen

Abbildung 40 zeigt eine mögliche Lösung mit Hilfe der Generalisierungsoperation (Lenzerini/Santucci 83). In Beispiel a wird der Objekttyp spezialisiert, so daß anschließend jede Spezialisierung eine Beziehung eingehen kann, während in Beispiel b die an den Beziehungen beteiligten Objekttypen generalisiert wurden, so daß danach nur ein Beziehungstyp notwendig ist.

In Beispiel a kann die Generalisierung vollständig sein, falls die Beziehungen der Subtypen einen Komplexitätsgrad von (1,x) besitzen, oder unvollständig sein, falls auch Komplexitätsgrade (0,x) zulässig sind. In Beispiel b ist nur eine vollständige Generalisierung sinnvoll. Die Lösung über den Generalisierungsoperator für Objekttypen ist zumindest im Beispiel b fragwürdig, da die beiden Teilobjekte außer der Beziehung keinerlei Gemeinsamkeit besitzen und auch keinen Schlüssel nicht aus der gleichen Domäne aufweisen (vgl. hierzu Eberlein 84).

Entsprechend der Ausführungen bezüglich der Komplexitätsgrade bei binären Aggregationen können auch bei der Aggregation alternativer Objekttypen des Beispiels b verschiedene Komplexitätsgrade zweier Kanten unterschieden werden. Dies ist zum einen die Kante zwischen der Aggregation *deckt* und den Objekttypen *Teilebedarf* und *Teiledeckung* aus Abbildung 40 b. Daraus folgt, daß die Kanten zwischen dem Objekttypen *Teilebedarf* und den Aggregationen *L-deckt* und *B-deckt* einerseits, sowie die Kanten zwischen den beiden Aggregationen und den Objekttypen *Lagerbestand* und *BeschaffAuftrag* andererseits, die gleichen Komplexitätsgrade besitzen müssen (s. Abbildung 39 b).

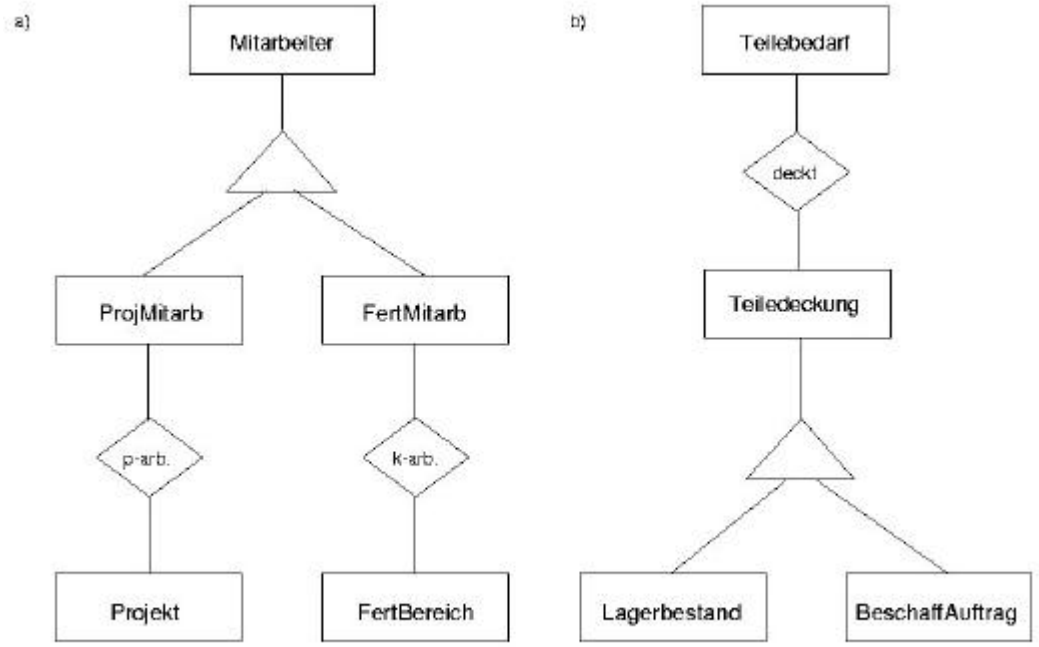
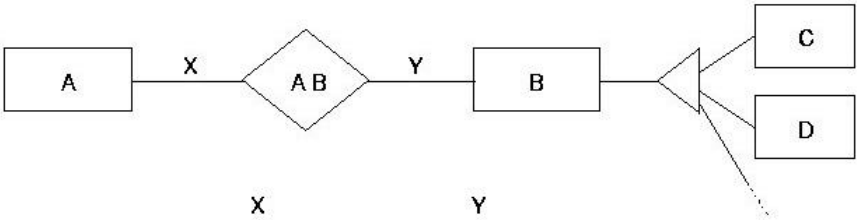


Abb. 40: Lösung alternativer Beziehungen mit Generalisierung

Im Gegensatz zu den binären Aggregationen ist die Darstellung allerdings nicht symmetrisch, so daß sich die Anzahl der unterscheidbaren Komplexitätsgradkombinationen auf 16 erhöht (vgl. Abbildung 41). Dabei ist die Anzahl der alternativen Objekttypen beliebig.

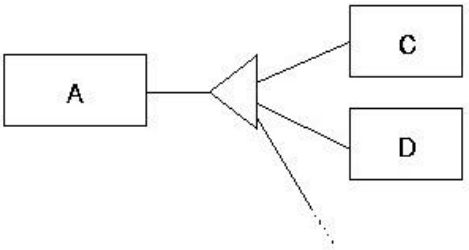
Die Aggregationen von Typ (2a), (3), (5a) und (7a) haben gemeinsam, daß die Kante zwischen dem hinzugefügten Objekttyp B und der Aggregation AB eine Komplexität von (1,1) besitzt. Dies bedeutet, daß jedes existierende Objekt der Typen C, D, usw. genau eine Beziehung zu einem Objekt des Typs A eingehen muß. Durch die Existenzbedingung und die alternative Beziehung verhalten sich die Objekttypen zu dem Objekttyp A wie Subtypen einer Generalisierung zu dem generischen Objekttyp. Abbildung 42 zeigt eine entsprechende Darstellung sowie eine Zuordnung zu den Generalisierungstypen. Es sei aber darauf hingewiesen, daß im Gegensatz zu der Aggregation die Generalisierung keine eigenen Attribute besitzen kann, und die Schlüssel der beteiligten Objekttypen nicht die gleiche Domäne besitzen müssen.



	X	Y
(1)	0,1	0,1
(2a)	0,1	1,1
(2b)	1,1	0,1
(3)	1,1	1,1
(4a)	0,n	0,1
(4b)	0,1	0,n
(5a)	0,n	1,1
(5b)	1,1	0,n
(6a)	1,n	0,n
(6b)	0,n	1,n
(7a)	1,n	1,1
(7b)	1,1	1,n
(8)	0,n	0,n
(9a)	1,n	0,n
(9b)	0,n	1,n
(10)	1,n	1,n

Abb. 41: Arten binärer Aggregationen alternativer Objektypen

Neben den einfachen binären Aggregationen können auch rekursive Aggregationen oder Mehrfachaggregationen alternative Objekttypen betreffen. Abbildung 43 zeigt eine binäre, rekursive Aggregation. Ein Fertigungsauftrag benötigt Ausgangsmaterial für die Produktion, das entweder über einen Beschaffungsauftrag oder über einen anderen Fertigungsauftrag gedeckt sein kann. Dementsprechend verbindet die Aggregation *deckt* die Objekttypen *FertigAuftrag* und *Auftragsdeckung*, wobei der letztere eine Generalisierung der Typen *FertigAuftrag* und *BeschaffAuftrag* darstellt.



(2a)	0,1	1,1	=>	nicht-vollständig, disjunkte Generalisierung
(3)	1,1	1,1	=>	vollständig, disjunkte Generalisierung
(5a)	0,n	1,1	=>	nicht-vollständig, nicht-disjunkte Generalisierung
(7a)	1,n	1,1	=>	vollständig, nicht-disjunkte Generalisierung

Abb. 42: Gegenüberstellung der Aggregationen alternativer Objekttypen und der Generalisierung

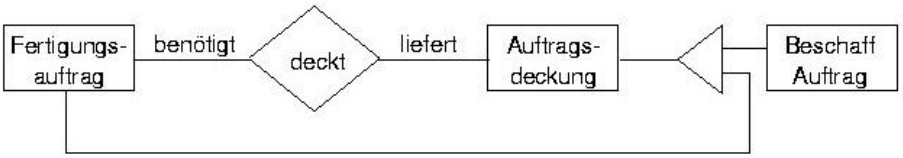


Abb. 43: Beispiel rekursiver, binärer Aggregationen alternativer Objekttypen

5.6 Versionsbehaftete Objekte

Unter einer Version sollen die Werte der Attribute eines Objektes zu einem bestimmten Zeitpunkt, der sich von anderen Zeitpunkten unterscheidet, verstanden werden. Mit Hilfe der Folgen von Versionen kann die Geschichte eines Objektes als chronologische Reihe der Zustände betrachtet werden.

An die Versionen eines Objektes sollen folgende Anforderungen gestellt werden (Müller/Steinbauer 83, S. 77f):

(1) Eine Änderung eines Objektes führt zu einer neuen Version, bei der die Objektidentifikation, d. h. das Objekt, erhalten bleibt.

(2) Zu einem Zeitpunkt darf nur genau eine Version eines Objektes existieren. Entsteht eine neue Version des Objektes, wird die vorherige Version damit automatisch abgelöst.

(3) Die Geschichte eines Objektes von der Entstehung bis zum Untergang muß lückenlos mit Versionszuständen belegbar sein.

(4) Eine Objektidentifikation darf nach dem Untergang des Objektes nicht für ein anderes Objekt verwendet werden.

In diesem Sinne stellt die Version als eines Objektes eine Objektes eine Revision dar (Wilkes 89). Davon sind Alternativen als unterschiedliche Realisierungsansätze eines Objektes, Varianten als prinzipiell gleiche Objekte mit unterschiedlichen Ausprägungen einzelner Attribute und Repräsentation als unterschiedliche Darstellungsform des gleichen Objekts abzugrenzen. Auch implizite Versionen eines Datenbanksystems, die in der Ausführungsebene für die Effizienzsteigerung bei verteilten Datenbeständen (z. B. zu Synchronisationszwecken oder für Recovery-Maßnahmen) vorgehalten werden (Meier 87, S. 50), und auf die der Benutzer keinen direkten Zugriff hat, sollen nicht betrachtet werden.

In Abhängigkeit von der Aussagemöglichkeit über ein Objekt zu einem bestimmten Zeitpunkt lassen sich verschiedene Geschichtsarten unterscheiden (Klopprogge 83a, S. 475 ff; Klopprogge 83b, S. 8 ff; Härder 84, S. 5 ff). Je nach Geschichtsart können Aussagen zu Zeitpunkten gemacht werden, an denen keine Aufzeichnung erfolgte (s. Abbildung 44):

(1) Bei der **zustandserhaltenden** Geschichte bleibt der Wert einer Aussage bis zu dem Zeitpunkt einer neuen Aussage konstant. Ein Beispiel ist der Kostensatz einer Maschine, der von einem Festsetzungszeitpunkt bis zu einem nächsten Festsetzungszeitpunkt gleich bleibt. Um den Kostensatz zu einem bestimmten Zeitpunkt zu ermitteln, wird nur der Wert zu dem letzten Festsetzungszeitpunkt vor dem gewünschten Zeitpunkt benötigt. Eine statische Betrachtung entspricht in der Regel der zustandserhaltenden Geschichte ab dem letzten Änderungszeitpunkt.

(2) Bei der **zustandserhaltenden** Geschichte sind nur genaue Aussagen zu den Änderungszeitpunkten möglich. Für eine exakte Dokumentation sind unendlich viele Aufzeichnungszeitpunkte nötig. Häufig werden Werte zwischen den Zeitpunkten über Interpolation geschätzt. Als Beispiel lassen sich Temperaturen und Druckmessungen von Maschinen nennen.

(3) Können Aussagen über die Werte des Objekts für jeden Zeitpunkt aus den Änderungszeitpunkten errechnet werden, so spricht man von **ableitbarer** Geschichte. So läßt sich ein Sparguthaben jederzeit als eine Funktion aus dem letzten Guthaben, dem Zinssatz sowie der Zeitspanne errechnen.

Für die zustandsverändernde und ableitbare Geschichte gilt, daß die Anforderung nach lückenlosen Versionszuständen nicht über Datensätze, sondern über Funktionen (annähernd) bestimmt wird. Darüber hinaus unterscheidet Klopprogge noch **ereignisorientierte** Geschichte und Geschichte mit **ungenauen Zeit- und Wertangaben**, die hier nicht weiter berücksichtigt werden sollen (Klopprogge 83a, S. 477).

Für die Betrachtung der Änderungsgeschichte sind drei Zeitwerte zu unterscheiden (Klopprogge 83b, S. 39 ff; Härder 84, S. 8 ff). Die zum **Bestimmungspunkt** beobachtete Aussage wird als Änderung zum **Aufzeichnungszeitpunkt** festgehalten. Die Aussage soll ab dem **Gültigkeitszeitpunkt** wahr sein. Während der Aufzeichnungszeitpunkt nie vor dem Bestimmungszeitpunkt liegen darf, kann der Gültigkeitszeitpunkt bei vorausschauender Bestimmung später oder bei nachträglicher Bestimmung früher liegen.

Es zeigt sich, daß durch Einbringen neuer Erkenntnisse die Objektgeschichte verändert werden kann (vgl. Abbildung 45). So liefert die Frage: "Welchen Wert hatte das Objekt zum Zeitpunkt t_1?" eine andere Aussage, wenn sie zum Zeitpunkt t_2 oder zum Zeitpunkt t_3 gestellt wird. Allgemein muß die Frage nach einer detaillierten Aussage lauten: "Welcher Wert gilt zur Zeit t_g gemäß dem Erkenntnisstand von t_b, so wie dieser zur Zeit t_a aufgezeichnet war?" (Klopprogge 83b, S. 47).

Mit den bisherigen Modellierungskomponenten kann ein versionsbehafteter Objekttyp nur unzureichend konstruiert werden. Abbildung 46 zeigt einen Objekttyp A (vgl. auch Müller/Steinbauer 83; Ferg 85; Studer 88).

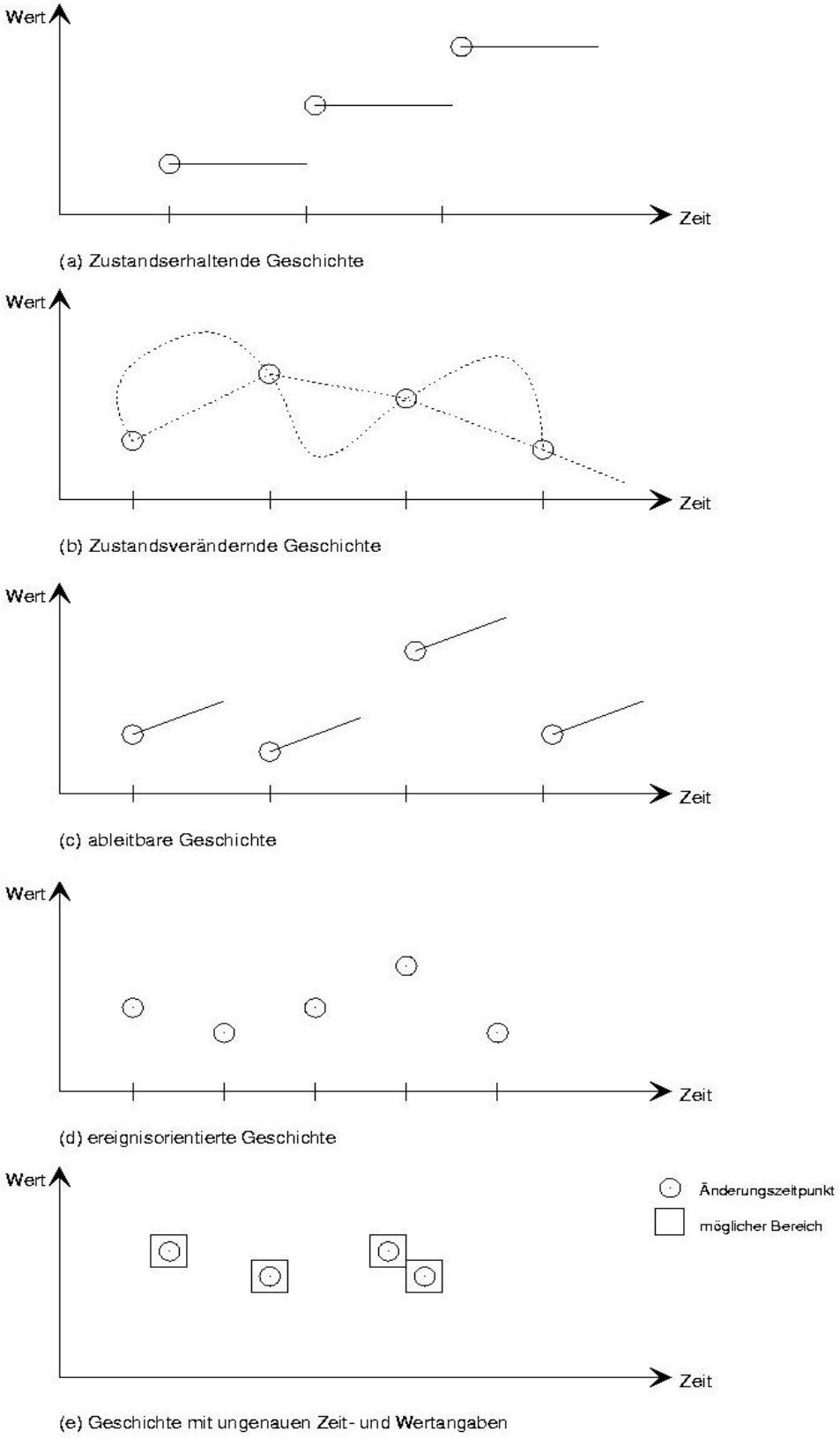


Abb. 44: Arten von Geschichte (in Anlehnung an Klopprogge 83a)

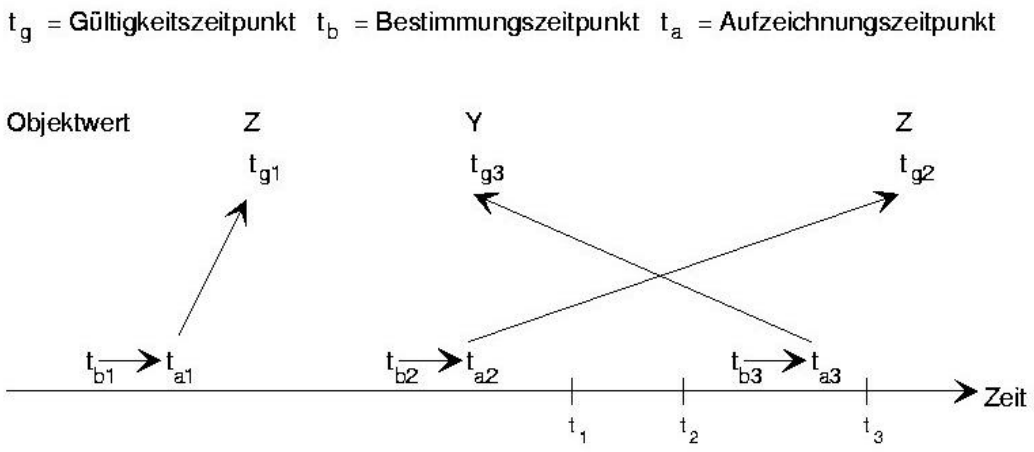


Abb. 45: Dynamische Zeitwerte der Änderungsgeschichte

Der Typ A wird als eine Beziehung zwischen der Domäne des Schlüssels *ANr* und der *Zeit* als Gültigkeitszeitpunkt modelliert. Jede Beziehung, die der Typ *ANr* mit der *Zeit* eingeht, ist eine neue Version des gleichen Objektes A. Der Typ A wird spezialisiert in einen Typ lebender Objekte und einen Typ mit Historienobjekten, wobei die Submengen sowohl vollständig als auch disjunkt sind.

Zusätzlich gelten mindestens die folgenden Integritätsbedingungen:

(1) Das Abspeichern eines neuen Objektes A ist nur möglich, wenn das entsprechende Objekt *ANr* nicht bereits Beziehungen eingegangen ist, d. h. das gleiche Objekt nicht bereits existiert oder existiert hat.

(2) Das Ändern eines Objektes A führt dazu, daß der alte Inhalt von der Spezialisierung *A-lebend* in die *A-Historie* übernommen wird mit dem Vermerk der Eintragzeit, die der neuen Zeit des Typs *A-lebend* entspricht, sowie ein neues Objekt A mit der Spezialisierung *A-lebend* anstelle des alten entsteht.

(3) Das Löschen eines Objektes A führt zum Übertrag von der Spezialisierung *A-lebend* in *A-Historie* mit entsprechendem Vermerk im Attribut *Eintragzeit*.

Nicht berücksichtigt ist die bereits oben erwähnte Unterscheidung der Zeitwerte in Gültigkeit, Bestimmungs- und Aufzeichnungszeitpunkte, die Möglichkeit des nicht

chronologischen Entstehens der Zeitwerte sowie die Möglichkeit der Korrektur der Geschichte (Härder 87, S. 11).

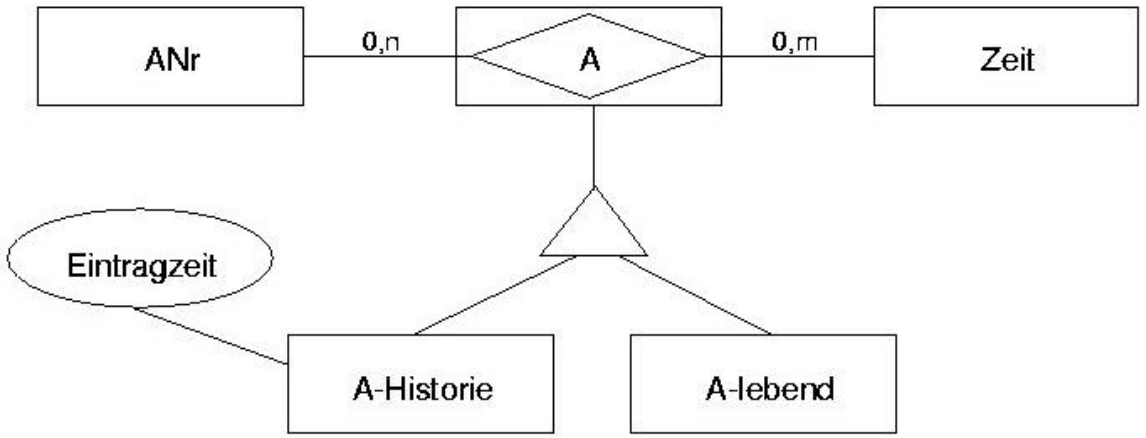


Abb. 46: Modellierung eines versionsbehafteten Objekttyps

5.7 Clusterbildung und komplexe Objekte

Unter einem Cluster soll eine Zusammenfassung von Objekttypen, Aggregationen, Gruppierungen und Generalisierungen verstanden werden. Eine solche Zusammenfassung stellt keinen Modellierungsoperator mit zusätzlicher Semantik dar, sondern dient vor allem dazu, eine Darstellung eines großen Netzes anschaulicher und übersichtlicher zu gestalten. Dies ist beispielsweise in folgenden Fällen sinnvoll:

(1) Komplexe oder hierarchisch strukturierte Objekte, die sich selbst aus Objekten unterschiedlicher Typen zusammensetzen und aufwendige Integritätsbedingungen besitzen (Härder et al. 85, S. 7 ff), können in einem Cluster zusammengefaßt werden. Für die weitere Modellierung, bei der die "interne" Struktur des Objektes nicht berücksichtigt werden muß, genügt die Angabe des Clusters.

(2) Ein großes Modell mit mehreren hundert Objekttypen, den zugehörigen Beziehungen sowie den verbindenden Kanten kann leicht unübersichtlich wirken. Mit Hilfe hierarchisch geordneter Cluster kann eine solche Darstellung, analog zu den Strukturen von Datenflußdiagrammen (DeMarco 78), übersichtlich gestaltet werden.

Gleichzeitig kann man damit dem Problem der bildlichen Darstellungsgröße begegnen.

(3) Mit unterschiedlichen Ebenen können unterschiedliche Zielgruppen angesprochen werden. Für einen Endbenutzer kann eventuell der Abstraktionsgrad eines Clusters genügen, während für die Implementierung eine genaue Spezifikation benötigt wird (Feldmann/Miller 86, S. 348 ff).

(4) Anwendungsbezogen, funktional abgegrenzte Teilgebiete lassen sich in Cluster zusammenfassen und dadurch klarer voneinander abgrenzen (Teorey et al. 89, S. 980 ff).

Abbildung 47 zeigt eine mögliche Clusterung eines Modells nach Funktionsbereichen.

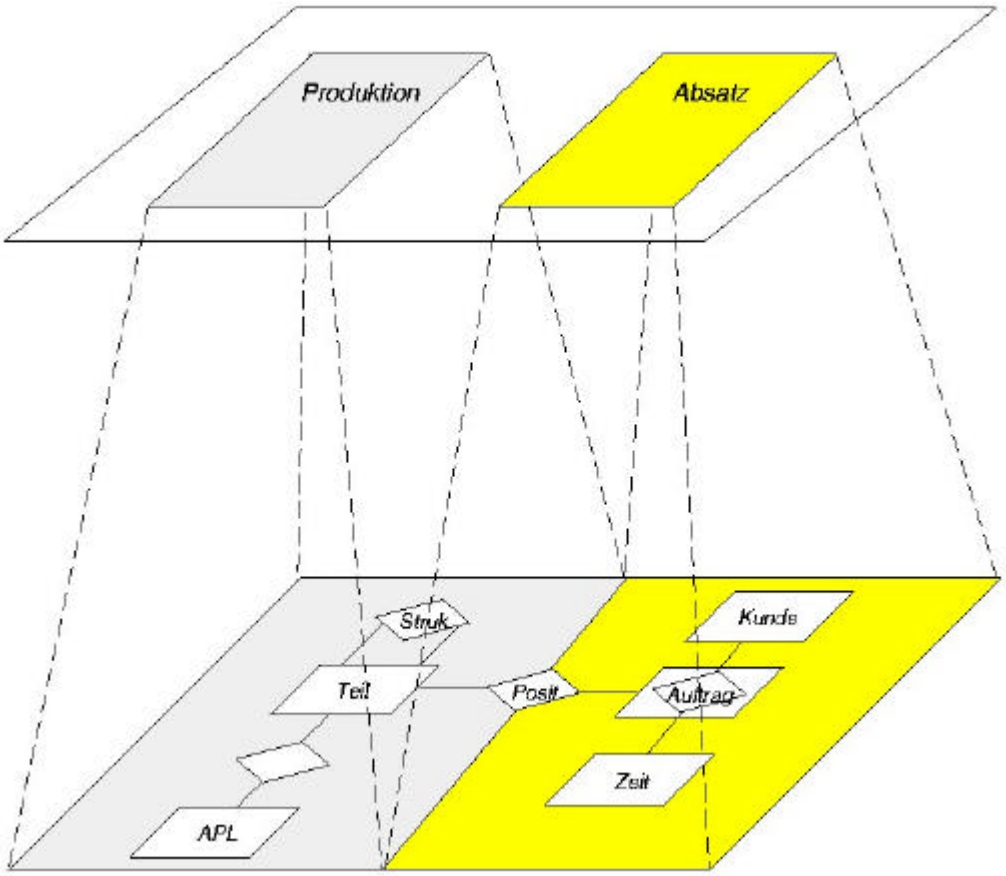


Abb. 47: Clusterung eines Modells nach Funktionsbereichen

5.8 Semantische Integritätsbedingungen

Unter Integrität wird die Korrektheit und die Konsistenz des Modells gegenüber der abzubildenden Wirklichkeit verstanden. Lipeck unterscheidet die Integritätsbedingungen in modellinhärente und Integritätsbedingungen (Lipeck 89). Modellinhärente Integritätsbedingungen werden unmittelbar durch das Modell der gewählten Beschreibungssprache gewährleistet. Explizite Integritätsbedingungen werden in statische, transitionale und dynamische Bedingungen unterteilt. Statische Bedingungen schränken die möglichen Abbildungszustände auf zulässige Zustände ein, transitionale Bedingungen beschreiben den Übergang von einem zulässigen Zustand in einen anderen und dynamische Bedingungen sind Folgen zulässiger Zustände.

Röhrle und Kratzer (Röhrle/Kratzer 88) differenzieren darüber hinaus nach dem Wirkungsbereich (intrarelational und interrational), dem Prüfzeitpunkt (direkt, versetzt, benutzerinitiert), dem Komplexitätsgrad (Einzelwerte, Wertmengen, Objektmengen), dem Bedingungsziel bei dynamischen Bedingungen (Vorzustand, Nachzustand, Übergang) und nach möglichen Reaktionen auf Integritätsverletzungen (Abbruch, Hinweis, Trigger).

Im folgenden soll eine Klassifizierung der statischen Integritätsbedingungen für die Bewertung der Beschreibungssprachen aufgestellt werden. Hierbei erfolgt eine Trennung in modellinhärente und semantische Integritätsbedingungen.

Die transitionalen und dynamischen Integritätsbedingungen beschreiben das Verhältnis unterschiedlicher Abbildungszustände untereinander und sind damit dem Ablaufsteuerungsmodell zuzuordnen. Physische und technische Integritätsbedingungen (Steinbauer/Wedekind 85) wie z. B. Wiederherstellung einer Abbildung nach Systemfehler (s. hierzu Reuter 81) sollen als Probleme der Ausführungsebene (s. Kapitel "Die Architektur eines Informationssystems", S. 5) hier nicht diskutiert werden.

Unter modellinhärenten Bedingungen werden die bei der Modellierung implizit durch die Modellkomponenten und -restriktionen der Beschreibungssprache gegebenen Integritätsbedingungen verstanden. Steinbauer/Wedekind bezeichnen sie deshalb als Integritätsbedingungen (Steinbauer/Wedekind 85).
Hierunter fallen beispielsweise die Bedingungen, daß

- Attributausprägung eines Objektes nur Werte aus der Domäne des Attributtyps annehmen kann,

- identische Attributtupel und damit gleiche Objekte in einem Objekttyp nur einmal auftreten können,

- Beziehungen innerhalb eines Beziehungstyps nur existieren können, wenn entsprechende Objekte der beteiligten Objekttypen existieren oder

- Objekte in einer Submenge nur existieren, wenn ein entsprechendes Objekt in der generischen Menge der Generalisierung vorhanden ist.

Bei Implementierung einer Datenbank mit dem Relationenmodell werden diese strukturellen Integritäten z. B. mit Hilfe der

- Typintegrität, der

- Primärschlüsselintegrität, der

- referentiellen Integrität und der

- semireferentiellen Integrität

gewährleistet. Integritätsbedingungen, die nicht mit den modellinhärenten Bedingungen abgedeckt werden, müssen zusätzlich durch semantische Integritätsbedingungen modelliert werden. Die semantische Integrität soll die Menge der möglichen Abbildungszustände auf sinnvolle und gewünschte Zustände reduzieren. Steinbauer und Wedekind bezeichnen dies als semantische Datenintegrität und pragmatische Integrität (Steinbauer/Wedekind 85).

Objekttypinterne Integritätbedingungen

Objekttypinterne Integritätsbedingungen wirken nur auf einen einzigen Objekttyp. Dabei können sie sich entweder auf ein einzelnes Objekt beschränken, sich auf eine Teilmenge der Objekte der Objektklasse oder auf die Menge aller Objekte beziehen. Objektgebundene Bedingungen lassen sich ohne großen Aufwand durchführen, da von der Konsistenzprüfung nur eine Klasse betroffen ist (Rebsam 83).

(1) Integritätsbedingungen für ein einzelnes Objekt (**Einzelobjekt**) lassen sich folgendermaßen untergliedern:

(1.1) Domänenbeschränkungen beziehen sich auf ein einziges Attribut innerhalb eines Objektes, z. B. ist die **Domäne** der Postleitzahlen auf Werte größer oder gleich 1000 und kleiner 9000 beschränkt.

(1.2) Einzelne Attribute eines Objektes können von der Ausprägung eines anderen Attributs oder Attribute des gleichen Objektes abhängig sein (Hohenstein et al. 87). Diese Bedingung soll als **abgeleitetes Attribut** bezeichnet werden. Beispielsweise läßt sich das Attribut *Alter* aus dem Attribut *Geburtsdatum* und der aktuellen Zeit ableiten.

(1.3) Ausprägungen einzelner Attribute eines Objektes können aufgrund von Ausprägungen eines anderen Attributs des gleichen Objektes ausgeschlossen werden. Beispielsweise schließt der Wert "ledig" des Attributs *Familienstand* die Werte "drei" bis "fünf" des Attributs *Steuerklasse* aus. Diese Bedingung soll als **bedingtes Attribut** bezeichnet werden.

(1.4) Attribute eines Objektes können direkt voneinander abhängig sein. Beispielsweise muß der Wert des Attributs *Starttermin* immer kleiner sein als der Wert des Attributs *Endtermin*. Diese Bedingung soll als **wechselseitig abhängige Attribute** bezeichnet werden.

(2) Ein Attributwert eines Objektes kann von Werten des gleichen Attributs anderer Objekte aus der gleichen Objektklasse abhängig sein. So kann es bsw. bei fortlaufend vergebenen Auftragsnummern zwingend sein, daß alle Aufträge mit einer niedrigeren Auftragsnummer als der des betrachteten Auftrags auch einen kleineren Generierungs- oder Starttermin besitzen. Diese Bedingung soll als **Objektteilmengenabhängigkeit** bezeichnet werden.

(3) Objektmengenabhängige Integritätsbedingungen sind z. B. die minimale oder maximale Anzahl von Objekten oder Summen- und Durchschnittswerte von Attributausprägungen der Objekte eines Objekttyps. Lenzerini und Santucci sprechen bei der minimalen und der maximalen Anzahl von Objekten von "absolute cardinality constraint" (Lenzerini/Santucci 83). Allgemein sollen diese Bedingungen als **Objektmengenkardinalität** bezeichnet werden.

Beziehungstypabhängige Integritätsbedingungen

Beziehungstypabhängige Integritätsbedingungen zwischen Objekttypen werden danach unterschieden, ob eine einzelne Beziehung oder das Verhältnis mehrerer Beziehungen zueinander betroffen sind:

(1) Bedingungen für Beziehungen eines **einzelnen Beziehungstyps**:

(1.1) **Komplexitätsgrade** dienen zur Spezifizierung eines Beziehungstyps und müssen, soweit sie nicht modellinhärent sind, als zusätzliche Integritätsbedingung angegeben werden. Sie können mit Hilfe von funktionalen Abhängigkeiten spezifiziert werden. (vgl. auch Kapitel "Aggregation", S. 46)

(1.2) Attribute können aus den Beziehungen abgeleitet werden, z. B. das Attribut *Mitarbeiteranzahl* im Typ *Abteilung*. Diese Bedingung soll **beziehungsabgeleitetes Attribut** genannt werden.

(1.3) Nur eine Teilmenge der Objekte eines Objekttyps darf eine Beziehung eines Beziehungstyps eingehen bzw. ist für sie gesperrt. Abbildung 48 a zeigt ein Stücklistenbeispiel, bei dem Objekte des Typs *Teil*, die fremdbezogen werden, keine Beziehung der Rolle *Stückliste*, und solche, die *Endprodukt*e sind, keine Beziehung der Rolle *Verwendung* eingehen dürfen. Die in Abbildung 48 b gezeigte Lösung hat den Nachteil, daß die Teilestruktur über verschiedene Beziehungstypen verteilt ist (**Objektausprägungsabhängige Beziehung**).

(1.4) Nur bestimmte Objektkombinationen zweier (oder mehrerer) Typen dürfen eine Beziehung eines definierten Beziehungstyps eingehen bzw. sind dafür gesperrt. In Abbildung 49 ist das Beispiel einer Mitarbeiterhierarchie gezeigt. Ein *Mitarbeiter* kann mehrere Untergebene haben, selbst aber nur einen Vorgesetzten. Zusätzlich soll gelten, daß ein Vorgesetzter nicht jünger als seine Untergebenen sein darf. Diese Altersrestriktion ist nicht wie im vorherigen Beispiel über eine Spezialisierung zu lösen, da ein Mitarbeiter nicht generell die Eigenschaften *Vorgesetzter* oder *Untergebener* annehmen kann, sondern nur im Verhältnis zu einem anderen Mitarbeiter (**Objektkombinationsabhängige Beziehung**).

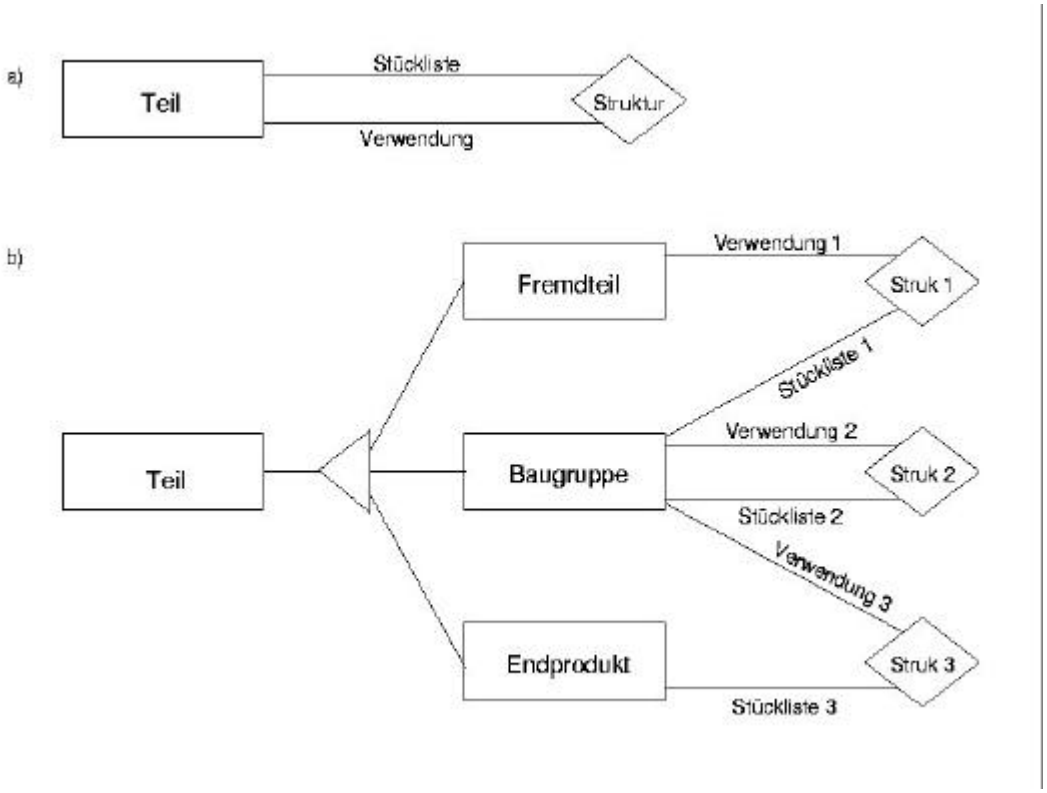


Abb. 48: Objektausprägungsabhängige Beziehung

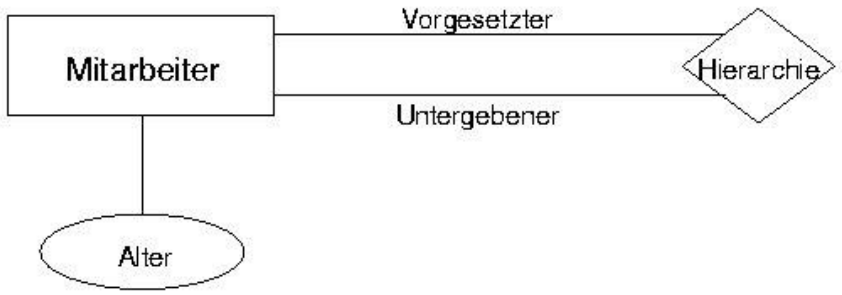


Abb. 49: Objektkombinationsabhängige Beziehung

(2) Semantische Integritätsbedingungen, die **mehrere Beziehungstypen** betreffen, lassen sich in folgende Fälle unterscheiden:

(2.1) Die Beziehung in einem Beziehungstyp eines Objektes hängt von einer Beziehung desselben Objektes in einem anderen Beziehungstyp ab (Tabourier/Nancy 83, Davis/Bonell 89, DeTroyer 89, Lin et al. 89). Abbildung 50 zeigt einen Objekttyp mit zwei Beziehungen.

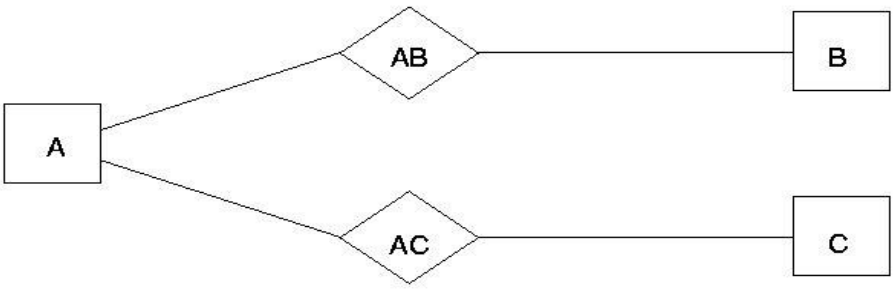


Abb. 50: Semantische Integritätsbedingungen zwischen zwei Beziehungen

Prinzipiell lassen sich die Bedingungen danach klassifizieren, ob sich die Beziehungen gegenseitig ausschließen (2.1.1) oder ob sie voneinander abhängen sind (2.1.2).

(2.1.1) Auf die Exklusivität haben sowohl die Unter- und Obergrenze der Anzahl der Beziehungen, die ein Objekt mit beiden Beziehungstypen eingehen kann, Einfluß als auch der Faktor, ob die Exklusivität nur für die erste Beziehung gilt.

	(2.1.1.1)	(2.1.1.2)	(2.1.1.3)	(2.1.1.4)	(2.1.1.5)
Exklusivität aller Beziehungen	j	j	j	j	n
mindestens 1 Beziehung insgesamt	j	n	n	j	j
maximal 1 Beziehung insgesamt	j	j	n	n	n

Abb. 51: Charakterisierung beziehungstypabhängiger Integritätsbedingungen mit Exklusivität

(2.1.1.1) Ein Objekt des Typs A geht genau eine Beziehung ein, entweder vom Typ AB oder AC.

(2.1.1.2) Ein Objekt des Typs A geht maximal eine oder keine Beziehung vom Typ AB oder AC ein.

(2.1.1.3) Ein Objekt des Typs A muß alle Beziehungen entweder vom Typ AB oder AC eingehen.

(2.1.1.4) Ein Objekt des Typs A muß mindestens eine Beziehung eingehen. Alle Beziehungen müssen aber entweder vom Typ AB oder AC sein.

(2.1.1.5) Ein Objekt des Typs A kann beliebige Beziehungen des Typs AB und AC eingehen, mindestens aber eine Beziehung entweder vom Typ AB oder AC.
Bei "n" Beziehungstypen lautet die Bedingung, daß mindestens "m" Beziehungen existieren müssen, wobei "m" Werte zwischen 1 und (n-1) annehmen kann.

Diese Bedingungen können prinzipiell mit der Möglichkeit der Aggregation alternativer Objekttypen Fall (a) ausgedrückt werden, die den gleichen Sachverhalt aus umgekehrter Perspektive betrachten (vgl. Kapitel "Aggregation alternativer Objekttypen", S. 62).

(2.1.2) Die **Abhängigkeit** zweier Beziehungen kann einseitig oder zweiseitig sein:

(2.1.2.1) Eine oder mehrere Beziehungen des Objektes a im Typ AB setzen mindestens eine Beziehung im Typ AC voraus, d. h. Beziehung AB impliziert Beziehung AC.

(2.1.2.2) Geht ein Objekt a eine oder mehrere Beziehungen vom Typ AB ein, so müssen auch eine oder mehrere Beziehungen vom Typ AC existieren und umgekehrt. Diese Bedingung kann, wenn beide Beziehungen von A einen Komplexitätsgrad von (0,1) besitzen, als Dreierbeziehung dargestellt werden. Abbildung 52 zeigt eine äquivalente Darstellung zweier binären Beziehungstypen mit Gleichheitsinterdependenz und einer Beziehung vom Grad drei, die für folgende Komplexitätsgrade der beiden Zweierbeziehungen gilt:

(2.1.2.2.1) Die Komplexitätsgrade sind x_1 = (0,n) und x_2 = (0,n). Daraus ergeben sich folgende Abhängigkeiten:
$a \rightarrow b, c$
Dies entspricht Fall (5) der Dreierbeziehung (s. Kapitel "Mehrfachaggregation", S. 50)

(2.1.2.2.2) Die Komplexitätsgrade sind x_1 = (0,1) und x_2 = (0,n). Es gelten folgende Abhängigkeiten:
$a \rightarrow b, c \wedge b \rightarrow a, c$
Dies entspricht dem Fall (6) der Dreierbeziehung.

(2.1.2.2.3) Beide Komplexitätsgrade sind vom Typ (0,1). Damit gelten folgende Abhängigkeiten:
$a \rightarrow b, c \wedge b \rightarrow a, c \wedge c \rightarrow a, b$
Dies entspricht dem Fall (7) der Dreierbeziehung.

Die Integritätsbedingungen können auch auf mehr als zwei Beziehungstypen angewendet werden und gelten dann analog. Da diese Arten der Integritätsbedingungen im Gegensatz zu den folgenden nur die unmittelbar mit einem Objekttyp in Verbindung stehenden Beziehungen betrifft, sollen diese als bezeichnet werden.

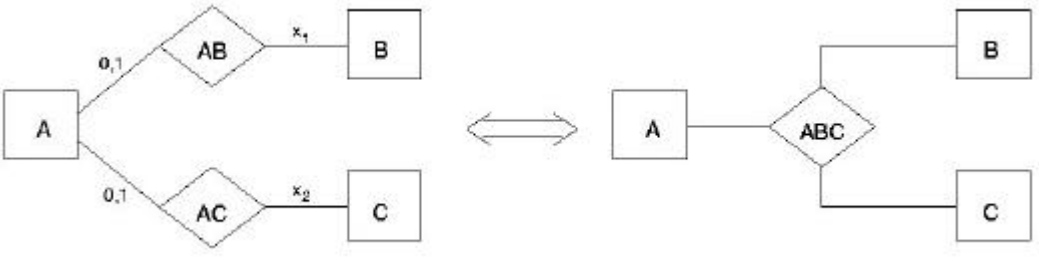


Abb. 52: Äquivalente Darstellung zweier Beziehungstypen mit Gleichheitsinterdependenz und einer Dreierbeziehung

(2.2) Beziehungskomplexitäten, wie sie in den Kapiteln "Allgemeine Erweiterungen des Entity-Relationship-Modells" (s. S. 20) und "Aggregation" (s. S. 46) definiert sind, können auch über mehrere Beziehungen hinweg wirken. Abbildung 53 zeigt ein Beispiel, bei dem *Maschine*n zu *Maschinengruppe*n und diese zu *Kostenstelle* gruppiert werden. Einer Maschine kann ein *Wartungsbeauftragter* zugeordnet werden, der seinerseits maximal für 20 Maschinen zuständig ist. Gleichzeitig soll aber gelten, daß diese 20 Maschinen auf maximal 3 Kostenstellen verteilt sind, d. h. von dem Typ *Wartungsbeauftragter* zu dem Typ *Kostenstelle* gibt es eine Obergrenze, ohne daß eine direkte Beziehung besteht. Bedingungen dieser Art sollen **Beziehungspfadkomplexität** als bezeichnet werden.

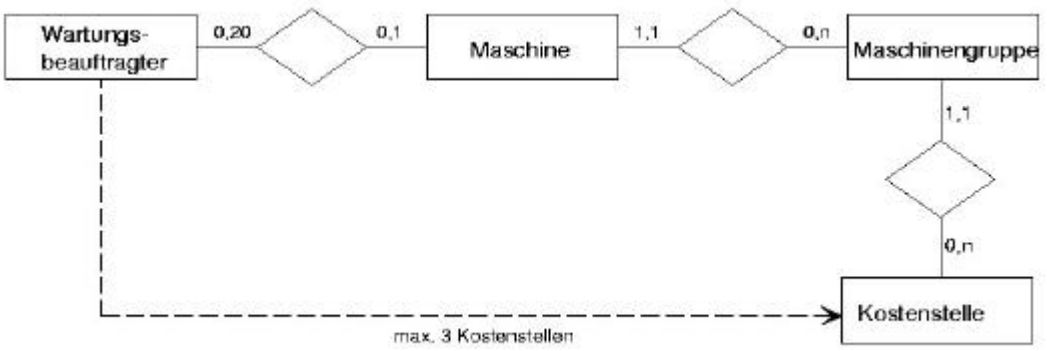


Abb. 53: Komplexitätsgrad über mehere Beziehungen

(2.3) Objekte verschiedener Typen können über verschiedene Beziehungspfade miteinander verbunden sein. Häufig ist es notwendig, daß eine Beziehung über einen Pfad auch eine Beziehung über einen anderen Pfad voraussetzt. Diese Arten der Integritätsbedingungen sollen als **Beziehungspfadbedingungen** bezeichnet werden.

Abbildung 54 zeigt ein Stücklisten-Arbeitsplanbeispiel (vereinfacht aus: Scheer 90f, S. 165). Ein *Arbeitsgang* ist immer genau einem *Arbeitsplan* zugeordnet, der seinerseits mehreren *Teil*en zugeordnet sein kann. In einem Arbeitsgang können mehrere Teile, die als Stücklistenkomponenten für die Fertigung des Zielteils notwendig sind, montiert werden. Dies wird durch den

Beziehungstyp *AGKomp* zwischen den Typen *Arbeitsgang* und *Struktur* dargestellt, wobei die Kante zwischen *Struktur* und *Teil* mit der Rolle *UT* auf die betreffenden Komponenten verweist.

Gleichzeitig muß aber auch gelten, daß die Kante mit der Rolle *OT* auf das herzustellende Teil verweist, für das der Arbeitsplan mit dem Arbeitsgang gilt. Dies bedeutet, daß der Beziehungspfad zwischen einem Objekt des Typs *Arbeitsgang* und einem Objekt des Typs *Teil* über die Kanten *Arbeitsgang-AGKomp-Struktur-OT-Teil* auch einen Beziehungspfad der beiden Objekte über die Kanten *Arbeitsgang-Arbeitsplan-APLZuo-Teil* voraussetzt.

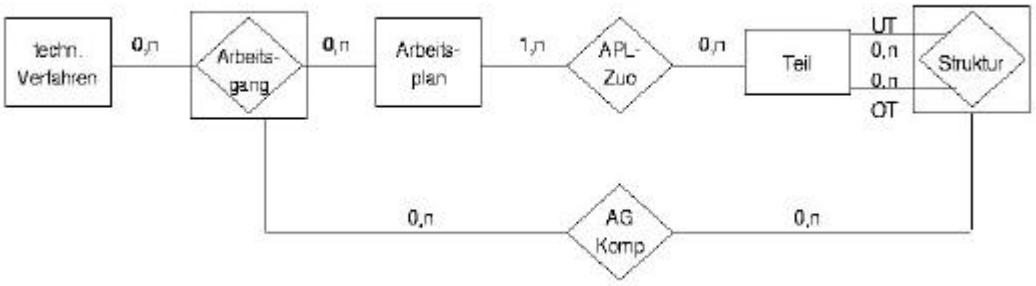


Abb. 54: Implizierung eines Beziehungspfades durrch einen anderen Pfad

Allgemein lassen sich analog zu den Integritätsbedingungen zwischen zwei Beziehungen verschiedene Arten von Abhängigkeiten über Beziehungspfade unterscheiden. Abbildung 55 zeigt zwei Objekttypen A und B, die über Pfade verbunden sind. Eine Verbindung über Pfad 1 vom Typ A nach B soll als $P_1(A,B)$ bezeichnet werden.

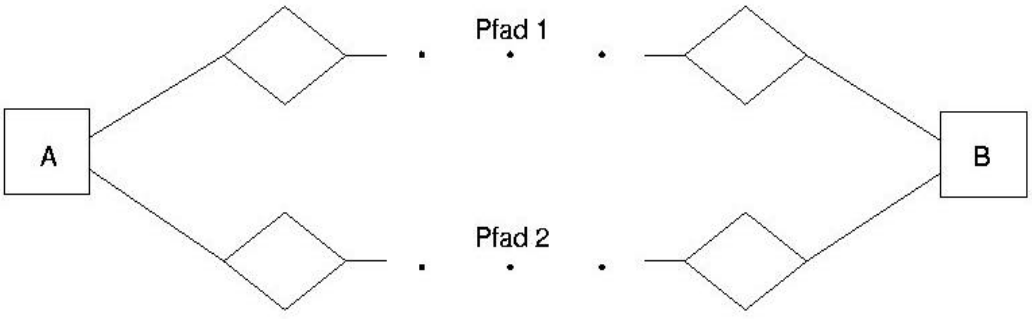


Abb. 55: Semantische Integritätsbedingung zwischen Beziehungspfaden

(2.3.1) Ein Objekt des Typs A kann entweder über Pfad 1 oder über Pfad 2 mit einem Objekt des Typs B verbunden sein, nicht jedoch über beide Pfade:

$$\{(a,b) \mid a \in A \wedge b \in B \wedge (a,b) \in P_1(A,B)\} \cap \{(a,b) \mid a \in A \wedge b \in B \wedge (a,b) \in P_2(A,B)\} \neq \varnothing$$

(2.3.2) Objekte des Typs A müssen mindestens entweder über Pfad 1 oder Pfad 2 mit einem Objekt des Typs B verbunden sein:

$$\{a \mid a \in A \wedge a \in P_1(A,B)\} \cup \{a \mid a \in A \wedge a \in P_2(A,B)\} = A$$

Dies impliziert, daß jeweils die erste Beziehung je Pfad von A ausgehend Integritätsbedingungen zwischen zwei Beziehungen der Typen (2.1.1) besitzt.

(2.3.3) Es gelten Bedingung (2.3.1) und (2.3.2), d. h. ein Objekt von Typ A ist genau über einen Pfad mit einem Objekt des Typs B verbunden. Auch dies impliziert, daß jeweils die erste Beziehung je Pfad von A ausgehend Integritätsbedingungen zwischen zwei Beziehungen von einer der Typen (2.1.1) besitzt.

(2.3.4) Eine Verbindung der Objekte a und b über Pfad 1 setzt eine Verbindung über Pfad 2 voraus:

$$\{(a,b) \mid a \in A \wedge b \in B \wedge (a,b) \in P_1(A,B)\} \subseteq \{(a,b) \mid a \in A \wedge b \in B \wedge (a,b) \in P_2(A,B)\}$$

Dies impliziert, daß die erste Beziehung des Pfades 1 von der ersten Beziehung des Pfades 2 im Sinne der Bedingung (2.1.2.1) abhängig ist. Das Beispiel der Abbildung 54 enthält eine solche Integritätsbedingung.

(2.3.5) Die Voraussetzung aus (2.3.4) gilt wechselseitig:

$$\{(a,b) \mid a \in A \wedge b \in B \wedge (a,b) \in P_1(A,B)\} = \{(a,b) \mid a \in A \wedge b \in B \wedge (a,b) \in P_2(A,B)\}$$

Dies impliziert, daß jeweils die ersten Beziehungen je Pfad gegenseitig abhängig sind im Sinne der Bedingung (2.1.2.2).

Die Beziehungspfadkomplexität (2.2) läßt sich auch durch einen zusätzlichen Beziehungstyp und eine Beziehungspfadbedingung ausdrücken. Abbildung 56 zeigt das Beispiel aus Abbildung 53. Um die Beziehungspfadkomplexität auszudrücken, wurde der neue Beziehungstyp *WartBeaufKoSt* (Wartungsbeauftragter-Kostenstellen-Zuordnung) zwischen den Entitytypen *Wartungsbeauftragter* und *Kostenstelle* aufgenommen, der aus Sicht des Typs *Wartungsbeauftragter* eine Komplexität von (0,3) besitzt. Aus Sicht des Typs *Kostenstelle* ist der Komplexitätsgrad nicht spezifizert. Gleichzeitig muß eine Beziehungspfadbedingung von Typ (2.3.4) gelten, wobei Pfad 2 von dem Objekttyp *Wartungsbeauftragter* über die Typen *Maschine* und *MGruppe* zu dem Objekttyp *Kostenstelle* verläuft, und der Pfad 1 durch den neuen Beziehungstyp *WartBeaufKoSt* repräsentiert wird.

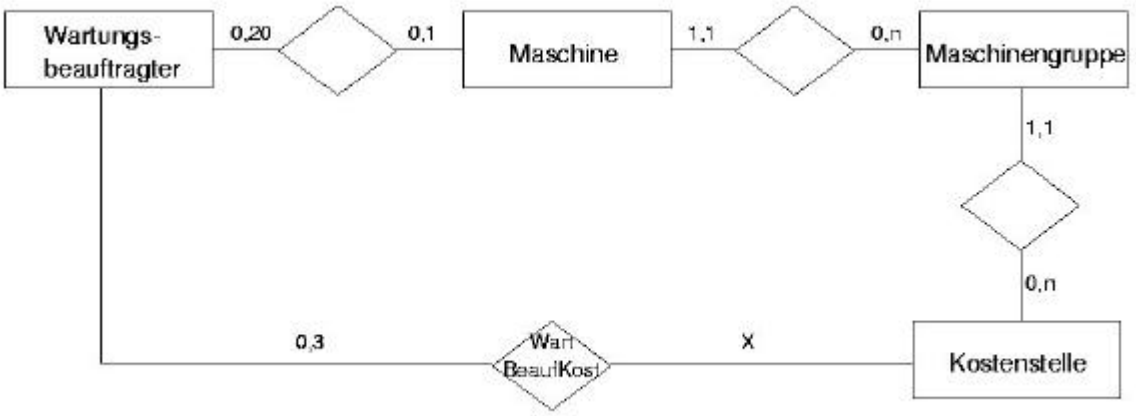


Abb. 56: Beziehungspfadkomplexität als Beziehungspfadbedingung

(2.4) Ein Objekttyp kann über einen Beziehungspfad mit sich selbst verbunden sein. Bei dem kürzesten Pfad geht der Typ direkt mit sich selbst eine Beziehung ein, z. B. bei der Stückliste (vgl. Abbildung 48). Es ist häufig sinnvoll, einen Zyklus auszuschließen, so daß ein Objekt weder direkt noch rekursiv über beliebig viele Stufen eine Beziehung mit sich selbst eingehen kann. Bei dem Stücklistenbeispiel bedeutet dies, daß ein Teil nicht als Komponente in sich selbst enthalten sein kann, was zumindest in der

Stückgutproduktion sinnvollerweise auszuschließen ist. Diese Bedingung soll als **Rekursionsbedingung** bezeichnet werden

Abbildung 57 zeigt zusammenfassend einen Überblick über die dargestellten semantischen Integritätsbedingungen.

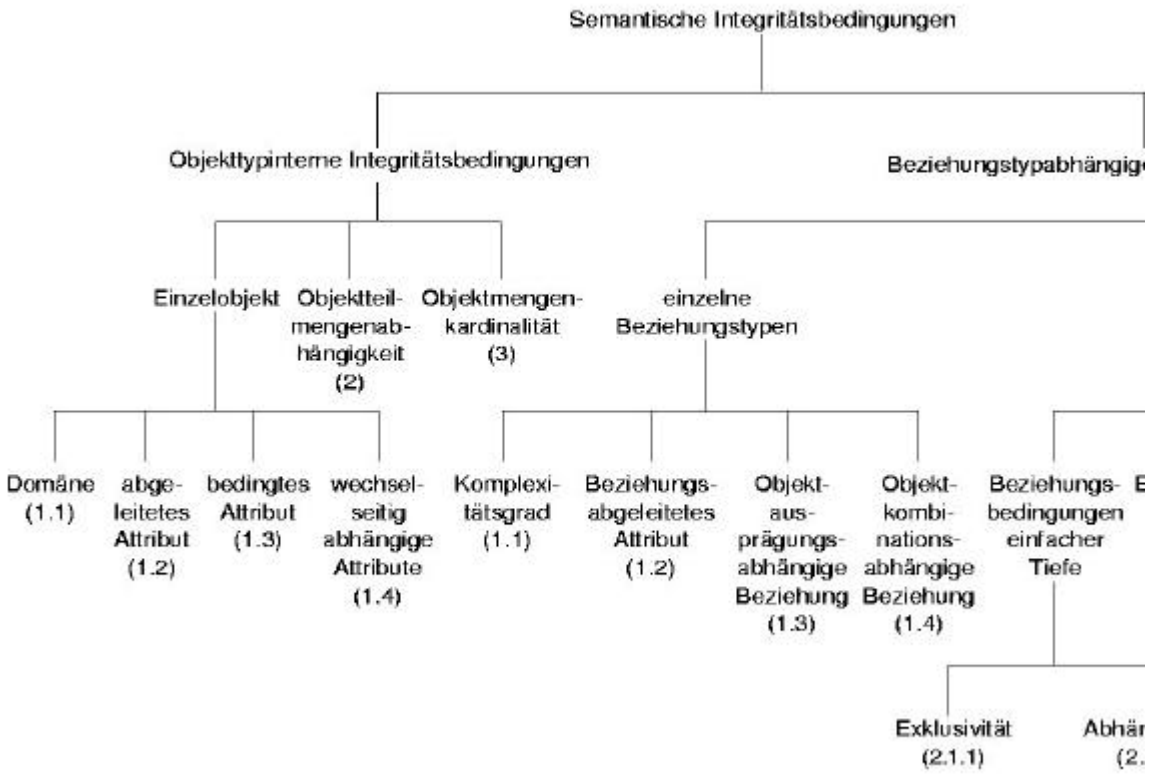


Abb. 57: Zusammenfassung der semantischen Integritätsbedingungen

5.9 Beurteilung der Beschreibungssprachen

Die Beurteilung der Beschreibungssprachen nach den vorgestellten Bewertungskriterien ist in Abbildung 58 dargestellt.

Die **Klassifizierung** läßt sich prinzipiell in allen Modellen darstellen. Da aber weder im NIAM-Modell noch im Semantic-Association-Modell für die Klassifizierung eigene Symbole verwendet werden, ist sie in den graphischen Darstellungen dieser Modelle schwer erkennbar. In NIAM-Modellen ohne Unterscheidung von LOTs und NOLOTs wird keine Klassifizierung unterstützt.

Binäre Aggregationen sind im Entity-Relationship-Modell mit eingeschränkten Komplexitätsangaben möglich. Die Aggregationen können durch Attribute näher beschrieben werden. Die Einschränkungen der Komplexitätsangaben sind bei dem Erweiterten ERM, beim Strukturierten ERM sowie beim Entity-Category-Relationship-Modell aufgehoben. Bei dem NIAM-Modell können (m:n)-Beziehungen mit eigenen Attributen nur über eine Auflösung in zwei (1:n)-Beziehungen und einer Uniqueness-Integritätsbedingung (vgl. Abbildung 19 b) abgebildet werden. Im Entitäten-Diagramm werden Aggregationen als Beziehungskanten zwischen Entitäten dargestellt, wobei (m:n)-Beziehungen immer in (1:n)-Beziehungen aufgelöst werden müssen. Zusätzliche Integritätsbedingungen wie im NIAM-Modell können nicht angegeben werden. Im Semantic-Association-Modell kann eine binäre Beziehung über eine Interaction Association und zugehörige Attribute über eine zusätzliche Aggregation Association abgebildet werden. Explizite Komplexitätsangaben sind nicht vorgesehen.

Aggregationen von mehr als zwei Objekttypen sind in allen Modellen außer dem NIAM-Modell und dem Entitäten-Diagramm direkt abbildbar. Allerdings unterstützt kein Modell hinreichend die Möglichkeit der Komplexitätsangaben bzw. der Spezifizierung der Aggregationsart. Bei dem NIAM-Modell sowie dem Entitäten-Diagramm muß jeweils ein neuer Objekttyp eingeführt werden, der dann seinerseits binäre Beziehungen zu den an der Aggregation beteiligten Objekttypen eingeht.

Die **Gruppierung** ist über die Aggregation mit Komplexitätsangaben in allen Modellen außer dem ERM und dem SAM* möglich. Das im ERM vorgeschlagene Weak Entity ist bei mehreren Beziehungen desselben nicht eindeutig.

Generalisierungen sind, außer im Entity-Relationship-Modell, in allen Modellen möglich, die sich nur in der Mächtigkeit (vollständig, disjunkt, leere Mengen) der Abbildung unterscheiden.

Aggregationen alternativer Objekttypen sind in keinem Modell direkt abbildbar. Allerdings können die Alternativen über den Umweg der Generalisierung bzw. der Category im ECRM modelliert werden.

Eine **Versionsunterstützung** sowie die **Clusterbildung** und die Darstellung **komplexer Objekte** werden, außer von SAM*, von keiner Modelldarstellung geboten. In SAM* wird dies mit Hilfe der Composition Association dargestellt, wobei die Versionsdarstellung nicht über ein einfaches Aufzählen der Versionen hinausgeht und eher der Alternativendarstellung

dient. Für die Clusterbildung können alle Arten von Associations mit einer Composition Association zusammengefaßt werden.

Kriterium \ Verfahren	ERM	EERM	SERM	ECRM	NIAM	ED	SAM*
Klassifizierung	++	++	++	++	+	++	+
Binäre Aggregation	o	++	++	++	+	o	o
Mehrfach-Aggregation	o	o	o	o	-	-	o
Gruppierung	o	+	+	+	+	+	-
Generalisierung	--	o	o	o	+	o	+
Aggreg. altern. Objekttypen	--	-	-	o	-	-	-
Versionsunterstützung	--	--	--	--	--	–	-
Clusterbildung	--	--	--	--	--	–	o
modellinhärente Integr.-Bed.	-	o	o	o	+	o	+
Objekttypinterne Integr.-Bed.	--	--	--	--	--	–	--
Beziehungstypabhängige Integr.-Bed.	--	-	-	-	o	-	--
Konstruktionsprozeß	-	+	+	+	-	-	o

Bewertungsskala: (++) = sehr gut; (+) = gut; (o) = möglich; (-) = schlecht; (--) = nicht möglich

Abb. 58: Beurteilung der Beschreibungssprachen

Die **modellinhärenten Integritätsbedingungen** sind u. a. von der Fülle der bereitgestellten Konstruktionsoperatoren abhängig. Dadurch ist die Bewertung indirekt in den vorherigen Bewertungspunkten enthalten. Alle darüber hinausgehenden Bedingungen müssen zusätzlich modelliert werden.

Bis auf die Beziehungskomplexität stellen außer dem NIAM-Modell keine Darstellungen graphische Hilfsmittel für die Modellierung der **Integritätsbedingungen** bereit. In den NIAM-Diagrammen kann ein Teil der **beziehungstypabhängigen Integritätsbedingungen** modelliert werden.

Die Darstellung des **Konstruktionsprozesses** und damit die Möglichkeit für einen Dritten, den Modellierungsverlauf nachzuvollziehen, wird bei dem Erweiterten ERM, dem SERM und dem ECRM durch die Trennung von Objekttypen, Attributen und Aggregationen gut unterstützt. Auch die Uminterpretationsmöglichkeit der Beziehungstypen beim EERM, die gerichteten Kanten beim SERM sowie das Category-Konzept beim ECRM erhöhen die Lesbarkeit. Da im SAM* alle Typen von Associations als Kreise dargestellt werden, ist die Lesbarkeit nicht gut. Beim Entitäten-Diagramm und NIAM-Diagramm müssen Mehrfachbeziehungen aufgelöst werden, so daß die anwendungsnahe Darstellung verloren geht.

Wie die Bewertung der Darstellungsmethoden zeigt, weisen die auf dem Entity-Relationship-Modell aufbauenden Methoden EERM, SERM und ECRM sowie das SAM* die besten Ergebnissen auf. Besonders bezüglich nicht modellinhärenten, semantischen Integritätsbedingungen ist allerdings eine Verbesserung notwendig, wie sie ansatzweise im NIAM zu finden ist. Da das Entity-Relationship-Modell-Diagramm die am meisten eingesetzte Methode ist, soll in das Erweiterte Entity-Relationship-Modell die fehlenden Darstellungsmöglichkeiten übernommen werden. Das daraus resultierende Modell soll in Abgrenzung zu den bereits bestehenden Begriffen Erweitertes oder Extended ERM als **Expanded Entity-Relationship-Modell** (PERM) bezeichnet werden.

6. Expanded Entity-Relationship-Modell

Für das Expanded Entity-Relationship-Modell (**PERM**) werden die Strukturen des Entity-Relationship-Modells und die am häufigsten benutzten Erweiterungen übernommen. Dies sind die Darstellung des Entitytyps als Rechteck, des Beziehungstyps als Raute und die der Generalisierung als Dreieck. Beziehungstypen können für den weiteren Konstruktionsprozeß zu Entitytypen uminterpretiert werden (vgl. Kapitel "Allgemeine Erweiterungen des Entity-Relationship-Modells", S. 20). Integritätsbedingungen werden in Anlehnung an Hohenstein et al. als offene Hexagone (Hohenstein et al. 87, S. 67 ff), Cluster als doppelt umrahmte Rechtecke abgebildet (vgl. Abbildung 59). Neben den für das Entity-Relationship-Modell geltenden Bedingungen sollen bei der Erstellung eines Expanded Entity-Relationship-Modells die im folgenden aufgestellten Regeln beachtet werden.

6.1 Entitytypen und Beziehungstypen

Entitytypen und Beziehungstypen können Attribute besitzen. Beziehungstypen werden, wenn sie Ausgangspunkt eines neuen Beziehungstyps sind, zu einem Entitytyp uminterpretiert. Entitytypen, die nicht aus einer Uminterpretation entstanden sind, werden als **Kern-Entitytypen** bezeichnet.

Entitytypen und Beziehungstypen haben Determinanten, mit denen ein einzelnes Entity oder eine einzelne Beziehung identifiziert werden kann. Eine solche Determinante ist bei einem Kern-Entity ein Attribut mit Primärschlüsseleigenschaft, das, wenn es nicht explizit angegeben wird, implizit aus dem Entitytypnamen gebildet wird. Beziehungstypen und damit auch aus Beziehungstypen uminterpretierte Entitytypen haben keine explizit angegebenen Schlüsselattribute. Die Determinante einer Beziehung wird vielmehr implizit aus den funktionalen Abhängigkeiten der Determinanten der die Beziehung bildenden Entitytypen abgeleitet. Besitzt ein Beziehungstyp mehrere Kanten zu einem Entitytyp, so wird die Determinante aus Entitytypname und Kantenrollenname gebildet. Ist bei einem Kern-Entitytyp mehr als ein Attribut in der Determinante angegeben, deutet dies darauf hin, daß der Typ eine nicht ausformulierte Beziehung darstellt. Eine solche Möglichkeit soll aus praktischen Überlegungen zulässig sein.

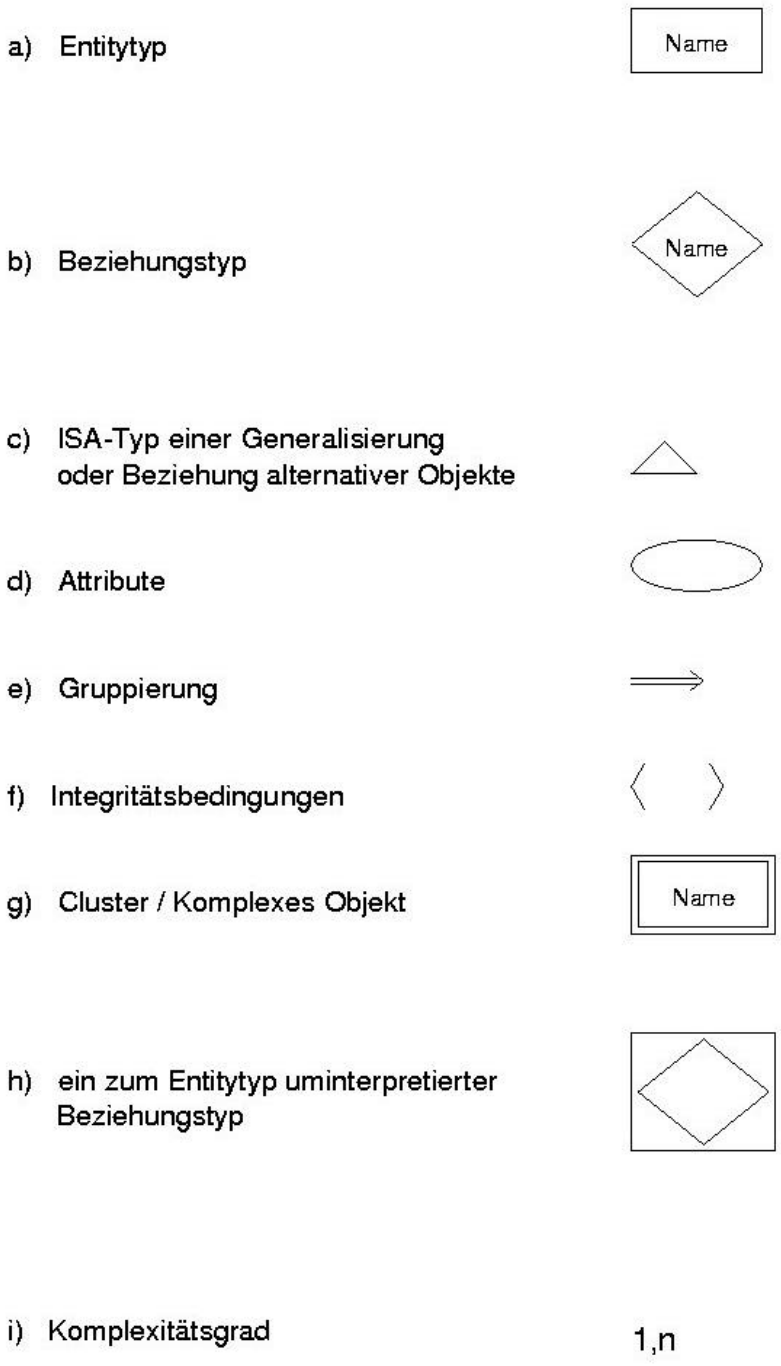


Abb. 59: Symbole des PERM-Diagramms

Beziehungen werden mit der (min,max)-Notation (vgl. Kapitel "Allgemeine Erweiterungen des Entity-Relationship-Modells", S. 20) beschriftet. Der Wert der Untergrenze muß immer kleiner oder gleich dem Wert der Obergrenze sein. Falls zwischen einem Entitytyp und einem Beziehungstyp mehrere Kanten existieren, müssen diese mit einem eindeutigen Rollennamen gekennzeichnet werden. Bei einer einzelnen Kante ist der Rollenname optional.

Bei Beziehungen vom Grad größer zwei werden die Abhängigkeiten (vgl. Kapitel "Mehrfachaggregation", S. 50) mit einer Integritätsbedingung angefügt. Zusätzlich kann die (min,max)-Notation angegeben werden. Dies ist dann notwendig, wenn der (min)-Wert ungleich null oder der (max)-Wert ungleich eins oder "n" ist. Es ist darauf zu achten, daß die (min,max)-Angabe nicht den Abhängigkeiten widerspricht. Abbildung 60 zeigt eine Dreifachbeziehung vom Typ 8, bei der zusätzlich ein Entity des Entitytyps A mindestens eine und höchstens 6 Beziehungen eingehen muß. Die funktionalen Abhängigkeiten einer binären Beziehung müssen nicht explizit angegeben werden, da diese aus der (min,max)-Notation abgeleitet werden können.

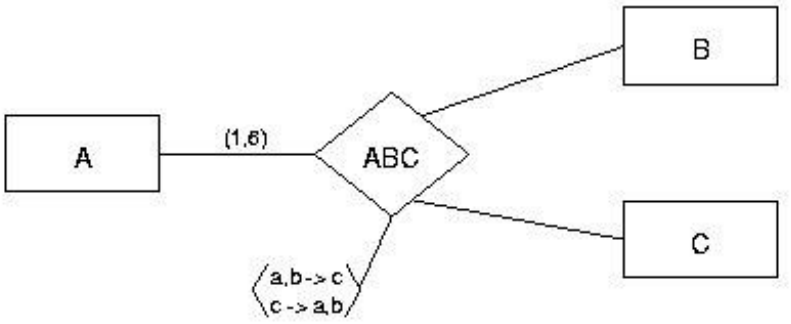


Abb. 60: Darstellung einer Dreierbeziehung im PERM

Abbildung 61 zeigt die Ableitung der Determinanten für die einzelnen Objekttypen. Es werden vier Entitytypen und zwei Beziehungstypen dargestellt. Während für die Entitytypen A, C und F explizit keine Determinanten angegeben sind, ist dem Typ D eine Attributkombination mit Primärschlüsseleigenschaft angefügt. Der binäre Beziehungstyp B ist mit einer Komplexitätsangabe gekennzeichnet. Der Dreifachbeziehungstyp E ist mit einer Integritätsbedingung spezifiziert, die die funktionalen Abhängigkeiten enthält. Es ergeben sich folgen Determinanten:

- Für die Typen A, C und F werden die Determinanten **A**, **C** und **F** abgeleitet.

- Der Typ D ist aufgrund seiner Attributkombination ein zum Entitytyp uminterpretierter Beziehungstyp. Zur Verdeutlichung wurde D gleichzeitig als Beziehungstyp zwischen den Entitytypen X und Y mit Hilfe der gestrichelten Linien dargestellt. Als Determinante ergibt sich **X,Y**.

- Aufgrund der Komplexitätsangabe des Beziehungstyps B läßt sich die funktionale Abhängigkeit ($a \rightarrow b$) ableiten. Daraus ergibt sich die Determinante **A** für den Beziehungstyp B.

- Für den Dreifachbeziehungstyp E sind zwei funktionale Abhängigkeiten gegeben. Da jede Abhängigkeit Primärschlüsseleigenschaft besitzt, lassen sich alternativ zwei Determinanten ableiten. Die erste Abhängigkeit wird durch das aus dem Typ B abgeleitete Attribut mitbestimmt, so daß dafür dessen Determinante A eingesetzt werden muß. Daraus ergeben sich die beiden möglichen Determinanten:
  ($b,d \rightarrow f$) $\wedge$ **A,X,Y** oder
  ($f,d \rightarrow b$) $\wedge$ **F,X,Y**

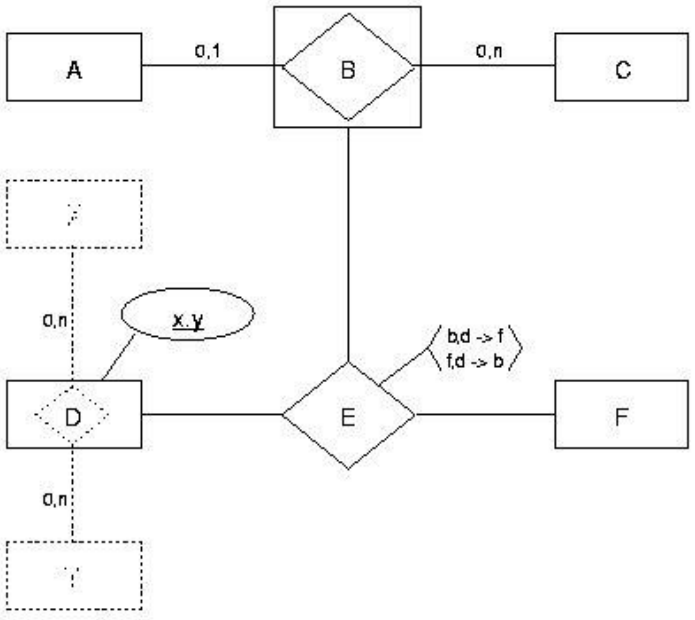


Abb. 61: Beispiel für die Ableitung von Determinanten im PERM

6.2 Gruppierung

Gruppierungen können als binäre Beziehungstypen dargestellt werden (s. Kapitel "Gruppierung", S. 60). Zusätzlich soll das Symbol eines Doppelpfeils benutzt werden können (vgl. Abbildung 59 e).

6.3 Generalisierung

Die mit dem graphischen Symbol des Dreiecks dargestellten Generalisierungen werden durch eine Kennzeichnung spezifiziert, die sich an die (min,max)-Notation der Komplexitätsgrade von Beziehungstypen anlehnt (vgl. hierzu Kapitel "Entitytypen und Beziehungstypen", S. 89). Ein (min)-Wert von null besagt, daß die Generalisierung nicht-vollständig ist, d. h. ein Entity der Generalisierung kann "null mal" spezialisiert sein. Entsprechend kennzeichnet ein (min)-Wert von eins eine vollständige Generalisierung. Ein (max)-Wert von eins besagt, daß ein Entity maximal einmal spezialisiert werden kann, d. h. die Teilmengen sind disjunkt (vgl. hierzu Kapitel "Generalisierung", S. 61). Bei nicht-disjunkten Teilmengen mit einer Obermenge wird diese mit "O", die Untermenge mit "U" gekennzeichnet. Abbildung 62 a a zeigt eine vollständige, disjunkte Generalisierung. Teil b der Abbildung zeigt eine nicht-vollständige, nicht-disjunkte Generalisierung, bei der die Menge des Entitytyps B eine Obermenge des Typs A ist.

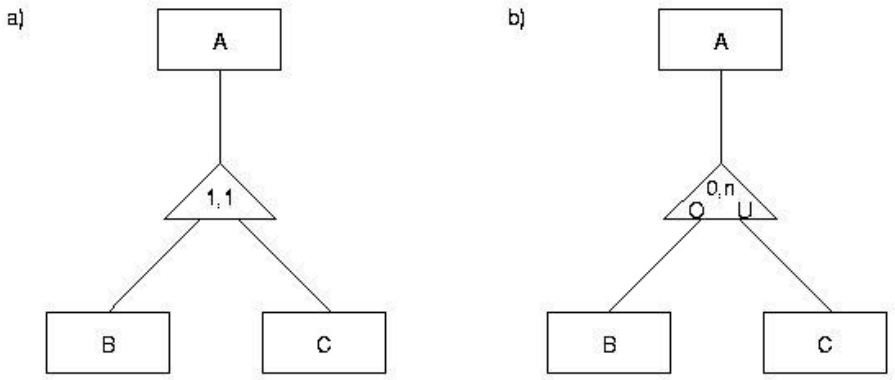


Abb. 62: Generalisierung im PERM

6.4 Beziehungstypen mit alternativen Entitytypen

Beziehungstypen mit alternativen Entitytypen werden mit dem gleichen Symbol wie die Generalisierung, dem Dreieck, dargestellt. Abbildung 63 a zeigt eine Beziehung, bei der die Kante mit der Rolle cd entweder zu genau einem Entity des Typs C oder zu einem Entity des Typs D geht. Die abgebildete Beziehung ist keine Dreifachbeziehung, sondern eine binäre Beziehung. Alle alternativen Entitytypen besitzen den gleichen Komplexitätsgrad. Die funktionale Abhängigkeit wird mit der Determinante einer der alternativen Entitytypen spezifiziert (Abbildung 63 b). Diese Darstellung entspricht dem Beispiel aus Abbildung 39 b.

Fall a wird mit zwei Beziehungstypen und entsprechenden Integritätsbedingungen dargestellt (s. "Beziehungstypabhängige Integritätsbedingung", S. 76).

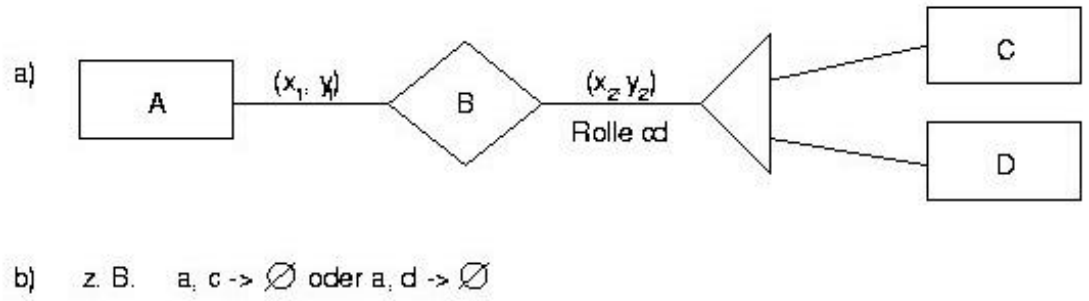


Abb. 63: Beziehungen alternativer Entitytypen im PERM

6.5 Versionsbehaftete Objekte

Die Komponenten Entitytyp, Beziehungstyp und Generalisierung sollen als versionsbehaftet deklariert werden können (vgl. Abbildung 64). Bei einem versionsbehafteten Entitytyp sollen folgende Bedingungen gelten:

(1) Jede Änderung eines Attributwertes eines Entities soll dokumentiert werden.

(2) Ein gelöschtes Entity soll dokumentiert bleiben.

(3) Ein neues Entity mit gleicher Determinantenausprägung wie ein bereits gelöschtes Entity muß von diesem unterscheidbar sein. Für beide müssen unabhängige Versionsgeschichten existieren.

(4) Beim Löschen eines Entities werden aufgrund modellinhärenter Integritätsbedingungen ebenfalls alle Beziehungen des Entities gelöscht. Dies gilt auch für versionsbehaftete Entitytypen. Daraus folgt, daß ein gelöschtes Entity keine Beziehungen eingehen kann.

Für versionsbehaftete Beziehungen soll gelten:

(1) Jede Änderung eines Attributwertes einer Beziehung soll dokumentiert werden.

(2) Eine gelöschte Beziehung soll dokumentiert bleiben, unabhängig davon, ob sie direkt oder über ein gelöschtes Entity der Aggregation gelöscht wurde.

(3) Vergangene Versionen können sich auf nicht mehr existente Entities beziehen.

(4) Zukünftige Versionen können sich nur auf bereits existierende Entities oder bereits bekannte, zukünftig existierende Entities beziehen.

Für versionsbehaftete Generalisierungen soll gelten:

(1) Ist der generische Entitytyp einer Generalisierung versionsbehaftet, so sind auch alle Subtypen versionsbehaftet.

(2) Ein einzelner Subtyp allein kann nicht versionsbehaftet sein.

(3) Es gelten die gleichen Regeln wie bei versionsbehafteten Entitytypen.

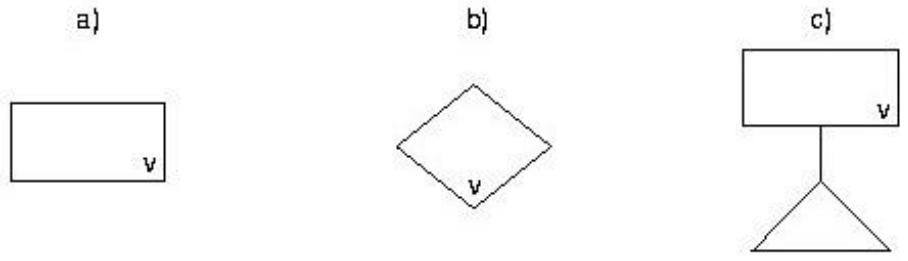


Abb. 64: Versionsbehafteter Entitytyp (a), Beziehungstyp (b) und Generalisierung (c)

6.6 Clusterbildung und komplexe Objekte

Cluster und komplexe Objekte werden durch ein doppelumrahmtes Rechteck dargestellt. Alle Objektarten können in einem Cluster zusammengefaßt werden, d. h., daß auch Cluster in anderen Clustern enthalten sein können. Cluster erhalten einen Clusternamen, der mit einem enthaltenen Entitytypname übereinstimmen kann. Beziehungskanten zu einem Cluster müssen mit dem Namen des referenzierten Objektes, das in dem Cluster enthalten ist, beschriftet sein. Abbildung 65 a zeigt eine Geometriedarstellung nach dem Flächenmodell (Eberlein 84, S. 72; Scheer 90f, S. 266), die als Geometrie-Cluster eine

Beziehung zu der Teiledarstellung in Abbildung 65 b eingeht. Die Teiledarstellung ihrerseits sowie die Ressourcendarstellung aus Abbildung 65 c gehen als Cluster in das Beispiel des Fertigungsbereichs in Abbildung 65 d ein. Die Beziehungskante von dem Beziehungstyp *Bedarf* zu dem Cluster *Ressource* ist nicht beschriftet, da der Clustername und der Name des referenzierten Entitytyps identisch sind.

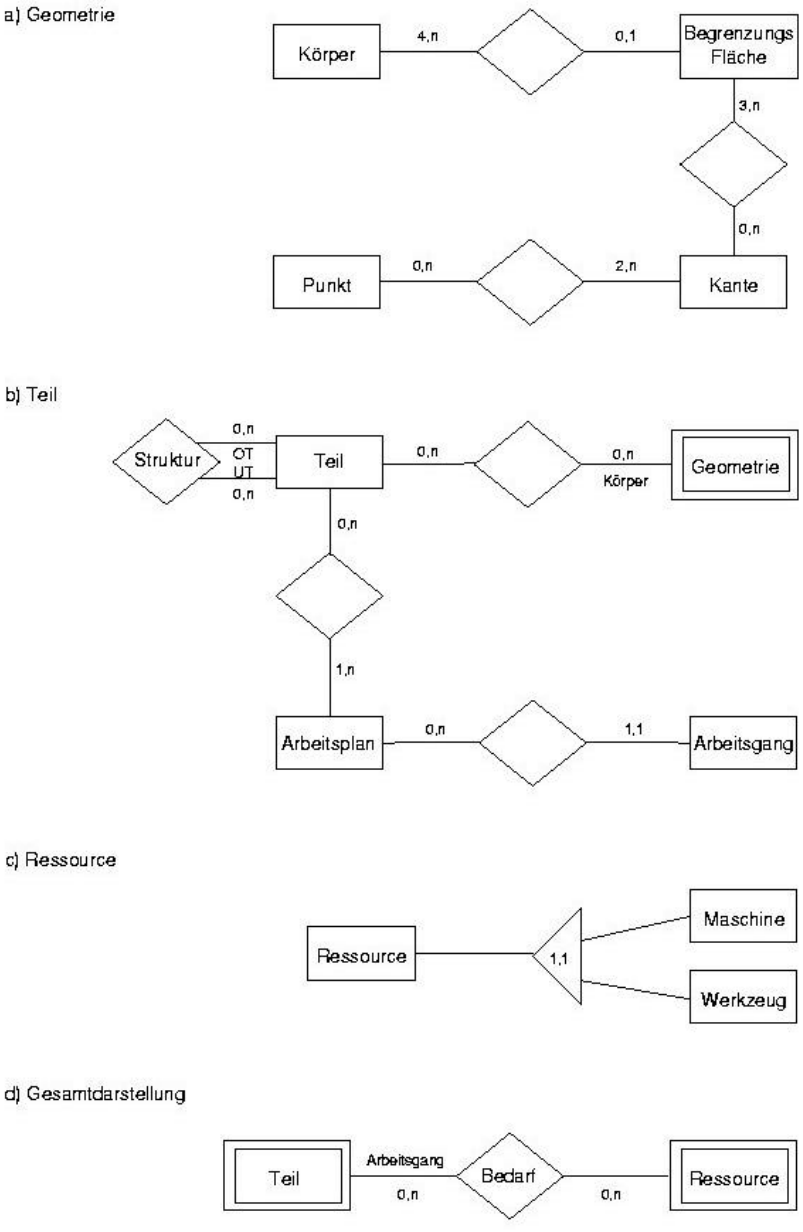


Abb. 65: Beispiel aus dem Fertigungsbereich mit Clusterdarstellung

6.7 Semantische Integritätsbedingungen

Objekttypinterne Integritätsbedingungen

Objekttypinterne Integritätsbedingungen werden durch ein an den Objekttyp angehängtes oder in den Objekttyp eingefügtes offenes (bzw. spitze Klammern) dargestellt. Die Integritätsbedingungen werden in einer Syntax formuliert, deren Notation in einer Backus-Noraml-Form spezifiziert ist (Hopcroft/Ullmann 69). Bezüglich der Sprachelemente ist die Definition an die Datenbanksprache SQL angelehnt (ISO 89). Abbildung 66 zeigt die Syntax der Integritätsbedingungen, bei der folgende Konvention genutzt wurde:

- Konstruktionselemente des Expanded ERM (z. B. das Attribut) werden kursiv dargestellt.

- Sequenzen mehrerer Sprachkonstrukte sind in geschweifte Klammern eingeschlossen ({}).

- Nichtelementare Sprachkonstrukte werden in spitzen Klammern (<>) wiedergegeben und an anderer Stelle genauer spezifiziert.

- Optional einsetzbare Sprachkonstrukte werden in eckige Klammern ([]) eingeschlossen.

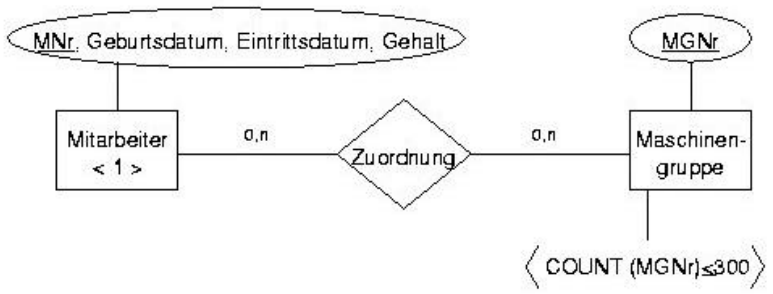
- Aufzählungen von Sprachkonstrukten, von denen genau ein Element eingesetzt werden muß, sind durch Striche getrennt (|).

- Optionale Wiederholungsgruppen werden durch drei Punkte (...) ausgedrückt.

Abbildung 67 zeigt die Entitytypen *Mitarbeiter* und *Maschinengruppe*, die durch eine Zuordnung verbunden sind. Der Typ *Maschinengruppe* hat eine objektmengenabhängige Integritätsbedingung, so daß maximal 300 Entities existieren dürfen. Die Bedingung ist durch ein offenes Hexagon an den Entitytyp angefügt. Der Typ *Mitarbeiter* hat drei Integritätsbedingungen. Da die Angaben der Bedingungen die Graphik sprengen würden, verweist die Nummer in dem eingefügten Hexagon auf eine separate Darstellung. Die einzelnen Bedingungen bedeuten, daß das *Geburtsdatum* kleiner als das *Eintrittsdatum* (<1.1>), das *Gehalt* größer als 1.500 sein muß (<1.2>) und das Durchschnittsgehalt aller Mitarbeiter unter 5.000 liegen muß (<1.3>). Objekttypinterne Integritätsbedingungen können auch für Beziehungen definiert werden.

```
<Domäne> ::=              Attribut {<Vergleichsop.> Wert | = <Enum-Typ> }

<Vergleichsop.> ::=       < | > | = | ≤ | ≥ | ≠

<Enum-Typ> ::=            Wert [{,Wert}...]

<Abgeleitetes Att.> ::=   Attribut = {Attribut | <Pseudoatt.>} <Verknüpf.>
                          {Attribut | <Pseudoatt.> | Wert }
                          [<Verknüpf.> {Attribut | <Pseudoatt.> | Wert} ...]

<Verknüpf.> ::=           + | - | x | / | ...

<Pseudoatt.> ::=          Aktuelles Datum | Aktuelle Uhrzeit

<Bedingtes Att.> ::=      <Domäne> ⇒ <Domäne>

<wechsels. abh.Att.> ::= {Attribut <Vergleichsop.> Wert ⇔
                          Attribut <Vergleichsop.> Wert} |
                          {Attribut <Vergleichsop.> Attribut}

<Obj.-Teilmeng-Abh.> ::= Attribut  = <Funktion> :Attribut WHERE
                          Attribut <Vergleichsop.> :Attribut
                          [{{AND|OR} Attribut <Vergleichsop.>
                          :Attribut} ...]

<Objektmengenkard.> ::= {(<Funktion> (Attribut <Vergleichsop.> Wert)) |
                          (COUNT <Vergleichsop.> Wert)}

<Funktion> ::=            SUM | AVG | MIN | MAX

<Bezieh-.abgel.Att.> ::= Attribut = COUNT(Beziehungstyp)

<Bezieh.-Pfadkompl.> ::= Komplexität(min,max)

<Rekursionsbed.> ::=      Rekursion(tiefe)
```

Abb. 66: Syntaxdefinition der Integritätsbedingungen



< 1 > Mitarbeiter

< 1.1 > Geburtsdatum < Eintrittsdatum

< 1.2 > Gehalt > 1500

< 1.3 > AVG (Gehalt) < 5000

Abb. 67: Beispiel objekttypinterner Integritätsbedingungen

Beziehungstypabhängige Integritätsbedingungen

Beziehungstypabhängige Integritätsbedingungen werden durch verschiedene Konzepte dargestellt. Ebenso wie objekttypinterne Integritätsbedingungen werden auch beziehungsabgeleitete Attribute mit Hilfe der vorgestellten Syntax (vgl. Abbildung 66) definiert. Abbildung 68 zeigt das Beispiel des beziehungsabgeleiteten Attributs *Maschinenzahl* des Entitytyps *Kostenstellen*.

Abhängigkeiten zwischen Beziehungen eines Entitytyps lassen sich zum Teil über das Dreieck-Symbol als Beziehungsalternative mit einer entsprechenden Komplexitätsangabe darstellen. Dabei wird die Komplexitätsangabe so aufgeteilt, daß der (min)-Wert für alle Beziehungen, der (max)-Wert dagegen für jede einzelne Beziehung gilt (vgl. Abbildung 69 a).

Sind sowohl der (min)-Wert als auch alle (max)-Werte gleich 1, so entspricht dies dem Fall (2.1.1.1) der abhängigen Beziehungen (s. Kapitel "Beziehungstypabhängige Integritätsbedingungen", S. 76), ein (min)-Wert von null und alle (max)-Werte gleich 1 entsprechen Fall (2.1.1.2), ein (min)-Wert von null und mindestens ein (max)-Wert größer eins entsprechen Fall (2.1.1.3) und ein (min)-Wert von größer oder gleich eins und mindestens einem (max)-Wert größer eins entsprechen Fall (2.1.1.4). Für den Fall (2.1.1.4) kann der (min)-Wert größer oder gleich eins sein und mindestens ein (max)-Wert größer eins. Dabei darf der (min)-Wert maximal die Größe des kleinsten (max)-Wertes annehmen.

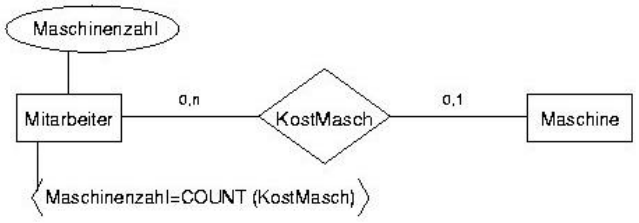


Abb. 68: Beispiel eines beziehungsabgeleiteten Attributs

Im Fall (2.1.1.5) gilt, daß der Entitytyp beliebig viele Beziehungen pro Beziehungstyp eingehen kann. Es wird nur gefordert, daß bei zwei Beziehungstypen mindestens eine Beziehung existiert. Daraus folgt, daß der (min)-Wert bei "n" Beziehungstypen Werte zwischen 1 und (n-1) annehmen kann. Zur Unterscheidung zu den anderen Fällen wird der (min)-Wert in das Dreiecksymbol gestellt. Abbildung 69 b zeigt Fall (2.1.1.5) für zwei Beziehungstypen.

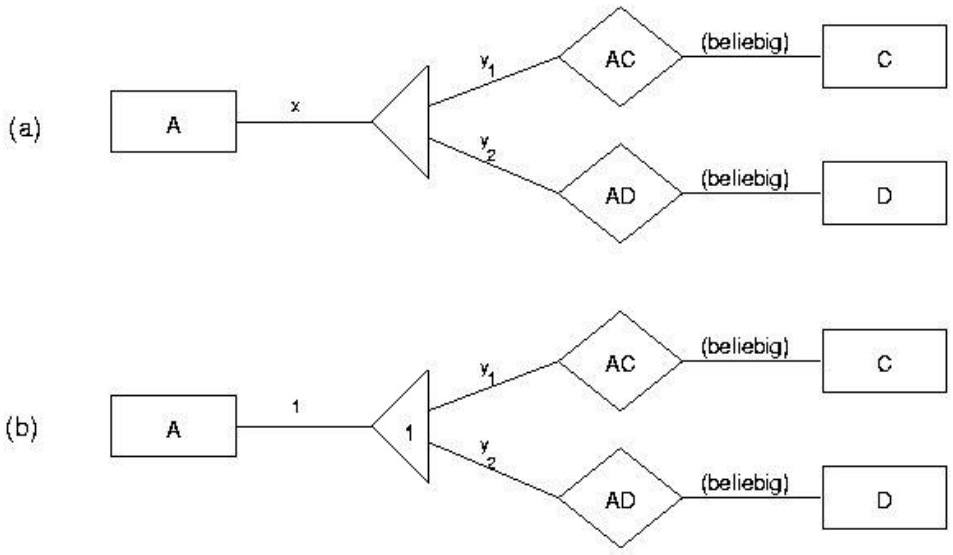


Abb. 69: Beispiele beziehungstypabhängiger Integritätsbedingungen

Alle anderen beziehungstypabhängigen Integritätsbedingungen werden durch eine Darstellung auf Objektebene formuliert, die sich an das Occurence Structure Concept von Tabourier et al. anlehnt (Tabourier 83, Tabourier/Nanci 83). Bei dieser Darstellung, die als **Relationship-Constraint-Diagramm** (**RC**) bezeichnet werden soll, werden die Bedingungen

an einzelnen Objekten, d. h. an Entities und deren Beziehungen verdeutlicht. Für ein Relationship-Constraint-Diagramm gelten folgende Regeln:

(1) In der PERM-Darstellung wird mit einem Hexagon an dem die Bedingung auslösenden Entitytyp oder Beziehungstyp auf die RC-Darstellung verwiesen.

(2) Alle an der zu formulierenden Bedingung beteiligten Objekttypen, wie Entitytypen und Beziehungstypen einschließlich der Verbindungskanten, sowie Generalisierungen, werden als Objekt in die RC-Darstellung übernommen.

(3) Die Namen der Objekttypen und Rollen werden in die RC-Darstellung übernommen.

(4) Ein Objekttyp kann auf Objektebene mehrmals in einem RC-Diagramm vorkommen. Sind dabei unterschiedliche Objekte gemeint, so sind diese durch einen in runde Klammern gesetzten Index zu unterscheiden.

(5) Die Objektverbindungskanten können gerichtet sein. Dies wird durch Pfeile bei den Kanten verdeutlicht. Die Integritätsbedingung gilt dann entsprechend nur in der angegeben Richtung.

(6) RC-Diagramme können durch Operatoren ergänzt werden: Der Operator **NICHT** besagt, daß der abgebildete Sachverhalt den Integritätsbedingungen widerspricht. Der Operator **ODER** läßt nur einen von zwei Zuständen zu. Der Operator Þ besagt, daß für das Auftreten des ersten Zustandes der zweite Zustand Voraussetzung ist. Sind mehrere Zustände Voraussetzung für das Eintreten des ersten Zustandes, so werden die notwendigen Zustände mit dem Operator **UND** verbunden. Bei dem Operator Û sind beide Zustände voneinander abhängig.

(7) Die Objekte des RC-Diagramms werden mit den gleichen graphischen Symbolen wie die entsprechenden Objekttypen im PERM-Diagramm, allerdings zur besseren Unterscheidung mit gestrichelten Linien dargestellt.

(8) In den RC-Diagrammen können Integritätsbedingungen entsprechend der in Abbildung 66 gegebenen Syntaxdefinition formuliert werden.

(9) Komplexitätsrestriktionen über mehrere Beziehungen hinweg werden im RC-Diagramm durch den Ausdruck **Komplexität (min,max)** dargestellt.

(10) Rekursive Beziehungen können über den Ausdruck **Rekursion (max)** in einem RC-Diagramm dargestellt werden, wobei über (max) eine maximale Rekursionstiefe definiert werden kann.

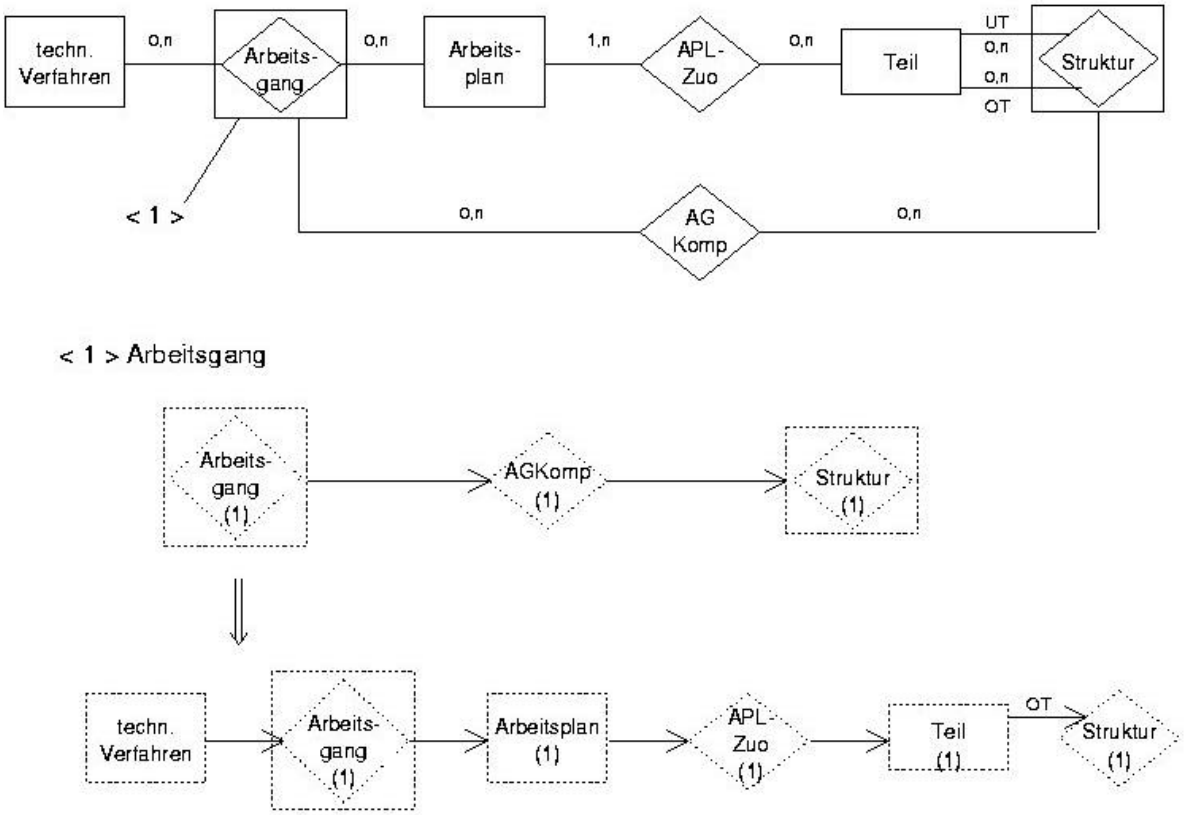


Abb. 70: RC-Diagramm einer Bedingung zwischen zwei Beziehungspfaden

Abbildung 70 zeigt das Stücklisten-Arbeitsplanbeispiel aus Abbildung 54. In dem PERM-Diagramm ist der Typ *Arbeitsgang* mit einer Integritätsbedingung gekennzeichnet, die auf das RC-Diagramm <1> verweist, in der die oben beschriebenen Beziehungsabhängigkeiten definiert sind.

Das Beispiel einer Komplexitätsbedingung über mehrere Beziehungen hinweg aus Abbildung 53 ist mit Hilfe eines RC-Diagramms in Abbildung 71 dargestellt. Die Angabe Komplexität (0,3) besagt, daß ein Entity des ersten dargestellten Typs (*Wartungsauftrag*) mit mindestens null und maximal drei Entities des letzten dargestellten Typs (*Kostenstelle*) verbunden ist.

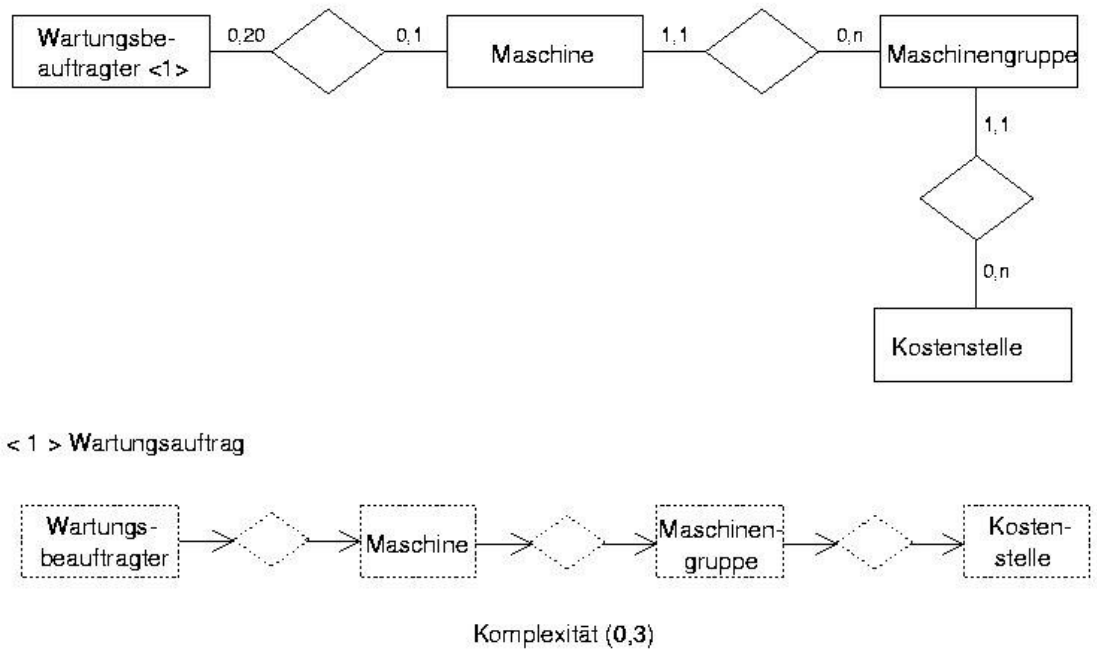


Abb. 71: RC-Diagramm einer Komplexität über mehrere Beziehungen

Objektausprägungsabhängige Beziehungen, wie sie anhand des Stücklistenbeispiels in Abbildung 48 diskutiert wurden sowie der Ausschluß rekursiver Beziehungen sind in Abbildung 72 dargestellt. Das PERM-Diagramm zeigt einen Entitytyp *Teil* mit einer Stücklistenstruktur über den Typ *Struktur*, der einen Verweis auf Integritätsbedingungen enthält. Die Generalisierung des Typs *Teil* zeigt durch die Kennzeichnung (1,1) an, daß ein Entity *Teil* in genau einem Subtyp *Baugruppe*, *Fremdteil* oder *Endprodukt* enthalten sein muß.

Die Bedingung <1.1> des RC-Diagramms schließt aus, daß ein Entity *Teil* direkt oder indirekt über beliebig viele Stücklistenstufen mit sich selbst verbunden ist. Bedingung <1.2> verbietet, daß ein Entity *Teil*, das dem Subtyp *Fremdteil* angehört, eine Strukturbeziehung über die Rolle *OT* (Oberteil) eingeht, analog verhindert Bedingung <1.3> eine Beziehung mit der Rolle *UT* (Unterteil) für Endprodukte. Abbildung 73 gibt den gleichen Sachverhalt ohne eine Generalisierungsstruktur wieder. Der Typ *Teil* enthält ein Attribut *Teileart*, das laut Bedingung <2> als Enumerationstyp die Werte *Baugruppe*, *Fremdteil* und *Endprodukt* annehmen kann. Entsprechend sind die Bedingungen <1.2> und <1.3> nicht über einen Subtyp, sondern über eine Attributausprägung spezifiziert.

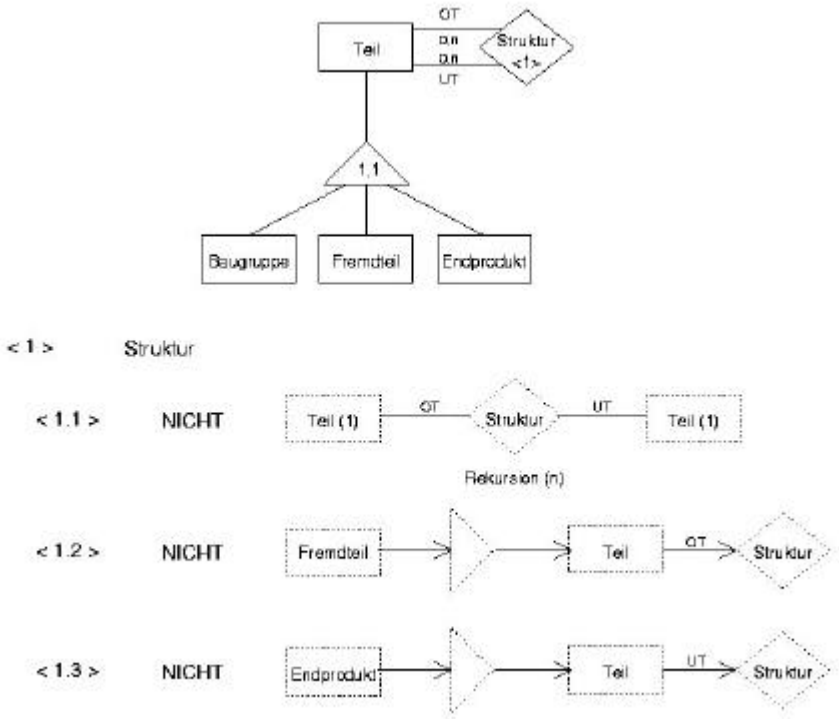


Abb. 72: Stücklistenbeispiel mit Integritätsbedingung, Beispiel 1

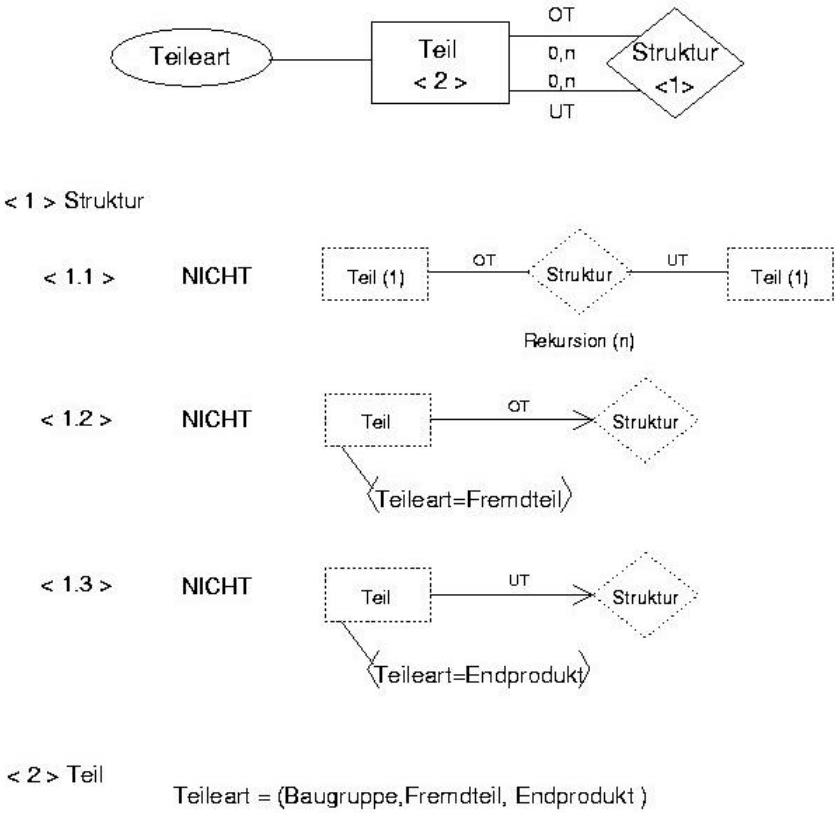


Abb. 73: Stücklistenbeispiel mit Integritätsbedingung, Beispiel 2

Das in Abbildung 74 dargestellte PERM-Diagramm zeigt eine MitarbeiterProjektzuordnung *Arbeit*, die angibt, daß ein *Mitarbeiter* in mehreren *Projekt*en arbeiten und in einem *Projekt* mehrere *Mitarbeiter* arbeiten können.

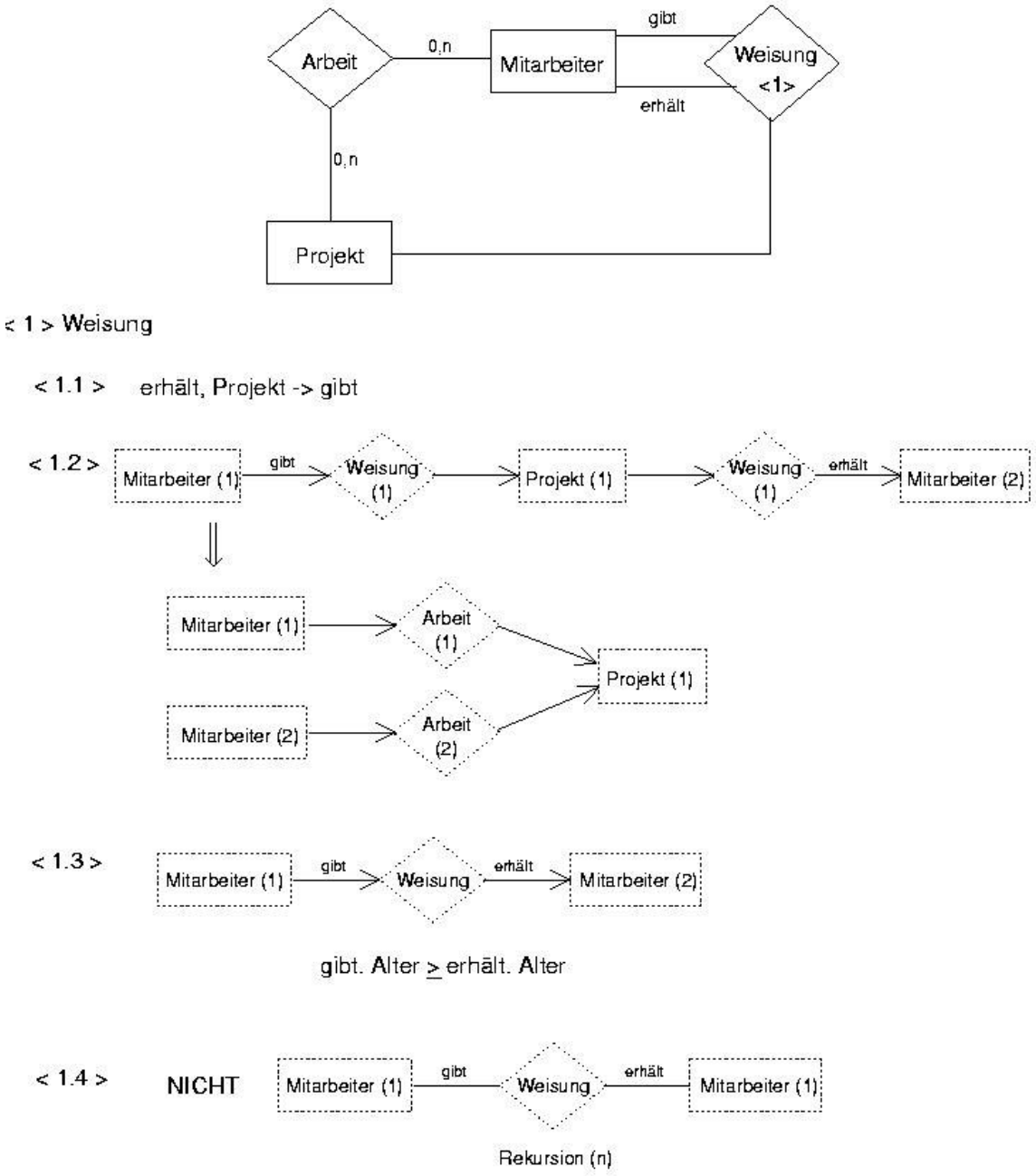


Abb. 74: Integritätsbedingung an einem Mitarbeiter_projekt-BEispiel

Der Beziehungstyp *Weisung* regelt, welcher Mitarbeiter in ein Projekt von einem anderen Mitarbeiter Anweisung erhalten kann. Dabei gelten die folgenden Bedingungen:

<1.1> Ein Mitarbeiter kann in einem Projekt nur von einem anderen Mitarbeiter Weisungen erhalten. Dieser kann aber seinerseits mehreren Mitarbeitern in dem Projekt Weisungen erteilen. Es ergibt sich die funktionale Abhängigkeit:

Mitarbeiter-erhält, Projekt → Mitarbeiter-gibt.

<1.2> Besteht zwischen zwei Mitarbeitern in einem Projekt eine Weisungshierarchie, so müssen auch beide Mitarbeiter in diesem Projekt arbeiten.

<1.3> Die Weisungshierarchie ist altersabhängig in dem Sinn, daß ein Weisungsberechtigter nicht jünger als ein Weisungsempfänger sein darf.

<1.4> Ein Zyklus in der Weisungshierarchie wird explizit ausgeschlossen, da dies sonst trotz der Bedingungen <1.2> und <1.3> bei mehreren gleichaltrigen Mitarbeitern in einem Projekt erfolgen könnte.

6.8 Metamodell

Das Expanded Entity-Relationship-Modell besitzt Sprachelemente, die nach den oben definierten Regeln zueinander in Beziehung stehen. Diese Strukturbeziehungen sollen mit einer geeigneten Beschreibungssprache dargestellt werden. Dazu eignet sich das Expanded Entity-Relationship-Modell selbst, d. h. die Strukturen des Expanded Entity-Relationship-Modells werden in einem PERM-Diagramm aufgezeigt. Ein solches Diagramm wird als Metamodell bezeichnet (vgl. auch Lenz 90). Scheer hat das Metamodell eines erweiterten ERM (Scheer 90c, S. 12 ff), Marti das Metamodell für das Entitäten-Diagramm (Marti 83, S. 382) dargestellt. Wird ein Data Dictionary eines Datenbanksystems mit dem eigenen Datenmodell realisiert, z. B. bei dem Relationenmodell und beim RM/T (Codd 79, S. 424), so ist das ein Metamodell.

Das Metamodell des PERM ist in Abbildung 75 dargestellt. Die Strukturelemente Entitytyp und Beziehungstyp werden auf der Metaebene durch die beiden Entitytypen *ETyp* und *BTyp* ausgedrückt. Beziehungen können nur als Verbindung mindestens zweier Entitytypen existieren. Deshalb muß ein *BTyp* einen Beziehungstyp *Kante* zum *ETyp* mit einer Komplexität von (2,n) eingehen. Ein *ETyp* muß nicht unbedingt Beziehungen eingehen und erhält deshalb die Komplexität von (0,n).

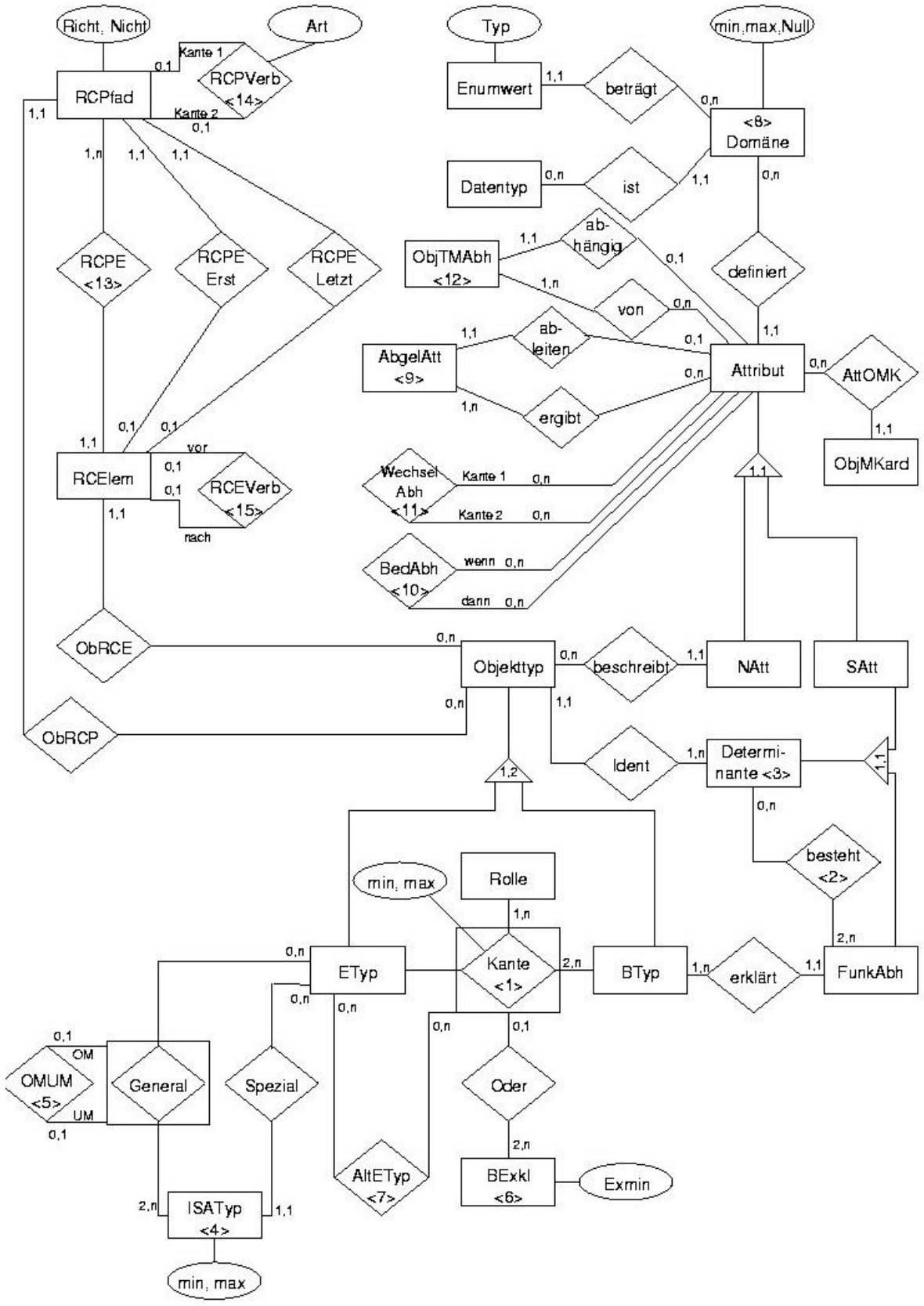


Abb.75-1: Metamodell des Expanded Entity-Relationship-Modells

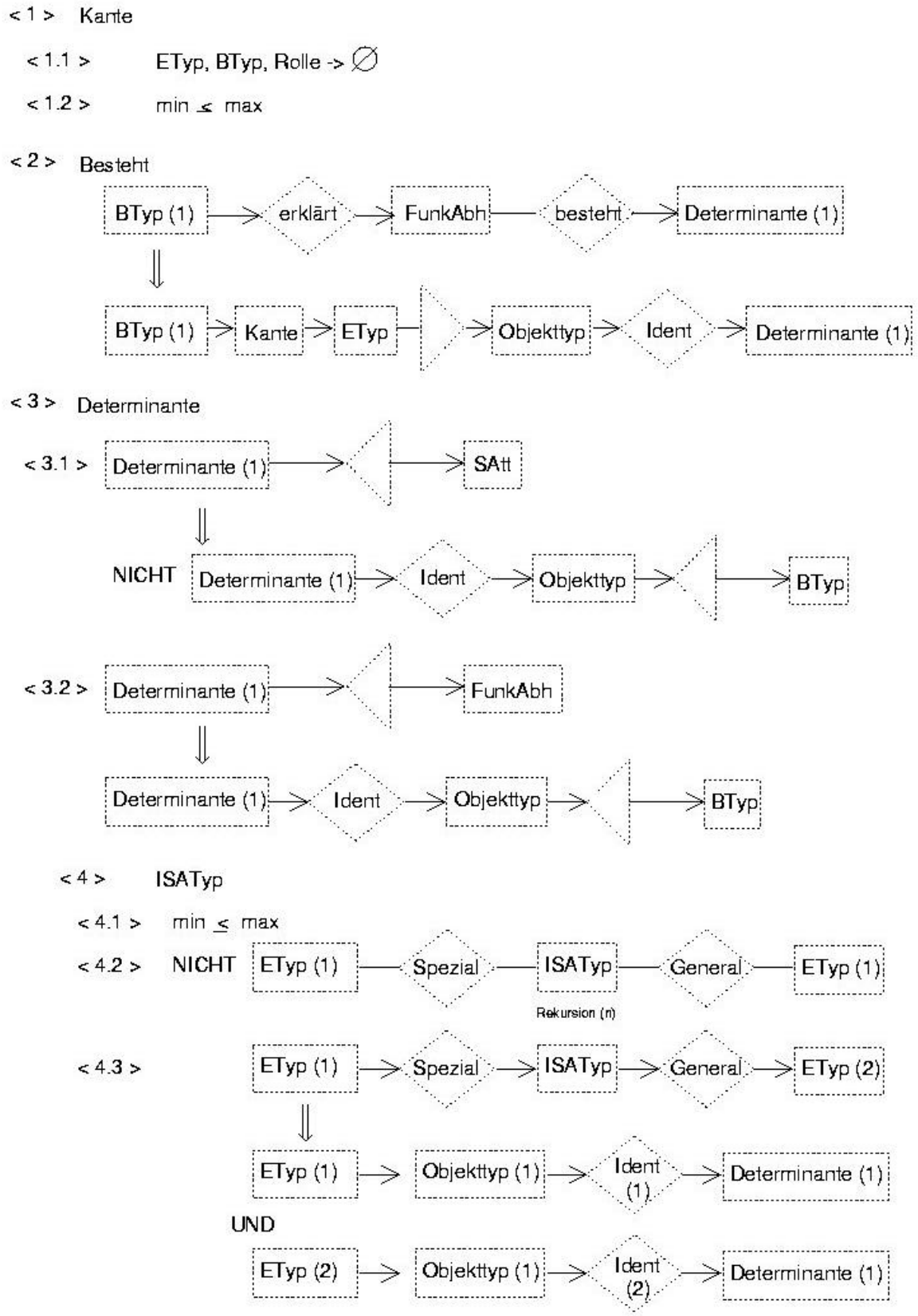


Abb. 75-2: Metamodell des Expanded Entity-Relationship-Modells

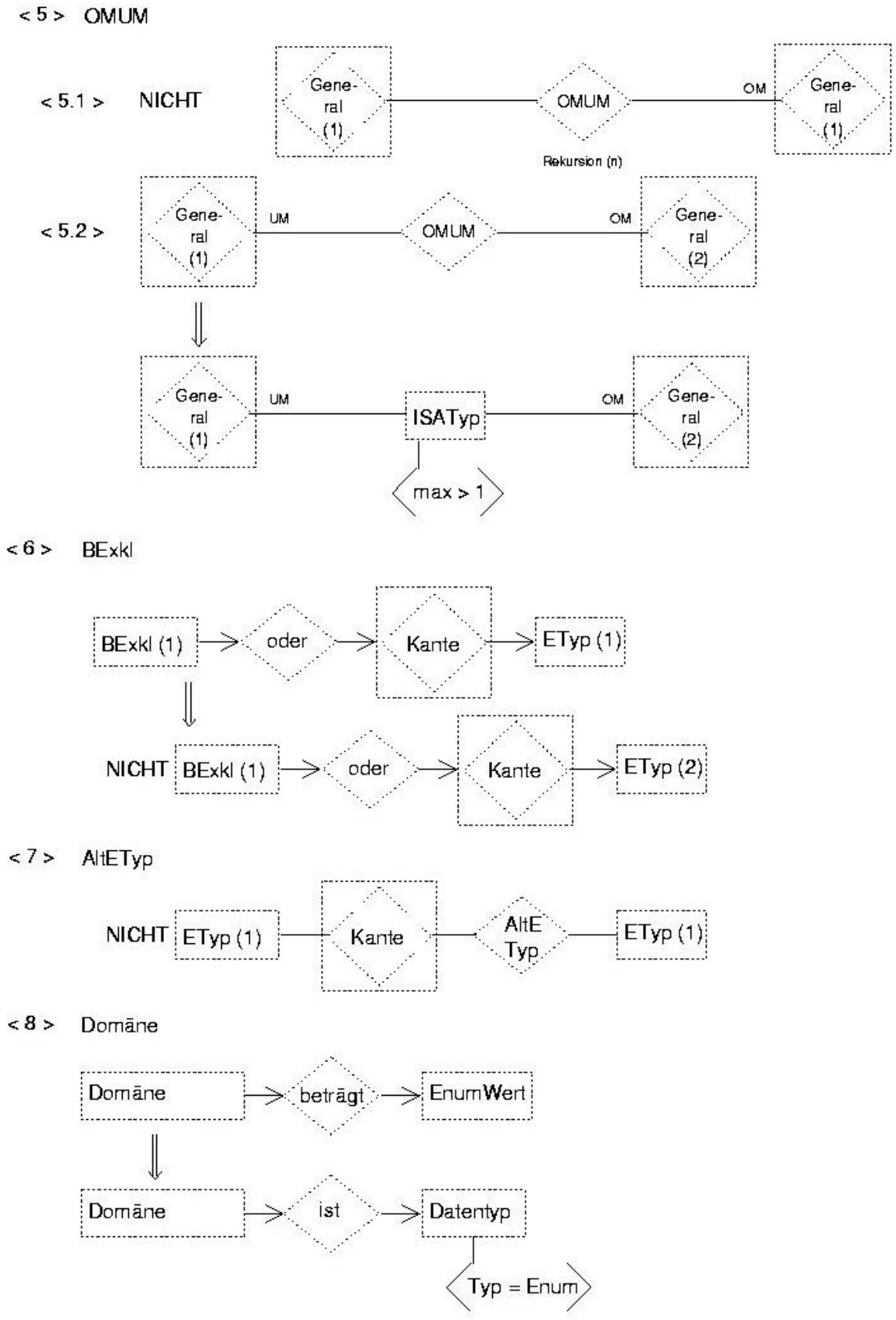


Abb. 75-3: Metamodell Modells des Expanded Entity-Relationship-Modells

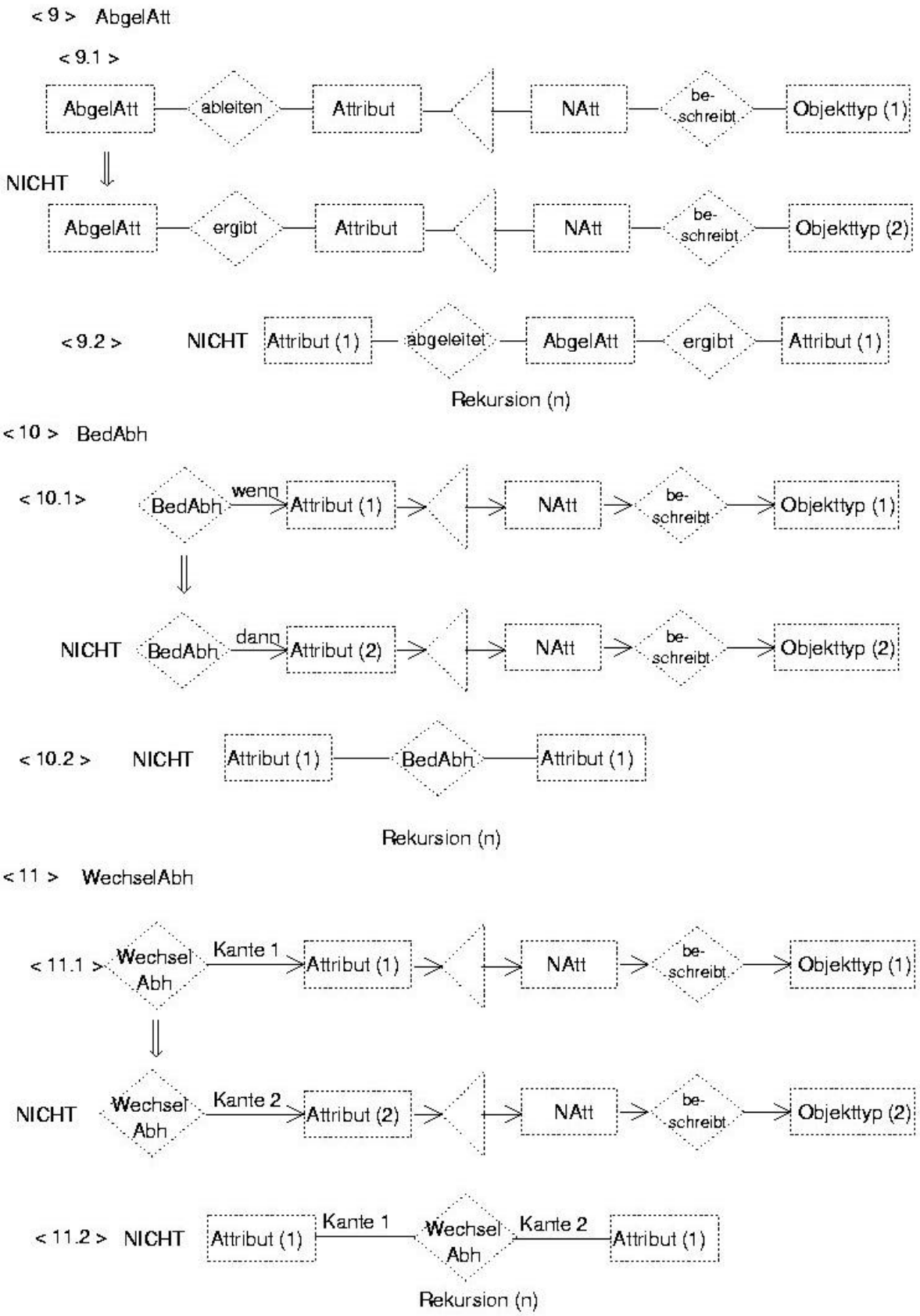


Abb. 75-4: Metamodell des Expanded Entity-Relationship-Modells

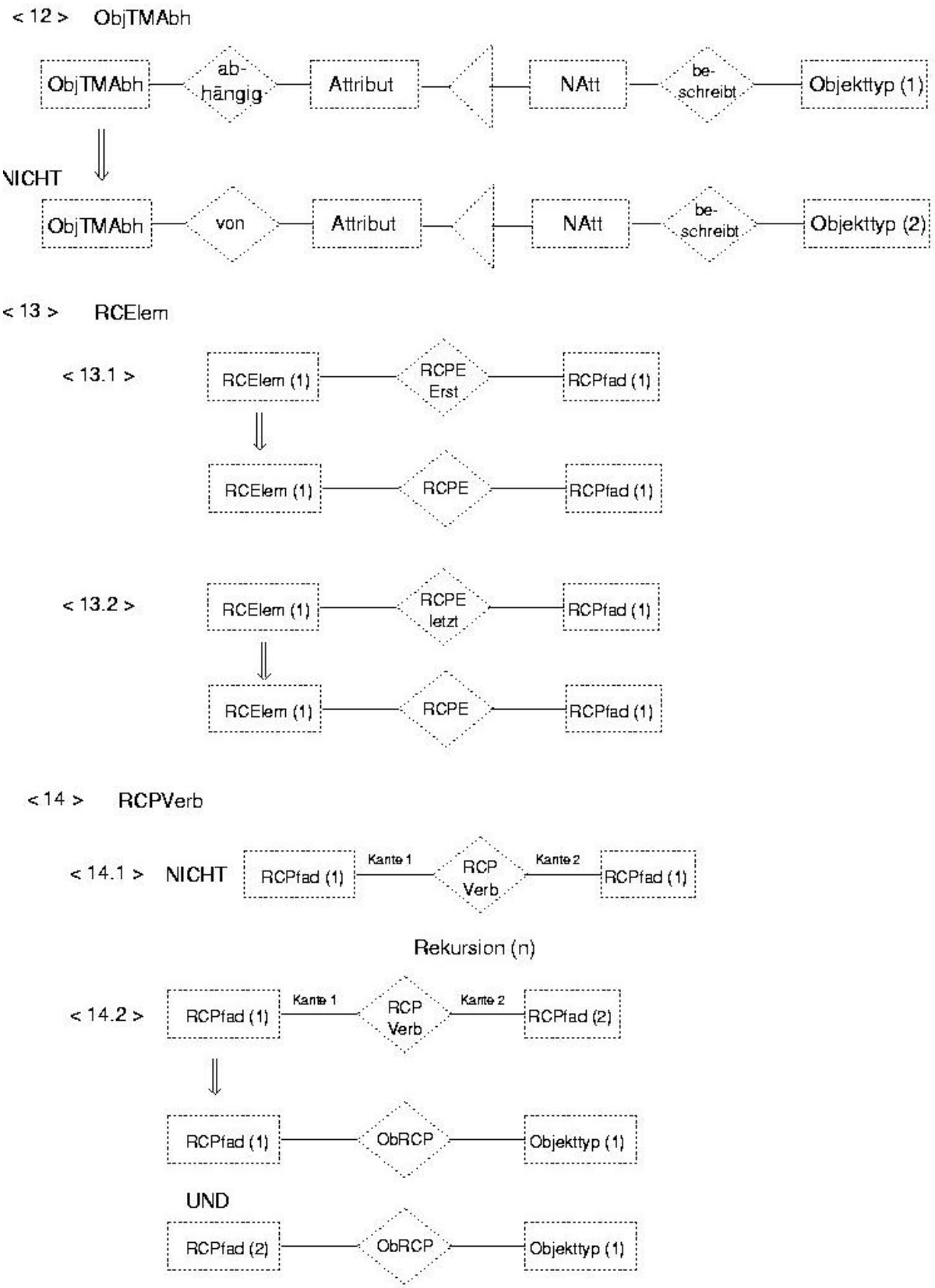


Abb. 75-5: Metamodell des Expanded Entity-Relationship-Modells

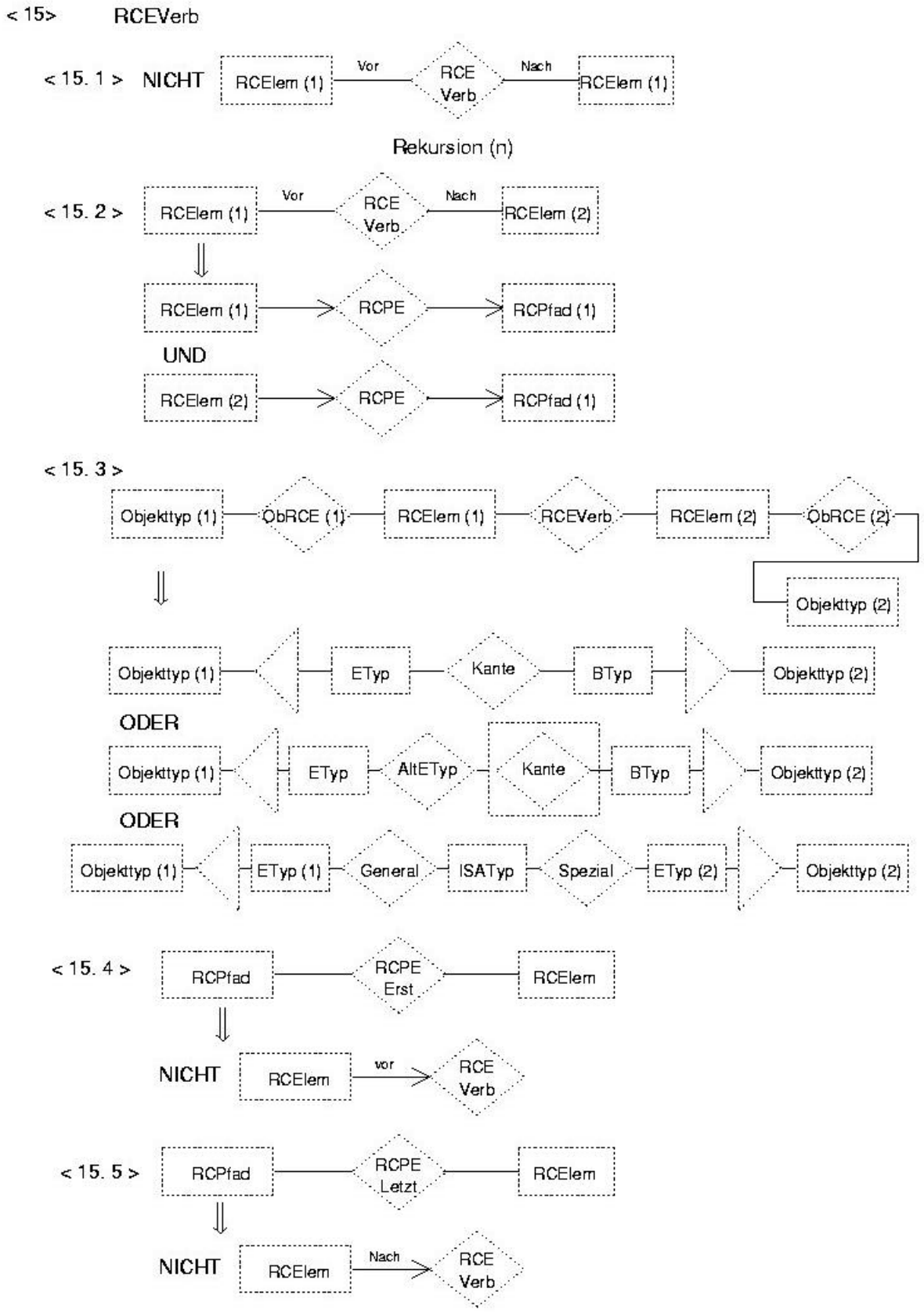


Abb. 75-6: Metamodell des Expanded Entity-Relationship-Modells

Da ein Entitytyp mehrfach in einem Beziehungstyp enthalten sein kann, wird zusätzlich der Entitytyp *Rolle* als Rollennamen zur eindeutigen Identifizierung des Beziehungstyps *Kante* benötigt. Der Komplexitätsgrad (1,n) besagt, daß ein definierter Rollenname auch einer Beziehung zugeordnet sein muß. Gleichzeitig ist aber zugelassen, daß ein Rollenname mehrmals in einem Modell vorhanden sein kann. Die funktionalen Abhängigkeiten des Beziehungstyps *Kante* sind ($BTyp,ETyp,Rolle \leftrightarrow \perp$). Die Verbindung *ETyp* zu *Kante* erhält explizit keine Komplexitätsangabe, da diese aus der Bedingung <1.1> abgeleitet werden kann. Eine *Kante* erhält die Attribute *min* und *max* zur Komplexitätsangabe, für die über Bedingung <1.2> sichergestellt wird, daß der (min)-Wert nicht größer als der (max)-Wert werden kann.

ETyp und *BTyp* sind beides Objekttypen und werden zu diesen in der Weise generalisiert, daß ein *Objekttyp* entweder ein *BTyp* oder ein *ETyp* oder im Falle einer uminterpretierten Beziehung sowohl *ETyp* als auch *BTyp* sein kann. Ein *Objekttyp* kann durch mehrere Nichtschlüsselattribute (*NAtt*) beschrieben werden, wobei ein Nichtschlüsselattribut aufgrund der Redundanzvermeidung nur zu genau einem Objekttyp gehört. Nichtschlüsselattribute bilden mit der zweiten Teilmenge Schlüsselattribute (*SAtt*) die vollständige und disjunkte Generalisierung *Attribut*. Diese müssen genau einer Domäne (Entitytyp *Domäne*) angehören, die den Wertebereich des Attributs definiert.

Ein Objekttyp wird genau durch eine Determinante (Entitytyp *Determinante*) identifiziert, eine Determinante kann mehrere Objekttypen, z. B. bei Generalisierungen, identifizieren (Beziehungstyp *Ident*). Funktionale Abhängigkeiten (Entitytyp *FunkAbh*) werden benötigt, um diese Beziehung zu erklären. Ein *BTyp* muß durch mindestens eine, kann aber auch durch mehrere Abhängigkeiten erklärt werden. Eine funktionale Abhängigkeit besteht aus genau den Determinanten der in dem erklärten *BTyp* aggregierten Entitytypen. Dies wird durch den Beziehungstyp *besteht* und die Integritätsbedingung <2> ausgedrückt. Eine einen Objekttyp identifizierende Determinante ist bei einem Kern-Entity ein Schlüsselattribut, bei allen Beziehungstypen, oder aus Beziehungstypen uminterpretierten Entitytypen, eine den Beziehungstyp erklärende funktionale Abhängigkeit. Deshalb stellt der Entitytyp *Determinante* eine aus den Entitytypen *SAtt* und *FunkAbh* vollständige und disjunkte Generalisierung dar und ist mit der Integritätsbedingung <3> spezifiziert.

Eine Generalisierung wird durch den Entitytyp *ISA-Typ* dargestellt. Ein ISA-Element spezialisiert über den Beziehungstyp *Spezial* genau einen *ETyp*, der die Obermenge repräsentiert, sowie generalisiert mindestens zwei *ETyp*en als Teilmengen der Generalisierung über den Beziehungstyp *General*. Der *ISA-Typ* hat die Attribute *min* und *max*, mit der die Vollständigkeit und Disjunktion der Generalisierung beschrieben werden.

Da ein *ETyp* sich nicht selbst generalisieren kann, wird dies in der Integritätsbedingung <4> ausgeschlossen. Durch die Integritätsbedingung wird auch definiert, daß alle an der Generalisierung beteiligten *ETyp*en die gleiche Determinante besitzen. Bei disjunkten Teilmengen einer Generalisierung können zwei Teilmengen in einem Unter- und Obermengenverhältnis zueinander stehen. Dies wird durch den Beziehungstyp *OMUM* zwischen zwei zum Entitytyp uminterpretierten Beziehungen *General* mit den Kanten *OM* und *UM* und die Integritätsbedingung <5> ausgedrückt.

Die Möglichkeit, daß die Beziehungen eines Entitytyps für ein Entity jeweils exklusiv sind, wird durch den Entitytyp *BExkl*, der über den Beziehungstyp *Oder* mit mindestens zwei Kanten desselben *ETyp*s verbunden ist, ausgedrückt. Die Anzahl der Beziehungen, für die die Exklusivität mindestens gilt, ist im Attribut *Exmin* definiert. Über Bedingung <6> wird sichergestellt, daß exklusive Beziehungen immer den gleichen Entitytyp betreffen.

Beziehungen mit alternativen Entitytypen werden durch den Beziehungstyp *AltETyp* ausgedrückt. Einer der alternativen *ETyp*en wird über den Beziehungstyp *Kante* mit dem *BTyp* verbunden. An der *Kante* wird die Komplexitätsangabe vermerkt. Die restlichen alternativen *ETyp*en gehen mit dieser *Kante* einen Beziehungstyp *AltETyp* ein. Die Komplexität von *Kante* zu *AltETyp* ist (0,n), da eine Beziehung keine oder auch mehrere alternative *ETyp*en haben kann. Aus Sicht des *ETyp*s ist die Komplexität ebenfalls (0,n), da ein *ETyp* in beliebig vielen *BTyp*en als Alternative auftreten kann. Bedingung <7> schließt aus, daß ein *ETyp* bei einem *BTyp* zu sich selbst eine Alternative ist. Für die Bestimmung der funktionalen Abhängigkeiten eines *BTyp*s wird, durch die Integritätsbedingung <2>, von den möglichen alternativen Entitytypen immer die Determinante des *ETyp*s herangezogen, der über den Beziehungstyp *Kante* direkt mit dem *BTyp* verbunden ist. Um die Anzahl der Beziehungen zu ermitteln, in der ein Entitytyp enthalten ist, müssen die Anzahl der *Kante*n-Beziehungen und die Anzahl der *AltETyp*-Beziehungen addiert werden.

Objekttypinterne Integritätsbedingungen werden an dem Entitytyp *Attribut* dargestellt. Der bereits eingeführte Entitytyp *Domäne* definiert den Wertebereich eines *Attribut*s. Dazu geht der Typ *Domäne* genau einen Beziehungstyp *ist* mit der Komplexität von (1,1) mit dem Entitytyp *Datentyp* ein. Der Typ *Datentyp* kann als Entities Standarddatentypen wie Integer, Floating Point, Number, Date, Character, Boolean, Enum usw. besitzen. Der Typ *Domäne* besitzt die Attribute *min* und *max*, mit denen der durch den *Datentyp* vorgegebene Wertebereich für einzelne *Attribut* eingegrenzt werden kann, sowie das Attribut *Null*, das definiert, ob ein *Attribut* eines Entities auch einen unbestimmten Wert annehmen kann (Codd 79; Wedekind 88). Ist der *Datentyp* einer *Domäne* ein Aufzählungstyp (Enumeration), so geht die *Domäne* für jeden Wert einen Beziehungstyp *beträgt* zu dem Entitytyp *EnumWert*

ein. Die Integritätsbedingung <8> stellt sicher, daß nur *Domänen* von Typ *Enum* Beziehungen zum Typ *EnumWert* besitzen.

Ist der Wert eines Attributs abgeleitet (beziehungsabgeleitetes Attribut), so geht der Entitytyp *Attribut* den Beziehungstyp *ableiten* mit dem Entitytyp *AbgelAtt* (*abgel*eitetes *Att*ribut) ein. Die für die Ableitung notwendigen *Attribut*e gehen über den Beziehungstyp *ergibt* eine Verbindung zu *AbgelAtt* ein, wobei ein *AbgelAtt* mindestens mit einem *Attribut* eine Beziehung eingehen muß (Komplexität (1,n)). Ein *Attribut* kann maximal einmal abgeleitet werden (Komplexität (0,1)), kann aber seinerseits zu Ableitung mehrerer Attribute dienen.

Bedingte Abhängigkeiten zwischen Attributen (bedingtes Attribut) werden über den binären Beziehungstyp *BedAbh* abgebildet, der den Entitytyp *Attribut* rekursiv mit einer Komplexität von (0,n) zu (0,n) verbindet. Die Kante *wenn* geht von dem bedingungsstellenden *Attribut*, die Kante *dann* vom bedingungsabhängigen *Attribut* aus.

Wechselseitig abhängige *Attribut*e werden über den Beziehungstyp *WechselAbh* (*wechsel*seitig *abh*ängige Attribute) mit den Kanten *Kante1* und *Kante2* und der Komplexität von (0,n) zu (0,n) miteinander verbunden.

*Attribut*e, die über *AbgelAtt*, *BedAbh* oder *WechselAbh* miteinander verbunden sind, müssen zum einen zu dem gleichen Objekt gehören und dürfen zum anderen nicht rekursiv auf sich selbst verweisen. Dies wird durch die Bedingungen <9>, <10> und <11> sichergestellt.

Objektteilmengenabhängigkeiten werden in dem Entitytyp *ObjTMAbh* definiert, der über den Beziehungstyp *abhängig*, der auf das abhängige Attribut verweist, sowie über den Beziehungstyp *von*, der auf die bei der Formulierung notwendigen Attribute verweist, mit dem Entitytyp *Attribut* verbunden ist. Die Komplexitätsangaben der Beziehungstypen sind analog zu denen des abgeleiteten Attributs. Integritätsbedingung <12> stellt sicher, daß alle referenzierten Attribute zu dem gleichen Objekt gehören. Im Gegensatz zu den Bedingungen <9>, <10> und <11> sind allerdings aufgrund der Definition der Objektteilmengen-abhängigkeit rekursive Verweise auf das gleiche Attribut zulässig.

Objektmengenkardinalitäten werden als Entitytyp *ObjMKard* über den Beziehungstyp *AttOMK* mit den Typ *Attribut* verbunden.

Integritätsbedingungen, die mit Hilfe eines RC-Diagramms formuliert werden, bilden den Entitytyp *RCPfad* (*R*elationship-*C*onstraint *Pfad*). Der Beziehungstyp *ObRCP* (*Ob*jekt-

*R*elationship-*C*onstraint *P*fad - Zuordnung) besagt, von welchem Objekttyp der Pfad in dem RC-Diagramm initiiert wurde und erhält deshalb aus Sicht des *RCPfad*s eine eindeutige Zuordnung mit der Komplexität (1,1). Ein *RCPfad* besteht aus mindestens einem oder mehreren Elementen, die die Klasse der *RCElem* (*R*elationship-*C*onstraint *Elem*ent) bilden. Alle *RCElem*s sind über den Beziehungstyp *RCPE* genau einem *RCPfad* zugehörig (Komplexität (1,1)). Zusätzlich sind die ersten und letzten *RCElem*s eines Pfads über die Beziehungen *RCPEErst* und *RCPELetzt* mit *RCPfad* verbunden. Bedingung <13> stellt sicher, daß alle über *RCPEErst* oder *RCPELetzt* verbundenen Typen auch über *RCPE* verbunden sind.

Ein *RCElem* entspricht genau einem *ETyp* oder einem *BTyp* eines PERM-Diagramms, weshalb ein Beziehungstyp *ObRCE* (*Ob*jekt-*R*elationship-*C*onstraint *E*lement - Zuordnung) mit einer Komplexität von (1,1) zu *Objekttyp* besteht. *Objekttyp*en können sowohl in einem *RCPfad* als auch in verschiedenen *RCPfad*en vorkommen. Ein *RCPfad* kann gerichtet (Attribut *Richt*) oder verneint (Attribut *Nicht*) sein. Zwei *RCPfad*e können über den Beziehungstyp *RCPVerb* (*R*elationship-*C*onstraint *P*fad - *Verb*indung) miteinander verbunden werden. Das Attribut *Art* bestimmt, ob die Verbindung bsw. eine Oder-Verbindung oder ein Verweis ist. Integritätsbedingung <14.1> schließt rekursive Beziehungen aus. Bedingung <14.2> stellt sicher, daß die verbundenen *RCPfad*e von dem gleichen *Objekttyp* initiiert sind. Die einzelnen Elemente eines Pfads sind über den Beziehungstyp *RCEVerb* miteinander verbunden, wobei jedes *RCElem* maximal einen Vorgänger (Kante *vor* mit Komplexität (0,1)) und maximal einen Nachfolger (Kante *nach* mit Komplexität (0,1)) besitzen kann. Integritätsbedingung <15> stellt sicher, daß die *RCElem* nicht zu einer Schleife verbunden werden (<15.1>), daß alle verbundenen *RCElem* zu dem gleichen *RCPfad* gehören (<15.2>), daß alle korrespondierenden *Objekttyp*en verbundener *RCElem*s auch eine Beziehung (über *Kante* oder *AltETyp*) oder eine Generalisierungsverbindung (über *ISATyp*) zueinander besitzen (<15.3>), und daß das erste bzw. letzte *RCElem* keinen Vorgänger (<15.4>) bzw. keinen Nachfolger (<15.5>) besitzt.

7. Referenzmodell der Fertigung

Mit der Methode des im vorherigen Kapitel entwickelten Expanded Entity-Relationship Modells soll im folgenden ein Referenzmodell für den Fertigungsbereich entworfen werden. Ein Referenzmodell eines Industriebetriebs auf Basis eines erweiterten ERM wurde von Scheer aufgestellt (Scheer 90f). Da es fast alle Funktionsbereiche eines Unternehmens umfaßt, wird es auch als Unternehmensmodell (UDM) bezeichnet (Scheer 88b). Ruffing skizzierte, ebenfalls auf Basis eines erweiterten Entity-Relationship-Modells, die Datenbeziehungen für die Steuerung autonomer Fertigungsinseln (Ruffing 90). Das folgende Referenzmodell lehnt sich teilweise an die genannten Modelle an, geht aber zum einen detaillierter auf die einzelnen Funktionen des Fertigungsbereichs ein und präzisiert zum anderen aufgrund der mächtigeren Semantik des Expanded ERM die möglichen Abbildungszustände der Fertigungsdatenstrukturen.

7.1 Aufbau des Referenzmodells

Die Datenstrukturen der Fertigung lassen sich nach dem **Funktionsbereich**, dem die Daten zugehörig sind, gliedern. Abbildung 76 zeigt in Anlehnung an Scheer die wichtigsten Funktionsbereiche der Fertigung (Scheer 90a, S. 3, 59 und 63). Die Datenströme verdeutlichen vereinfacht die wesentlichsten Beziehungen zwischen den Bereichen, wobei ein Bereich als Datenquelle und ein oder mehrere andere Bereiche als Datensenken agieren. Der Großteil der Daten wird allerdings von allen Bereichen benötigt, was durch den Integrationsgedanken des Computer Integrated Manufacturing verstärkt wird.

Eine weitere Möglichkeit der Gliederung ist die Unterteilung nach **Stamm- und Bewegungsdaten**. Stammdaten sind Informationen, die für einen längeren Zeitraum benötigt werden und deren Löschungstermin in der Regel noch nicht zum Zeitpunkt der Entstehung bekannt ist. Sie werden auch als Grunddaten bezeichnet (Scheer 90f, S. 163). Die Bewegungsdaten werden meistens durch Aktionen und Tätigkeiten des betrieblichen Prozesses erzeugt und nur für einen begrenzten Zeitraum benötigt. Typische Bewegungsdaten sind Kunden-, Fertigungs- oder Wartungsaufträge, Bestellungen und Angebote. Sie beziehen sich in der Regel auf Stammdaten und sind von diesen abhängig (vgl. auch Sinz 88, S. 199). Scheer modelliert die Bewegungsdaten als Aggregation der beteiligten Stammdaten und der Zeit (Scheer 90f, S. 28 und 34). Dadurch erhält die

ansonsten statische Betrachtung der Datenstrukturen eine Zeitdimension, die im folgenden Kapitel näher erläutert wird.

Für die **Gliederung des Referenzmodells** werden zuerst die Stammdaten konstruiert, die für mehrere Bereiche notwendig sind. Darauf aufbauend werden die Datenstrukturen der einzelnen Funktionsbereiche, d. h. sowohl die zusätzlichen Grunddaten als auch die Bewegungsdaten, modelliert. Bezüglich des Detaillierungsgrades werden die Datenstrukturen auf Typebene modelliert. Integritätsbedingungen werden formuliert, soweit damit unsinnige Zustände verhindert werden. Attribute werden nur beispielhaft, bzw. soweit wie für die Formulierung von Integritätsbedingungen notwendig, angegeben. Auf die Kennzeichnung von Datenstrukturen als versionsbehaftete Elemente wurde verzichtet, da sie durch ihre einfache Definition ohne Schwierigkeiten für gewünschte Ausschnitte nachgeführt werden kann.

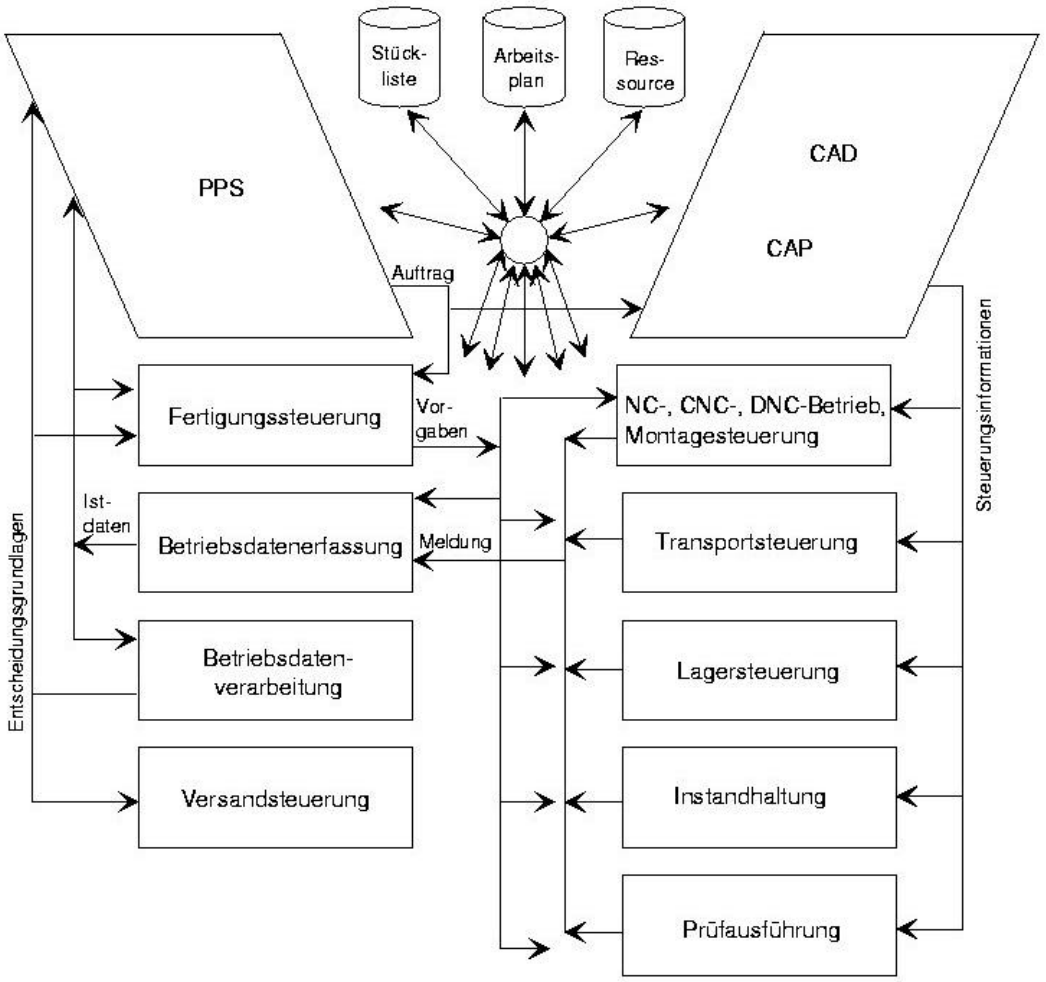


Abb. 76: Funktionsbereich der Fertigung und deren Datenbeziehungen

7.2 Modellierung der Zeit

Abbildung 77 zeigt die Zeitkategorien, die im folgenden für die Modellierung unterschieden werden sollen (vgl. auch Loos 90a).

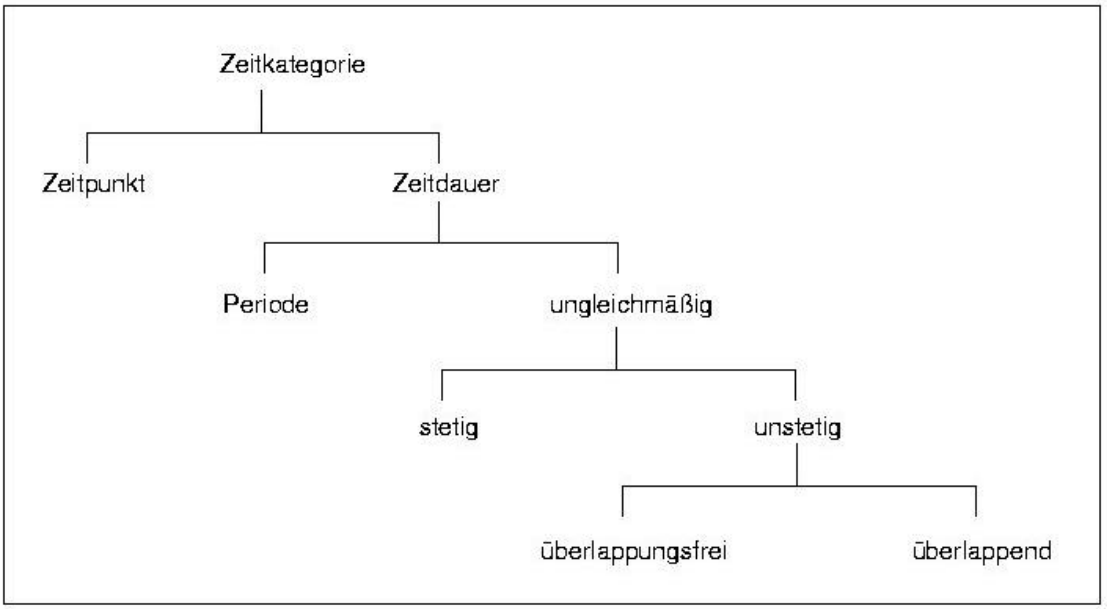


Abb. 77: Modellierbare Zeitkategorien

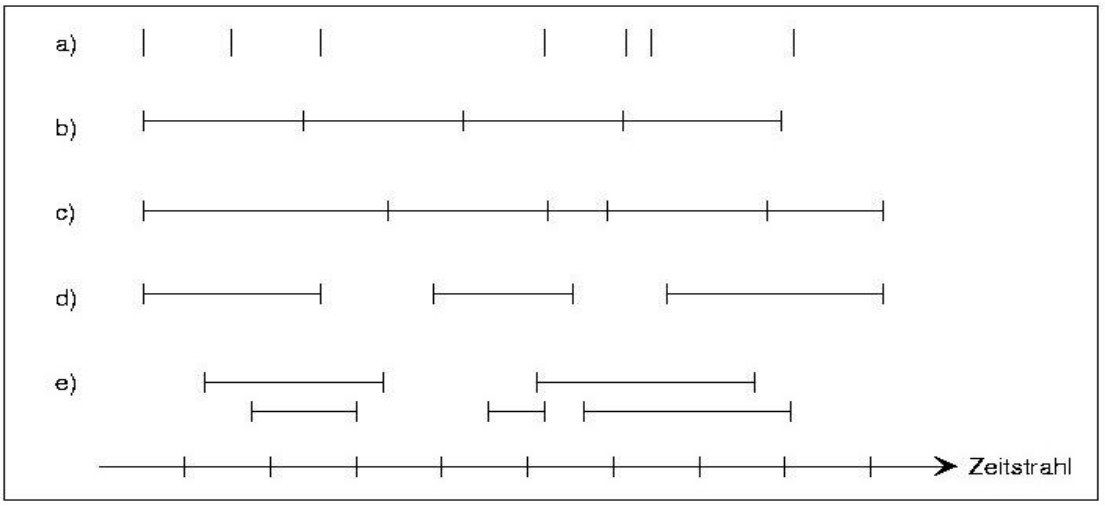


Abb. 78: Zeitkategorien auf den Zeitstrahl

Zeitpunkte sind Momente oder Augenblicke des Eintretens eines Ereignisses, denen keine Dauer zugeordnet werden kann, d. h. sobald ein Zeitpunkt auf dem Zeitstrahl erreicht ist, ist er bereits vergangen (s. Abbildung 78 a). Zeitpunkte werden durch eine Angabe mit dem bereits eingeführten Datentyp Date/Time spezifiziert. Zeitpunktangaben können als beschreibendes Attribut einem Entitytyp oder einem Beziehungstyp zugeordnet werden. Die Angabe des spezifizierten Zeitpunkts kann z. B. in der Form des Tagesdatums und der Uhrzeit erfolgen. Da die Zeit in beliebig kleine Einheiten unterteilt werden kann, muß die Angabe prinzipiell auf einem Punkt bezogen werden. So muß beispielsweise implizit die Uhrzeit 0:00 oder 12:00 angenommen werden, wenn nur das Tagesdatum angegeben wird. Scheer benutzt in seinem Referenzmodell einem Entitytyp *Zeit*, der nur aus einem Schlüssel mit dem Datentyp Date/Time besteht (Scheer 90f, S. 28 und 34). Dieser Entitytyp ermöglicht es, Objekte als Beziehungen zu modellieren, deren Existenz unter anderem von dem Entstehungszeitpunkt abhängig ist. Als eine solche Aggregation werden insbesondere die Bewegungsdaten wie Kunden- und Fertigungsaufträge, Bestellungen und Angebote modelliert.

Eine **Zeitdauer** ist durch einen Anfangs- und einen Endzeitpunkt definiert. Alle Zeitpunkte zwischen Anfangs- und Endzeitpunkt gehören zu der Zeitdauer. Scheer nutzt den Entitytyp *Zeit* auch zur Wiedergabe von Zeitdauern, z. B. dem geplanten Absatz eines Produkts in einer Periode (Scheer 90f, S. 117). Tasker modelliert einen Entitytyp *Time-Period*, der einen Beziehungstyp *begins-at* sowie einen Beziehungstyp *ends-at* zu einen Entitytyp *Time-Point* eingeht (Tasker 88).

Perioden sollen als Zeitdauern definiert werden, die sich regelmäßig und ohne Unterbrechung wiederholen und immer die gleiche Dauer bzw. eine ableitbare Dauer aufweisen (s. Abbildung 78 b). Perioden gleicher Dauer sind bsw. Minuten, Stunden oder Dekaden. Perioden wie Jahre, Monate oder Tage haben durch Schalttage, Schaltsekunden oder die Sommerzeit-Winterzeit-Umstellung nicht immer die gleiche Dauer, diese kann jedoch leicht abgeleitet werden. Für die Spezifizierung einer Periode innerhalb eines Rasters reicht die Angabe eines Zeitpunktes, z. B. wird mit der Angabe 1. Januar 1990 bei einem Tagesraster die Dauer von 0:00 Uhr bis 24:00 Uhr verstanden.

Unter **stetigen, ungleichmäßigen Zeitdauern** werden Zeitdauern verstanden, bei denen die Dauern der einzelnen Zeitstrecken unterschiedlich sind, der Anfangszeitpunkt einer Zeitstrecke aber jeweils mit dem Endzeitpunkt der vorherigen Zeitstrecke übereinstimmt (s. Abbildung 78 c). Dadurch läßt sich die Dauer durch einen Zeitpunkt (z. B. Anfangszeitpunkt) und einen Zeitpunkt einer benachbarten Dauer ableiten (Endzeitpunkt aus dem Anfangszeitpunkt des Nachfolgers). Abbildung 79 zeigt das Beispiel einer

Maschine, der ein über der Zeit veränderbarer Maschinenstatus zugewiesen werden kann. Die Dauer eines Zustands ist unterschiedlich. Einer Maschine muß aber immer genau ein gültiger Status zugewiesen werden. Abbildung 79 a zeigt den *Zustand* als eine Aggregation der Entitytypen *Maschine*, *Status* und *Zeitpunkt*, wobei der eingehende Zeitpunkt den Anfangszeitpunkt des Maschinenzustands repräsentiert. Das Attribut *Endzeitpunkt* der Beziehung wird mit Hilfe der Bedingung <1.2> aus dem Anfangszeitpunkt des nächsten Zustands der gleichen Maschine abgeleitet. Bedingung <1.1> legt über die Determinante fest, daß eine Maschine zwar über mehrere Zustände verfügen kann, dies aber nur zu unterschiedlichen Zeitdauern.

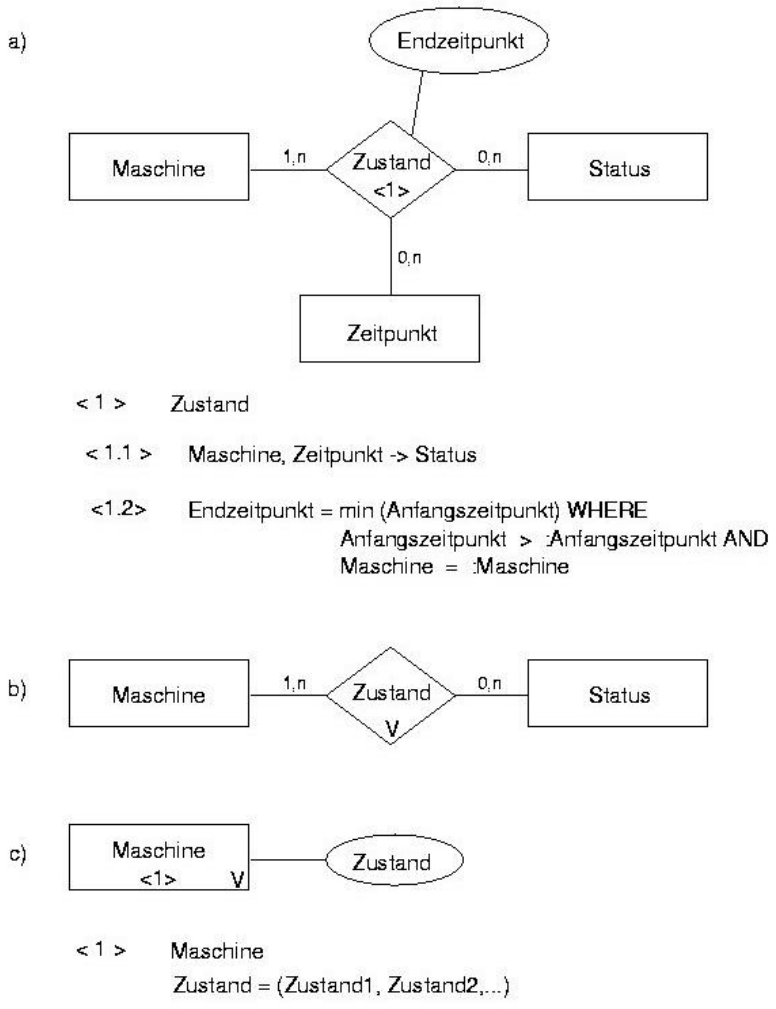


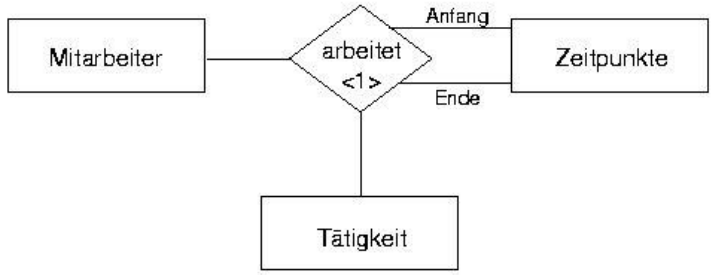
Abb. 79: Beispiel stetiger, ungleichmäßiger Zeitdauern

Abbildung 79 b zeigt die Darstellung des Sachverhalts mit Hilfe des versionsbehafteten Beziehungstyps *Zustand*. Eine Abbildung über eine Versionsdarstellung ist möglich, da die

Bedingung der Stetigkeit auch für Versionen gilt. Der Beziehungstyp *Zustand* hat aus Sicht des Typs *Maschine* einen Komplexitätsgrad von (1,1), da eine Maschine nur genau einen Zustand einnehmen kann, nämlich den gegenwärtigen. Vergangene oder zukünftig gültige Zustände werden über die Versionsfunktionen dargestellt. Zu beachten ist, daß auch bei einem versionsbehafteten Beziehungstyp *Zustand* eine Änderung des Status eine Aggregation verschiedener Entities der Typen *Maschine* und *Status* darstellt. Da die Determinante des Beziehungstyps *Zustand* allerdings allein vom Typ *Maschine* abhängt, können die einzelnen Zustände einer Maschinen gut abgeleitet werden. In Abbildung 79 c wird der Status als Attribut *Zustand* des versionsbehafteten Entitytyps *Maschine* dargestellt. Die Änderung eines Maschinenstatus führt zu einer neuen Ausprägung des Attributs *Zustand* und damit zu einer neuen Version des Entities der Maschine. Auch in dieser Darstellungsvariante wird die Bedingung der Stetigkeit der Version genutzt. Im Gegensatz zu den beiden vorherigen Ansätzen, werden die möglichen Statusbeschreibungen nicht als neutrales Stammdatum in einem eigenen Entitytyp hinterlegt. Daraus folgt, daß die Existenz eines möglichen Status nur dann ersichtlich ist, wenn eine Maschine diesen aktuellen Zustand annimmt bzw. zu einem früheren Zeitpunkt angenommen hatte und dieser über die Historie der Version ableitbar ist. Deshalb wird das Attribut *Zustand* vom Datentyp *Enum* definiert und alle möglichen Statuszustände in der Integritätsbedingung <1> hinterlegt, so daß diese jederzeit definiert sind. Allerdings bedeutet die Definition eines neuen Zustandes eine Änderung des konzeptuellen Schemas, während in den vorherigen Ansätzen neue Zustände über eigene Entities eingeführt werden können und das Schema unberührt bleibt.

Bei **unstetigen, überlappungsfreien Zeitdauern** kann zwischen zwei Zeitdauern eine beliebig lange Zeitstrecke liegen, d. h. daß End- und Startzeitpunkt zweier aufeinanderfolgender Dauern nicht identisch sein müssen (s. Abbildung 78 d). Deshalb läßt sich die Dauer auch nicht aus Zeitpunkten benachbarter Zeitdauern ableiten. In Abbildung 80 ist das Beispiel einer Tätigkeitszuordung zu einem Mitarbeiter dargestellt. Einem Mitarbeiter kann für eine Zeitdauer maximal eine Tätigkeit zugeordnet werden, ihm kann aber auch keine Tätigkeit zugeordnet werden. In Lösung a) wird der Beziehungstyp *arbeitet* aus den Entitytypen *Mitarbeiter*, *Tätigkeit*, *Zeitpunkt* mit der Kantenrolle *Anfang* sowie *Zeitpunkt* mit der Kantenrolle *Ende* aggregiert. Die Komplexität aller Kanten ist (0,n). Bedingung <1.1> regelt die funktionale Abhängigkeit des Vierfachbeziehungstyps *arbeitet*, Bedingung <1.2> die wechselseitigen Abhängigkeiten der beide Attribute Anfangs- und Endzeitpunkt innerhalb einer Beziehung, und <1.3> sowie <1.4> die wechselseitigen Abhängigkeiten innerhalb aller Beziehungen des Typs *arbeitet* eines Entities *Mitarbeiter*. In Lösung b) wird ein binärer Beziehungstyp *Unst-Übfr-ZDauer* (*unst*etige, *üb*erlappungs*fr*eie Zeit*dauer*) mit den Kanten *Anfang* und *Ende* aus den Entitytyp *Zeitpunkt* aggregiert.

a)



< 1 > arbeitet

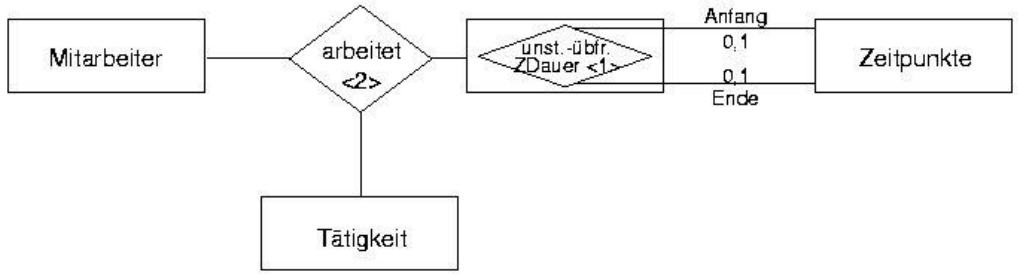
< 1.1 > Mitarbeiter, Anfangszeitpunkt -> Endzeitpunkt, Tätigkeit

<1.2> Anfangszeitpunkt -> Endzeitpunkt

<1.3> Anfangszeitpunkt > max (Endzeitpunkt) **WHERE**
Anfangszeitpunkt < :Anfangszeitpunkt **AND**
Mitarbeiter = :Mitarbeiter

<1.4> Endzeitpunkt < min (Anfangszeitpunkt) **WHERE**
Endzeitpunkt > :Endzeitpunkt **AND**
Mitarbeiter = :Mitarbeiter

b)



< 1 > unst.-übfr.-ZDauer

< 1.1 > Anfangszeitpunkt -> Endzeitpunkt

<1.2> Anfangszeitpunkt > max (Endzeitpunkt) **WHERE**
Anfangszeitpunkt < :Anfangszeitpunkt

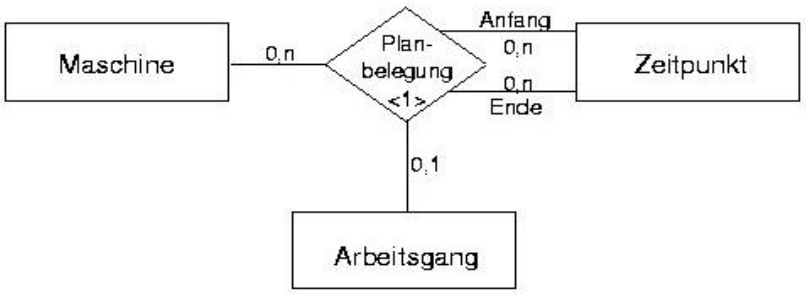
<1.3> Endzeitpunkt < min (Anfangszeitpunkt) **WHERE**
Endzeitpunkt > :Endzeitpunkt

< 2 > arbeitet
Mitarbeiter, unst.-übfr.-ZDauer -> Tätigkeit

Abb. 80: Beispiel unstetiger, überlappungsfreier Zeitdauern

Die Komplexität jeder Kante ist (0,1), da ein Zeitpunkt maximal je einmal als Start- und einmal als Endzeitpunkt von überlappungsfreien Dauern dienen kann. Integritätsbedingung <1> stellt analog zur Lösung a) die Überlappungsfreiheit sicher. Der Beziehungstyp *Unst-Übfr-ZDauer* bildet als uminterpretierter Entitytyp mit den Entitytypen *Mitarbeiter* und

Tätigkeit den Beziehungstyp *arbeitet*, dessen funktionale Abhängigkeit in Bedingung <2> formuliert ist. Im Unterschied zur Lösung a), bei der nur die Arbeitsdauern eines Mitarbeiters überlappungsfrei sein müssen, sind nach der Definition von Lösung b) die Dauern aller Mitarbeiter überlappungsfrei. Die Integritätsbedingungen sind folglich wesentlich strenger. Eine solche Abbildung ist nur dann sinnvoll, wenn die Tätigkeitszuordnungen für alle Mitarbeiter gleichzeitig wechseln bzw. wenn die Zeitdauern unternehmensweite Gültigkeit besitzen. Unstetige, überlappungsfreie Zeitdauern können auch als stetige, ungleichmäßige Zeitdauern dargestellt werden, wenn für die nicht abzubildenden Zeitdauern sogenannte Dummy-Dauern definierte werden. Für das o. g. Beispiel der Mitarbeiterzuordnung könnte eine (Schein-) Tätigkeit "abwesend" definiert werden, die immer dann zugeordnet wird, wenn der Mitarbeiter an keiner echten Tätigkeit arbeitet.



< 1 > Planbelegung

< 1.1 > Arbeitsgang -> Maschine, Anfangszeitpunkt, Endzeitpunkt

<1.2> Anfangszeitpunkt < Endzeitpunkt

Abb. 81: Beispiel unstetiger, überlappender Zeitdauern

Bei **unstetigen, überlappenden Zeitdauern** können sich die Zeitstrecken einzelner Dauern überlagern (s. Abbildung 78 e). Bei dem Beispiel in Abbildung 81 soll eine Maschine bei der Planbelegungsrechnung so mit Arbeitsgängen belegt werden, daß zu einem Zeitpunkt mehrere Arbeitsgänge auf einer Maschine einplanbar sind. Andererseits soll ein Arbeitsgang nur auf maximal einer Maschine für nur einen Zeitraum eingeplant werden können. Der Vierfachbeziehungstyp *Planbelegung* wird aus den Entitytypen *Maschine*, *Arbeitsgang*, *Zeitpunkt* mit der Kante *Anfang* sowie *Zeitpunkt* mit der Kante *Ende* gebildet. Da ein Arbeitsgang nur einmal verplant werden kann, weist die Kante zwischen dem Entitytyp *Arbeitsgang* und dem Beziehungstyp *Planbelegung* einen Komplexitätsgrad von (0,1) auf. Bedingung <1> gibt die daraus resultierende funktionale Abhängigkeit sowie die wechselseitige Abhängigkeit der Attribute *Anfangszeitpunkt* und *Endzeitpunkt* wieder.

7.3 Grunddaten

Ressourcen

Die für die Fertigung notwendigen Produktionsfaktoren werden zusammenfassend als Ressourcen bezeichnet. Zu ihnen gehören Fertigungsmittel wie NC-Maschinen und Handarbeitsplätze, Fertigungshilfsmittel wie Werkzeuge und Vorrichtungen, NC-Programme, Zeichnungen, Meß- und Prüfmittel, Lager- und Transportmittel sowie die menschliche Arbeitskraft (vgl. auch Ley 84, S. 4). Die einzelnen Arten von Ressourcen können zu Gruppen gleicher Funktionalität zusammengefaßt werden, z. B. alle funktionsgleichen Maschinen zu einer Maschinengruppe. Diese Zusammenfassungen sind allerdings nicht Gruppierungen im datenmodelltechnischen Sinn. So kann eine Maschine, die aus technologischer Sicht ein Bearbeitungszentrum oder eine Flexible Fertigungszelle darstellt, mehrere Funktionen erfüllen und damit mehreren Maschinengruppen zugeordnet werden. Abbildung 83 zeigt die Darstellung der Ressourcen im Expanded ERM.

Die Entitytypen *Maschine*, *WerkVorr* (*Werk*zeuge und *Vorr*ichtungen), *Fördermittel* und *MeßPrüfmittel* werden jeweils über Zuordnungsbeziehungen entsprechenden Gruppen zugeordnet. Dabei muß jedes Entity mindestens einer Gruppe zugeordnet sein. Aufgrund der bereits beschriebenen Multifunktionalität werden aber auch mehrere Gruppenzuordnungen zugelassen. Eine Gruppe benötigt nicht unbedingt eine Zuordnung einer individuellen Ressource, da häufig bei geringwertigen oder unwichtigen Ressourcen keine explizite Erfassung der einzelnen Objekte erfolgt. Einzelne Entities des Typs *Mitarbeiter* werden über den Beziehungstyp *Fähigkeit* zu der Ressourcengruppe *PersQualif* (*Pers*onal*qualif*ikation) zusammengefaßt. Die Beziehung ist aber optional (Komplexitätsgrad (0,n)), da nicht jeder Mitarbeiter eine Qualifikation für die Fertigung besitzt.

Werkzeuge und Vorrichtungen (im folgenden einfach Werkzeuge genannt) werden häufig, z. B. im Bereich der automatisierten, spanenden Fertigung, nicht als Komplettwerkzeuge, sondern als Einzelmodule vorgehalten (Eversheim/Wienand 87). Sie werden vor der Verwendung vom Werkzeugbau montiert und nach der Fertigung wieder in Module zerlegt. Abbildung 82 zeigt die Darstellung von Werkzeugstücklisten als Gozintograph. Die Struktur des Gozintographen wird durch Werkzeugfunktionen gebildet (dargestellt durch Quadrate), denen Einzelwerkzeuge (dargestellt durch Kreise) zugeordnet sind. Ein Komplettwerkzeug kann durchaus gleichzeitig als Einzelwerkzeug existieren (1) und als Baukasten aus Moduln montiert werden (3 und 4). Ein Einzelwerkzeug kann mehrere Funktionen erfüllen (11). Der Gozintograph wird als rekursiver, binärer Beziehungstyp *WVStruk* (*W*erkzeug- und *V*orrichtungs*struk*tur) des Entitytyps *WerkVorrFunk* (*Werk*zeug- und *Vorr*ichtungs*funk*tion)

mit den Kanten *OWV* und *UWV* im Expanded ERM abgebildet. Integritätsbedingung <1> schließt rekursive Stücklisten aus (die Darstellung ist auf Objekttypebene rekursiv, nicht jedoch auf Objektebene).

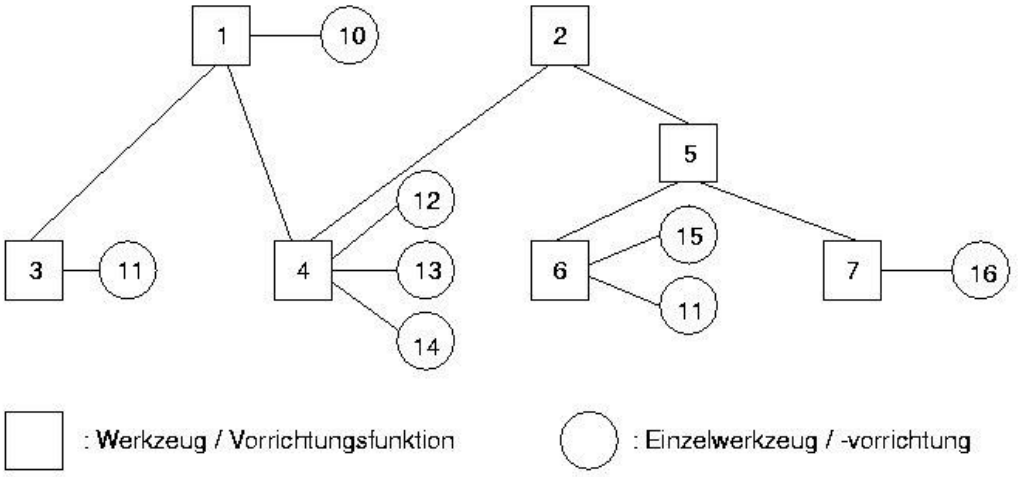


Abb. 82: Werkzeuge und Vorrichtungen als Gozintograph

Da die Ressourcen *NC-Programm* und *Zeichnung* beliebig vervielfältigt werden können, ist die Verwaltung von individuellen NC-Programmen oder Zeichnungen nicht sinnvoll. Eine Unterscheidung zwischen NC-Programm- oder Zeichnungsgruppen und NC-Programmen oder Zeichnungen entfällt daher.

Die Ressource *Teil* wird durch die für die Fertigung benötigten Materialen gebildet. Ein Entity dieses Typs bezeichnet eine Art von Materialien, z. B. Stahlblech mit 3mm Dicke. Die Verwaltung von individuellen Teilen, z. B. einer bestimmten Stahlplatte, wird im Zusammenhang mit den Chargendaten behandelt.

Alle genannten Entitytypen, die einen Gruppenbegriff darstellen, werden zu dem Entitytyp Ressourcengruppe generalisiert, wobei eine *Ressourcengruppe* genau einem Subtyp entsprechen muß. Daraus ergibt sich für die Generalisierung eine Kennzeichnung von (1,1). Alle einzelnen Ressourcen werden zu dem Entitytyp Einzelressource mit der Kennzeichnung (1,1) generalisiert.

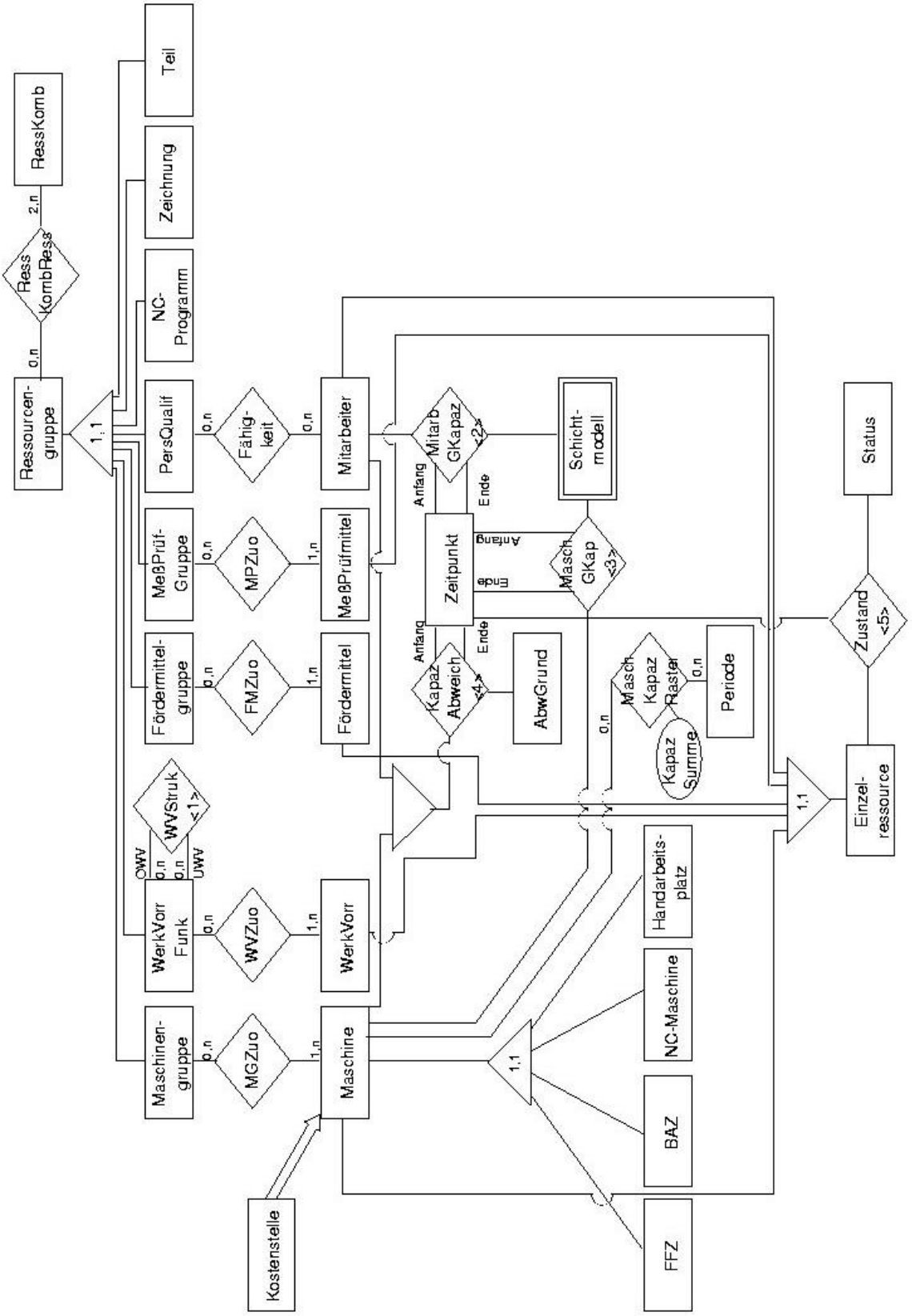


Abb. 83-1: PERM der Fertigungsgrunddaten

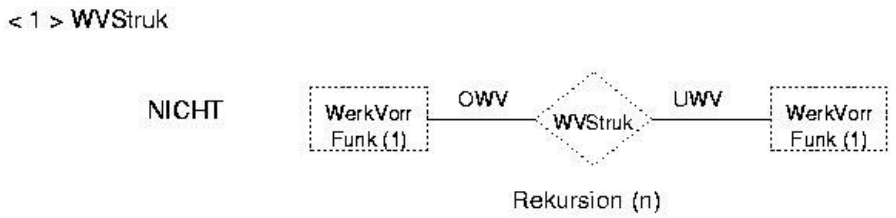


Abb. 83-2: PERM der Fertigungsgrunddaten

Der Typ *Maschine* kann aus technologischer Sicht in die Typen *FFZ* (Flexible Fertigungszelle), *BAZ* (Bearbeitungszentrum), *NC-Maschine* und *Handarbeitsplatz* mit der Kennzeichnung (1,1) spezialisiert werden. Maschinen bzw. Maschinengruppen werden

üblicherweise zu Kostenstellen zusammengefaßt. Da keine hierarchische Beziehung zwischen Maschinen und Maschinengruppen besteht, wird der Typ *Maschine* und nicht der Typ *Maschinengruppe* zu dem Entitytyp *Kostenstelle* gruppiert.

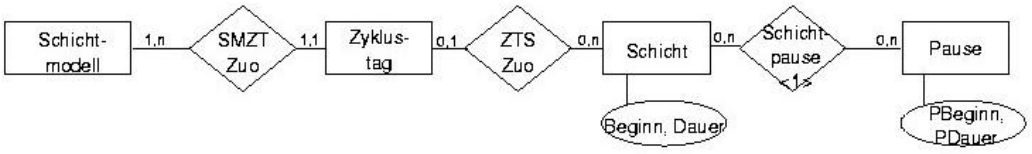
Weiterhin soll die Möglichkeit geschaffen werden, für die Fertigung sinnvolle Ressourcenkombinationen zu hinterlegen, wie sie z. B. für die Arbeitsplanerstellung benötigt werden. So kann auf einer Maschine nur mit bestimmten Werkzeugen unter Einsatz bestimmter Vorrichtungen gefertigt werden. Es können weiterhin eine gewisse Personalqualifikation und besondere Meß- und Prüfmittel notwendig sein. Würde man diese Ressourcenkombinationen als Beziehungen zwischen den einzelnen Ressourcenarten definieren, z. B. eine Beziehung zwischen *Maschinengruppe* und *Werkzeuggruppe*, so wären aufgrund der acht eingeführten Ressourcenarten allein für binären Kombinationen 28 verschiedene Beziehungstypen zu definieren. Nicht abgedeckt wären dabei Kombinationen mehrerer Ressourcenarten sowie die Möglichkeit, daß bei einer Kombination mehrere Entities einer Ressourcenart notwendig sind, z. B. für eine Maschine zwei Werkzeuge. Deshalb wird ein eigener Entitytyp *RessKomb* (*Ress*ourcen*komb*ination) eingeführt, der über den Beziehungstyp *RessKombRess* mit den Ressourcen verbunden ist. Der Komplexitätsgrad beträgt (2,n), da für eine Kombination mindestens zwei Ressourcen notwendig sind.

Kapazitätsdaten

Die Arbeitszeit wird durch eine Schichtregelung vorgegeben, wofür Schichtmodelle konstruiert werden. Ein Schichtmodell definiert die Regeln für die Ableitung der Schichtzeit, unabhängig von einem konkreten Tagesdatum. Das Schichtmodell soll bezüglich der Möglichkeiten

- der Arbeitsdauer pro Schicht,
- der Pausenanzahl pro Schicht,
- der Pausendauer pro Pause und
- des Zyklus, in dem sich die Schichten wiederholen,

variabel sein und eine genaue (minutengenau) Angabe der Arbeitszeiten enthalten.



< 1 > Schichtpause

< 1.1 > Dauer > PBeginn + PDauer

< 1.2 > PBeginn > max (PBeginn + PDauer) WHERE
PBeginn < :PBeginn AND
Schicht = :Schicht

< 1.3 > PBeginn + PDauer < min (PBeginn) WHERE
PBeginn > :PBeginn AND
Schicht = :Schicht

Abb. 84: PERM zum Schichtmodell

Abbildung 84 zeigt das PERM des Schichtmodells. Dem Entitytyp *Schichtmodell* werden mindestens ein oder mehrere *Zyklustag*e zugeordnet. Die Schichtregelung eines Modells kann sich beispielsweise wöchentlich, zweiwöchentlich oder zehntägig wiederholen. Für jeden Tag des Zyklusses wird dem Typ *Schichtmodell* ein Entitytyp *Zyklustag* zugeordnet. Die Reihenfolge kann über eine Sortierung des Schlüssels erfolgen. Die Länge eines Schichtzyklusses ergibt sich aus der Anzahl der Beziehungen des Typs *SMZTZuo* (*Schicht*m*odell-Zyklus*t*ag-*Zuo*rdnung). Einem *Zyklustag* kann maximal eine Schicht über den Beziehungstyp *ZTSZuo* (*Zyklus*tag-*Schicht*-*Zuo*rdnung) zugeordnet werden. Wird keine *Schicht* zugeordnet, so ist dies ein freier Tag, z. B. Samstag und Sonntag. Der Entitytyp *Schicht* besitzt die Attribute *Beginn* und *Dauer*. *Beginn* gibt die Uhrzeit des Schichtbeginns, *Dauer* die Länge der Schicht wieder. Einer Schicht können mehrere Pausen zugeordnet werden. Der Entitytyp *Pause* spezifiziert über das Attribut *PBeginn* mit der Dimension Stunde den relativen Beginn einer Pause nach dem Schichtbeginn sowie mit dem Attribut *PDauer* die Länge der Pause. Integritätsbedingung <1.1> der Zuordnung *Schichtpause* stellt sicher, daß keine Pausen nach dem Schichtende zugeordnet werden können, Bedingung <1.2> und <1.3>, daß sich die Pausen einer Schicht nicht überlappen.

In Abbildung 85 ist das Schichtmodell einer Wechselschicht mit einem zweiwöchigen Zyklus dargestellt. Die ersten fünf Tage wird in einer achtstündigen Schicht ab 6:00 Uhr gearbeitet, mit einer viertelstündigen Pause ab 8:00 Uhr und einer dreiviertelstündigen Pause ab 10:30 Uhr. Der 6. und 7. Tag sind arbeitsfrei. Am achten bis zwölften Tag beginnt die Schicht um 14:00 Uhr. Am letzten Arbeitstag eines Zyklusses wird nur von 14:00 Uhr bis 20:00 Uhr mit einer viertelstündigen Pause ab 16:00 Uhr und einer halbstündigen Pause ab 18:00 Uhr gearbeitet.

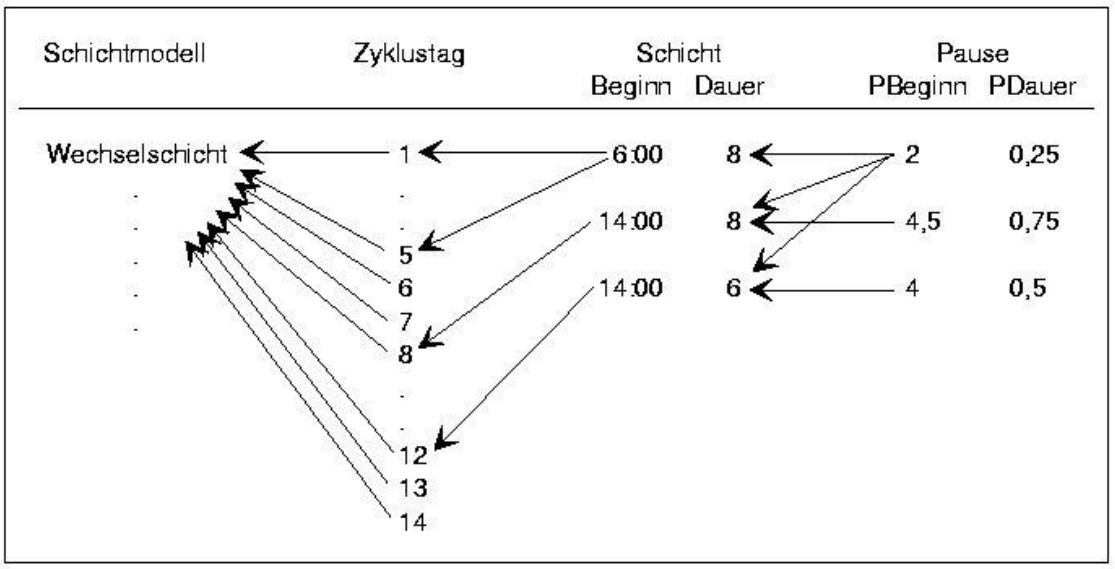


Abb. 85: Darstellung eines Schichtmodells

Aus den Entitytypen und Beziehungstypen der Schichtmodelldefinition aus Abbildung 84 wird ein Cluster *Schichtmodell* gebildet und in das PERM in Abbildung 83 übernommen. Mit dem Schichtmodell kann die Grundkapazität von Ressourcen definiert werden. Über den Beziehungstyp *MitarbGKapaz* (*Mitarb*eiter*g*rund*kapaz*ität) wird das Cluster *Schichtmodell* zusammen mit dem Entitytyp *Zeitpunkt* dem Typ *Mitarbeiter* zugeordnet. Die Kante des Clusters *Schichtmodell* referenziert den gleichnamigen Entitytyp innerhalb des Clusters und benötigt daher keinen Namen. Die Zuordnung ist bezüglich eines Mitarbeiters eine unstetige, überlappungsfreie Zeitdauer, da einem Mitarbeiter zu einem Zeitpunkt nur maximal ein Schichtmodell oder kein Schichtmodell zugeordnet sein kann. Integritätsbedingung <2.1> definiert die funktionalen Abhängigkeiten, <2.2> bis <2.4> die Überlappungsfreiheit bezüglich eines Mitarbeiters. Die Zuordnung der Schichtmodelle zu den Mitarbeitern wird beispielsweise für die Zutrittskontrolle, Personalabrechnung oder Belegungsplanung benötigt.

Im Allgemeinen wird die Belegungsplanung allerdings nicht auf die Mitarbeiterkapazität, sondern auf die Maschinekapazität bezogen (Kinzer 71; Aldinger 85; Kang 87), wobei von Mitarbeiterkapazität abstrahiert wird. Den Maschinen wird eine Kapazitätsbegrenzung zugewiesen, die den Anwesenheitszeiten des Bedienungspersonals entspricht. Deshalb wird analog zu dem Beziehungstyp *MitarbGKapaz* ein Beziehungstyp *MaschGKapaz* (*Masch*inen*g*rund*kapaz*ität) für die Zuordnung des Schichtmodells zu den Maschinen eingeführt. Im Unterschied zu den Mitarbeitern kann aber eine Maschine in mehreren Schichten zur Verfügung stehen, so daß die funktionalen Abhängigkeiten des

Beziehungstyps *MaschGKapaz* in der Integritätsbedingung <3.1> entsprechend geändert werden müssen. Bezüglich einer Maschine stellt die Zuordnung eine unstetige, überlappende Zeitdauer dar. Häufig erfolgt die Kapazitätsterminierung nur periodenrastergenau, z. B. auf Wochen- oder Tagesbasis, so daß die Kapazität nicht als ein Kontinuum, sondern als Raster, auch "Kapazitätstöpfe" genannt, abgebildet wird (vgl. Abbildung 86). Innerhalb eines Rasters wird die Kapazität als Summe (z. B. in Stunden) angegeben, über deren Verteilung innerhalb des Rasters keine Angaben gemacht werden. Ein solches Vorgehen ist nur mit einem einheitlichen Periodenraster für alle Kapazitätseinheiten sinnvoll. Dargestellt wird dies durch den Beziehungstyp *MaschKapazRaster* (*Masch*inen*kapaz*itäts*raster*) zwischen dem Typ *Maschine* und dem Entitytyp *Periode*. Die Beziehung wird durch das Attribut *KapazSumme* beschrieben.

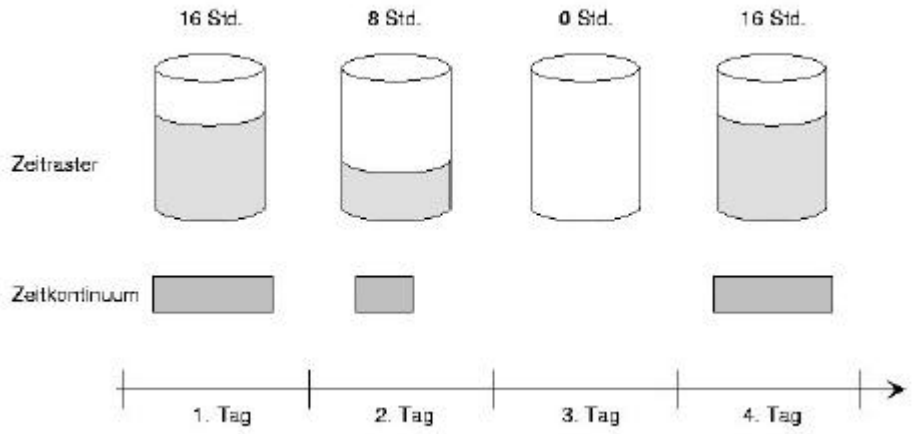


Abb. 86: Kapazitätsangabe als Raster und als Kontinuum

Die Anpassung an kurzfristige Kapazitätsschwankungen, z. B. eine Sonderschicht oder Überstunden, kann bei Kapazitätsangaben über Schichtmodelle durch Zuordnung eines neuen Schichtmodells für die entsprechende Zeitdauer erfolgen. Andererseits ist es sinnvoll, diese kurzfristigen Schwankungen von der Grundkapazität zu trennen, da die Grundkapazität eher den Charakter von Stammdaten aufweist. Dafür wird der Beziehungstyp *KapazAbweich* (*Kapaz*itäts*abweich*ung) eingeführt. Er ist eine Beziehung der alternativen Objekttypen *Maschine* und *Mitarbeiter*, der *Anfangs-* und *Endzeitpunkte* sowie des Entitytyps *AbwGrund* (*Abw*eichungs*grund*). Bezüglich der alternativen Objekttypen hat die Beziehung den Charakter unstetiger, überlappungsfreier Zeitdauern. Über den Abweichungsgrund können sowohl positive (Kapazitätsbereitstellung) als auch negative Abweichungen (keine Kapazität) formuliert werden. Integritätsbedingung <4.1> legt über die

funktionalen Abhängigkeiten fest, daß einem Mitarbeiter bzw. einer Maschine zu einem Zeitpunkt nur eine Kapazitätsabweichung zugeordnet werden kann.

Die Information der generellen Arbeitszeit in Form eines Fabrikkalenders wird in dem Entitytyp *FreieTage* hinterlegt. Dort wird für jeden Nichtarbeitstag ein Entity angelegt. Der Entitytyp hat die Eigenschaft einer Periode. Er wird deshalb durch einen Schlüssel des Datentyps DATE mit dem gregorianischen Datumsformat Tag/Monat/Jahr identifiziert und kann über ein Attribut *Grund* weiter spezifiziert werden. Wird die generelle Arbeitszeit in der klassischen aber unflexiblen Form eines Fabrikkalenders geführt, so könnte anstelle des Typs *FreieTage* ein Typ *Fabrikkalender* mit den Attributen *Datum* und *fortlaufende Tagesnummer* eingeführt werden, bei dem allerdings über komplizierte Integritätsbedingungen eine ununterbrochene Folge der Tagesnummern sowie eine korrespondierende, aufsteigende Folge von gregorianischen Tagesdaten sicherzustellen wäre.

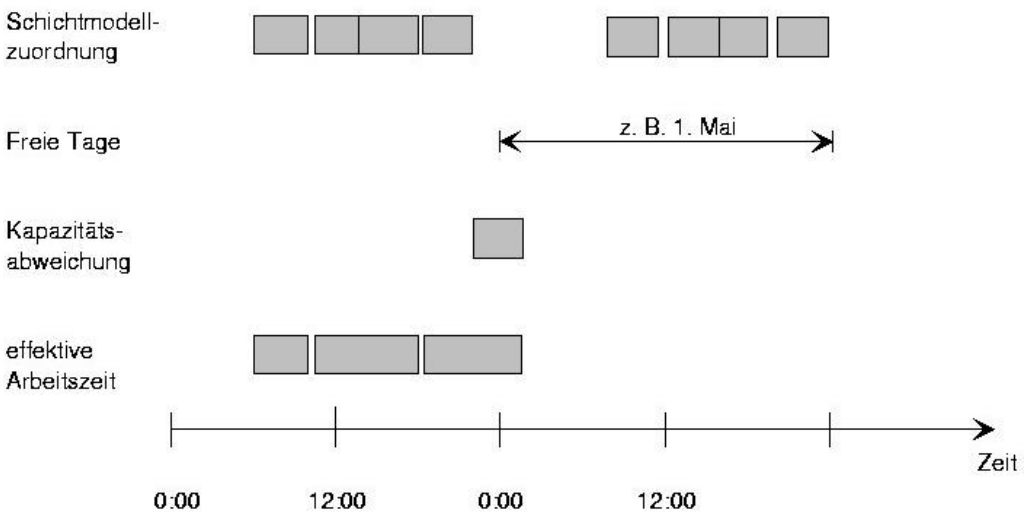


Abb. 87: Aus Schichtmodell, freien Tagen und Kapazitätsabweichungen abgeleitete effektive Arbeitszeit

Aus den genannten Möglichkeiten zur Kapazitätsangabe leitet sich die effektive Arbeitszeit wie folgt ab (vgl. auch Abbildung 87):

(1) Die höchste Priorität besitzt die Kapazitätsabweichung. Liegt ein gesuchter Zeitpunkt in einer Dauer einer Abweichung für den Mitarbeiter bzw. die Maschine, so gilt die Aussage des Abweichungsgrundes.

(2) Ist keine entsprechende Abweichung definiert, so wird aufgrund des Entitytyps *FreieTage* ermittelt, ob dieser Tag als arbeitsfrei definiert ist.

(3) Aufgrund der Beziehung *MitarbGKapaz* bzw. *MaschGKapaz* kann über das Schichtmodell die exakte Arbeitszeitregelung für die Kapazitätseinheit ermittelt werden. Eine Schicht, die über die Tagesgrenze um Mitternacht hinausläuft (z. B. von 22.00 Uhr bis 6.00 Uhr), wird üblicherweise komplett zum ersten Tag gezählt. Läuft beispielsweise eine Nachtschicht in einen Feiertag hinein, wird für die komplette Schicht Kapazität zur Verfügung gestellt. Dies bedingt ein gemeinsames Überprüfen der Stufen 2 und 3.

Der Zustand einer Einzelressource, d. h. die Frage nach der Verfügbarkeit einer Ressource, wird mit Hilfe des Entitytyps *Status* und der Zuordnung *Zustand*, bei der auch der Typ *Zeitpunkte* einbezogen wird, abgebildet. Zustände stellen bezüglich einer Einzelressource stetige, ungleichmäßige Zeitdauern dar (vgl. auch Abbildung 79). Entsprechende Integritätsbedingungen sind in <5> formuliert.

7.4 Arbeitsplandaten

Informationen über die Fertigungsvorschriften für die Produktion der Teile sind in den Arbeitsplänen hinterlegt. Sie werden von der Fertigungsplanung oder Arbeitsvorbereitung erstellt und haben längerfristige Gültigkeit. Damit besitzen sie den Charakter von Stammdaten. Die Herstellung eines Teils erfolgt meist in mehreren Arbeitsschritten auf unterschiedlichen Maschinen unter Einsatz unterschiedlichster Ressourcen. Die Arbeitsschritte werden als Arbeitsvorgänge (Döttling 81), Fertigungsaktionen (Zörntlein 88) oder Arbeitsgänge (Scheer 90f) bezeichnet. Die Arbeitsgänge innerhalb eines Arbeitsplans werden bei konventioneller Fertigung in der Regel sequentiell, jeder Arbeitsgang auf einer einzelnen Maschine, abgearbeitet. Durch die Flexibilisierung der Fertigung mit neuen Technologien und Organisationsformen werden aufgrund geänderter Abarbeitungs-möglichkeiten neue Anforderungen an die Datenorganisation gestellt. Dazu gehören u. a. folgende Funktionen:

(1) Beim **Splitten** von Aufträgen oder Arbeitsgängen wird das Fertigungslos in mehrere Teillose aufgeteilt, die zu unterschiedlichen Zeitpunkten oder auf mehreren Maschinen parallel gefertigt werden, um beispielsweise die Auftragsdurchlaufzeit zu verkürzen.

(2) Beim **Raffen** werden technologisch gleiche Bearbeitungsschritte unterschiedlicher Aufträge zu einem Fertigungslos zusammengefaßt, um beispielsweise Rüstzeiten zu sparen.

(3) Ein **Auftragsmix** an einer Arbeitsstation besteht aus mehreren Fertigungslosen unterschiedlicher Aufträge, die in einer verzahnten Reihenfolge ("quasi-parallel") bearbeitet werden. Dies ist z. B. bei Bearbeitungszentren notwendig, wenn nicht genug Werkstückträger für das Material eines Auftrages zur Aufrechterhaltung eines stetigen Bearbeitungsflusses vorhanden sind. Durch das schnelle, automatische Umrüsten eines Bearbeitungszentrums kann dieses abwechselnd mit unterschiedlichen Werkstücken beschickt werden.

(4) Für einzelne Arbeitsgänge können **Ausweichmaschinen** definiert werden, die bei Ausfall oder Kapazitätsüberlastung der ursprünglichen Maschine die Bearbeitung übernehmen können. Werden von der Ausweichmaschine andere Werkzeuge, Vorrichtungen oder NC-Programme benötigt oder müssen die Vorgabezeiten korrigiert werden, so handelt es sich um einen **alternativen Arbeitsgang**. Häufig kann eine Folge von Arbeitsgängen durch einen alternativen Arbeitsgang oder durch Arbeitsgangsequenzen ersetzt werden. Dies kann zu kompletten **alternativen Arbeitsplänen** ausgedehnt werden.

(5) Die Abarbeitung der Arbeitsgänge muß nicht immer in einer festen Reihenfolge geschehen. Wenn einzelne oder mehrere Arbeitsgänge vertauscht werden können, so wird dies als **alternative Reihenfolge** bezeichnet.

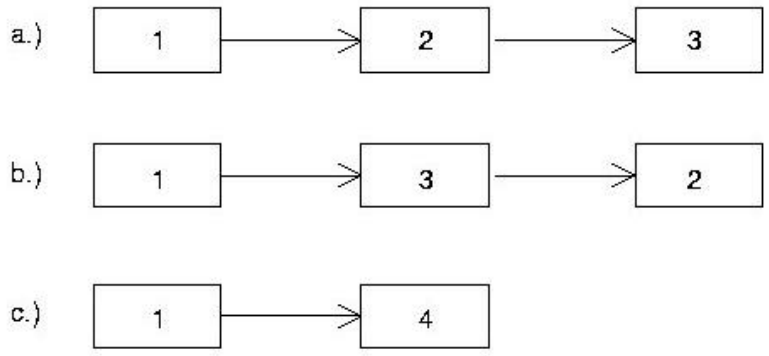


Abb. 88: Beispiel eines Arbeitsplans mit alternativen Bearbeitungspfaden

Die genannten Möglichkeiten können auch kombiniert werden. So kann ein kompletter Auftrag gesplittet werden, wobei die Bearbeitungsfolge eines Teilloses vertauscht wird, während für ein anderes Teillos ein alternativer Arbeitsgang benutzt wird, der zusätzlich mit einem anderen Fertigungsauftrag als Auftragsmix auf einem Bearbeitungszentrum gefertigt wird.

Für die Strukturierung von Arbeitsplänen wurden verschiedene Verfahren vorgeschlagen. Im folgenden werden sie anhand eines Beispiels erläutert und ihre Überführung in das Expanded ERM bewertet werden sollen. Das Beispiel des Arbeitsplans ist in Abbildung 88 dargestellt. Der Arbeitsplan enthält insgesamt vier Arbeitsgänge. Für die Bearbeitung stehen drei alternative Bearbeitungspfade zur Auswahl.

Isolierte Arbeitspläne

Üblicherweise wird ein Arbeitsplan als einfache Folge von Arbeitsgängen abgebildet (Loos 89, Scheer 90f). Bei einer solchen Darstellung muß jeder Bearbeitungspfad als **isolierter Arbeitsplan** wiedergegeben werden. Da einzelne Arbeitsgänge in mehreren Arbeitsplänen vorkommen, wird die Beschreibung der Arbeitsgänge von der Zuordnung zu den Arbeitsplänen getrennt. Abbildung 89 zeigt die Darstellung im Expanded ERM. Einem *Teil* können mehrere Arbeitspläne (*APL*) zugeordnet sein. Einem Arbeitsplan können mehrere Arbeitsgänge (*AG*) über den Beziehungstyp *AGZuo* (*A*rbeits*g*ang*zuo*rdnung) zugeordnet werden, und andererseits kann ein *AG* mehreren *APL* zugeordnet werden. Über den Beziehungstyp *AGZBez* (*A*rbeits*g*angzuordnungs*bez*iehung) wird die Arbeitsgangfolge innerhalb des Arbeitsplans definiert. Wird die Kante *Vor* (*Vor*gänger) der Beziehung nicht mit der Komplexität (0,1), sondern mit (0,n) definiert, so lassen sich innerhalb eines Arbeitsplans die Bearbeitung unterschiedlicher Teile und deren Montage abbilden. Integritätsbedingung <1.1> stellt sicher, daß Arbeitsgänge nicht zu einem Zyklus verbunden werden, Bedingung <1.2>, daß verbundene Arbeitsgänge zum gleichen Arbeitsplan gehören. Für das gezeigte Beispiel in Abbildung 88 müssen drei Arbeitspläne angelegt werden.

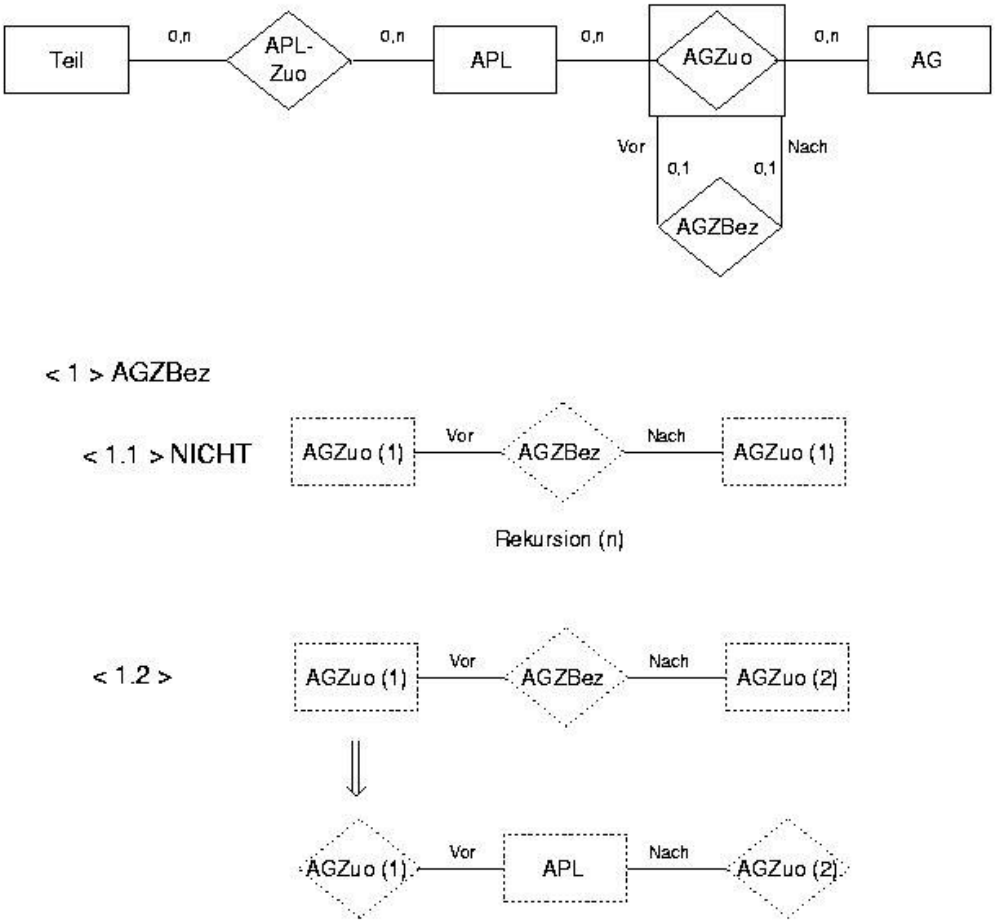


Abb. 89: PERM der Arbeitsplandarstellung mit isolierten Arbeitsplänen

Arbeitspläne über Definition notwendiger Vorgänger

Für die Darstellung alternativer Reihenfolgen innerhalb eines Arbeitsplans können zu jedem Arbeitsgang alle **notwendigen Vorgänger** definiert werden (Ruffing 90, S. 223 f). Dabei werden für jeden Arbeitsgang (*AG*), der durch die Komplexität von (1,1) eindeutig einem *APL* zugeordnet ist, mit Hilfe des Beziehungstyps *notwVorg* alle notwendigen Vorgängerarbeitsgänge definiert (vgl. Abbildung 90). Da ein Arbeitsgang mehrere Vorgänger besitzen und selbst für mehrere Nachfolger Vorgängerarbeitsgang sein kann, haben sowohl die Kante *Vor* als auch die Kante *Nach* (*Nach*folger) die Komplexität von (0,n). Integritätsbedingungen sind analog zu der Abbildung 89 für die Rekursionsfreiheit und die Zugehörigkeit zum gleichen Arbeitsplan notwendig.

Die Abbildung der Bearbeitungspfade a und b des Beispiels aus Abbildung 88 können in einem Arbeitsplan erfolgen. Für Bearbeitungspfad c mit einem alternativen Arbeitsgang ist allerdings ein zweiter Arbeitsplan erforderlich.

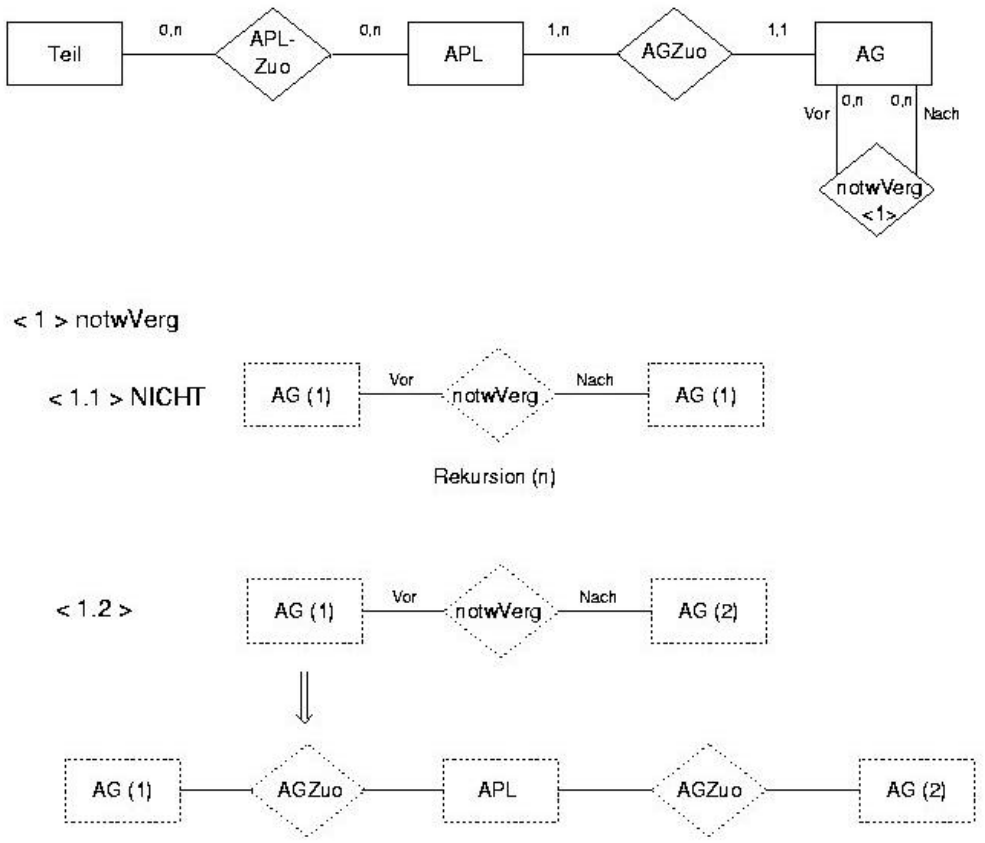


Abb. 90: PERM der Arbeitplandarstellung mit der Methode der Definition notwendiger Vorgänger

Arbeitspläne mit Strukturknoten

Mit Hilfe von **Strukturknoten** lassen sich alle Bearbeitungspfade in einer Darstellung integrieren (Zörntlein 88, S. 132 ff). Abbildung 91 zeigt das Beispiel des Arbeitsplans mit drei Bearbeitungspfaden mit Strukturknoten. Jeder Arbeitsgang wird dabei nur einmal abgebildet. Die Arbeitsgänge sind untereinander über Pfade verbunden, die mit der Kennzeichnung XOR oder AND bezeichnet sein können. XOR (= exklusiv oder) öffnet bzw. schließt alternative Bearbeitungspfade. AND verzweigt in Bearbeitungspfade, die jeweils alle durchlaufen werden müssen, deren Abarbeitungsreihenfolge aber beliebig ist. Die Information der Strukturknoten läßt sich mit Hilfe von drei n-reihigen, quadratischen Matrizen darstellen, wobei "n" die Anzahl der unterschiedlichen Arbeitsgänge ausdrückt.

Alle Matrix-Elemente sind Bool'sche Variablen. Die Ersetzungsmatrix E gibt an, welcher Arbeitsgang durch welchen anderen Arbeitsgang (bei mehreren Arbeitsgängen durch den letzten Arbeitsgang) ersetzt wird. In Abbildung 92 zeigt der Wert 1 des Elements (2,4) (= zweites Element senkrecht, viertes Element waagerecht), daß der Arbeitsgang 2 durch den Arbeitsgang 4 ersetzt werden kann. Die Präzedenzmatrix P stellt dar, welcher Arbeitsgang die Ausführung von anderen Arbeitsgängen voraussetzt. In dem Beispiel benötigen die Arbeitsgänge 2, 3 und 4 jeweils eine vorherige Ausführung des Arbeitsgangs 1. Die Ausschlußmatrix A drückt aus, welcher Arbeitsgang die Ausführung anderer Arbeitsgänge ausschließt. Im vorliegenden Fall schließen sowohl Arbeitsgang 2 als auch 3 den Arbeitsgang 4 aus. Arbeitsgang 4 schließt seinerseits die Bearbeitung der Arbeitsgänge 2 und 3 aus.

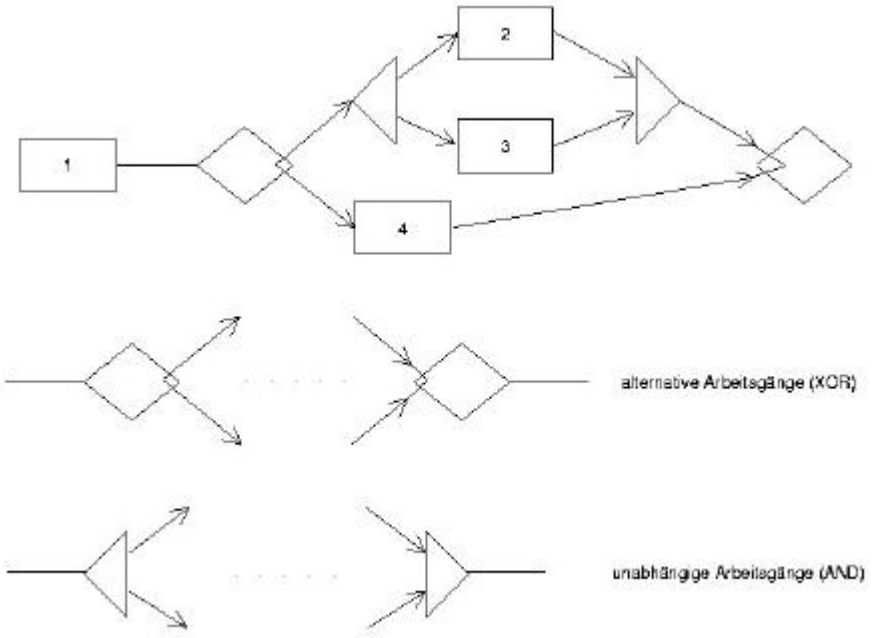


Abb. 91: Darstellung eines Arbeitsplans mit Strukturknoten

In Abbildung 93 wird die Matrizendarstellung im PERM ausgedrückt, wobei alle drei Matrizen in dem Beziehungstyp *EPAMatrix* (Ersetzungs-, Präzedenz- und Ausschlußmatrix) durch die Attribute *EWert*, *PWert* und *AWert* abgebildet werden. Eine Lösung mit drei Beziehungstypen, wobei jede Matrix in einem eigenen Beziehungstyp abgebildet wird, ist ebenso möglich. Da aber jede Beziehung die gleichen Arbeitsgänge verbindet und damit die gleichen Determinanten aufweisen würde, wird die Darstellung in einer Matrix bevorzugt. Alle Bearbeitungspfade können in einem Arbeitsplan untergebracht werden, so daß die Kante von *Teil* zu *APLZuo* eine Komplexität von (0,1) erhält. Integritätsbedingung <1.2> läßt

nur Matrix-Beziehungen von Arbeitsgängen aus gleichen Arbeitsplänen zu. Bedingung <1.1> schließt die Hauptdiagonale der Matrizen als Beziehungen aus, da deren Werte (EWert = 1, PWert = 0, AWert = 0) konstant sind.

E	1	2	3	4
1	1	0	0	0
2	0	1	0	1
3	0	0	1	1
4	0	1	1	1

P	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0

A	1	2	3	4
1	0	0	0	0
2	0	0	0	1
3	0	0	0	1
4	0	1	1	0

Abb. 92: Darstellung der Bearbeitungspfade über Matrizen

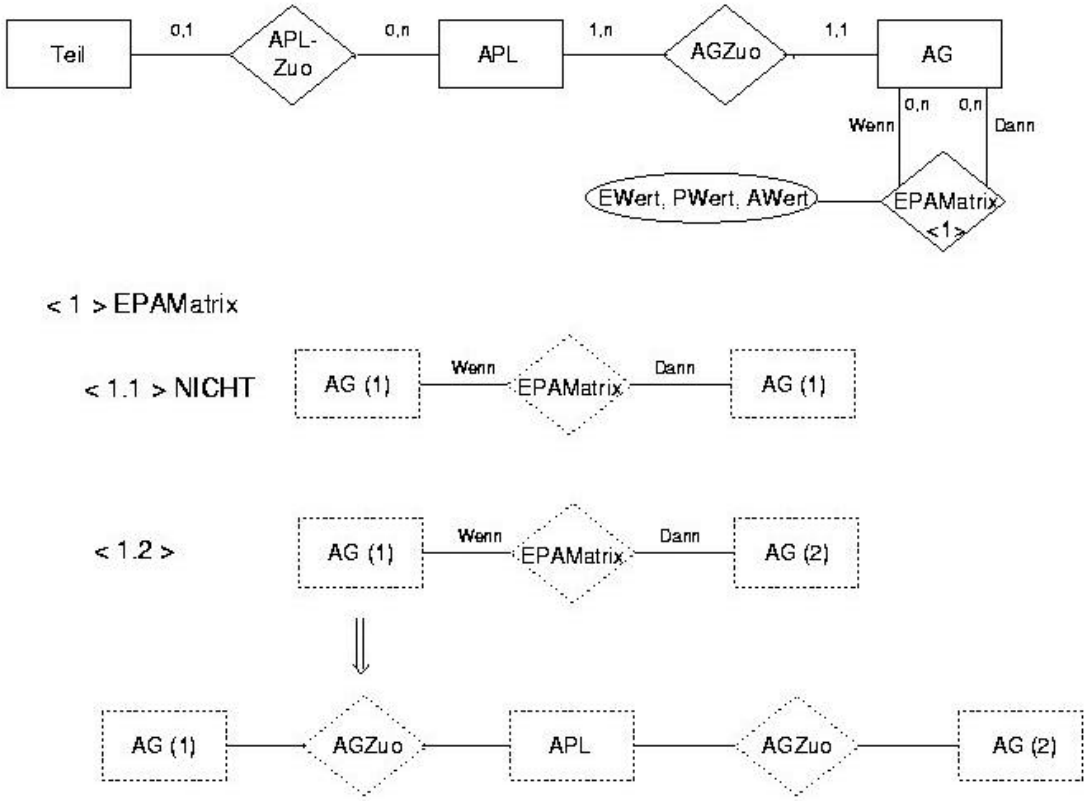


Abb. 93: PERM der Arbeitsplandarstellung mit Ersetzungs-, Präzedenz- und Ausschlußmatrix

Zustandsorientierte Arbeitspläne

Neben der bisher vorgestellten Bearbeitungsorientierung kann die Darstellung auch **zustandsorientiert** erfolgen (Döttling 81, S. 46 ff). Die Zustände des Werkstücks bilden die Knoten innerhalb eines gerichteten Graphen, wobei die Kanten die Arbeitsgänge repräsentieren (Zörntlein 88, S. 128). Abbildung 94 zeigt das Beispiel aus Abbildung 88 als Zustandsgraph. Ursprungszustand, d. h. das Rohmaterial des herzustellenden Teils, ist Zustand Z0, dargestellt durch einen Kreis. Ziel ist das fertige Teil im Zustand Z9. In jedem Zustand kann **eine** der abgehenden alternativen Bearbeitungskanten gewählt werden. So kann nach Zustand Z1, der nach der Bearbeitung durch Arbeitsgang 1 erreicht wird, entweder mit Arbeitsgang 2, 3 oder 4 fortgefahren werden.

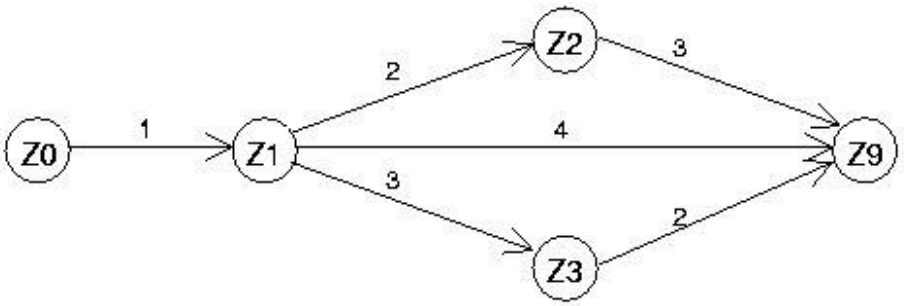


Abb. 94: Darstellung eines Arbeitsplans mit einem Zustandsgraph

Die Übertragung der Datenstrukturen in das Expanded ERM ist in Abbildung 95 enthalten. Ein Teil kann mehrere Zustände annehmen und hat mindestens einen Zustand, den Endzustand (im Beispiel Z9). Ein Zustand gehört zu genau einem Teil. Dies ergibt eine Komplexität von dem Typ *Teil* zu dem Beziehungstyp *TZZuo* (*T*eilezustands*zuo*rdnung) von (1,n) und von *TZustand* (*T*eile*zustand*) zu *TZZuo* (1,1). Von einem Teilezustand kann in einen anderen Zustand über den Beziehungstyp *TZKante* (*T*eilezustands*kante*) mit den Kanten *von* und *nach* gewechselt werden, wobei dies unter Zuhilfenahme von Arbeitsgängen erfolgt. Da ein Zustand über mehrere Arbeitsgänge erreicht werden kann bzw. von einem Zustand mehrere alternative Bearbeitungspfade ausgehen können, ist die Komplexität der Kanten *von* und *nach* jeweils (0,n). Ein Arbeitsgang kann an mehreren und muß an mindestens einem Zustandsübergang beteiligt sein (1,n). Die funktionale Abhängigkeit in Bedingung <1.1> drückt aus, daß auch zwischen zwei Zuständen mehrere Übergänge (mit alternativen Arbeitsgängen) möglich sind. Über <1.2> werden rekursive Zustandsübergänge ausgeschlossen. Bedingung <1.3> stellt sicher, daß sowohl der Ausgangszustand und der Zielzustand als auch der beteiligte Arbeitsgang der Zustandsübergangskante zu dem gleichen Teil gehören. Ein Arbeitsplan muß genau einem Teil zugeordnet werden, da die für

die Definition von Arbeitsplänen benötigten Zustände auch eindeutig einem Teil zugeordnet sind. Dies führt zu einem Komplexitätsgrad von (1,1) zwischen dem Entitytyp *APL* und dem Beziehungstyp *APLZuo*. Soll wie bei den vorherigen Methoden ein Arbeitsplan für mehrere Teile Gültigkeit besitzen können, so müßte auch ein Zustand mehreren Teilen zugeordnet werden können und in den Integritätsbedingungen entsprechend berücksichtigt werden. Da dies einerseits bei dieser Darstellungsart nicht notwendig ist und andererseits zu einer erheblichen Verkomplizierung der Zustandsdefinitionen führen würde, soll darauf verzichtet werden.

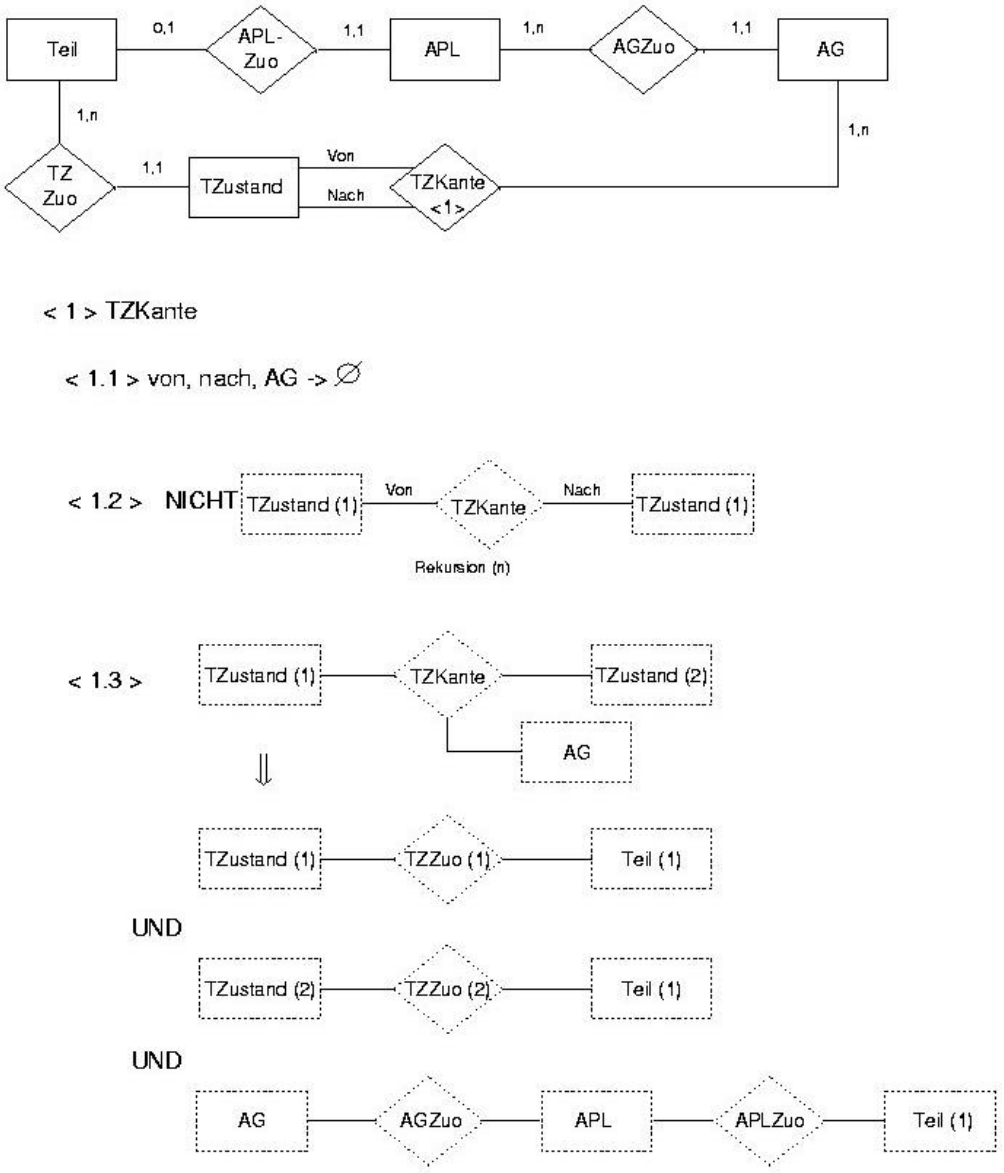


Abb. 95: PERM der Arbeitsplandarstellung mit Zustandsgraphen

Bewertung

Die Beurteilung der einzelnen Verfahren anhand verschiedener Kriterien zeigt Abbildung 96.

Kriterium \ Verfahren	isolierte Arbeitspläne	notwendige Vorgänger	Struktur-knoten	Zustands-orientierung
alternative Reihenfolge	+	++	++	++
alternative Arbeitsgänge	+	o	++	++
Bearbeitungspfadwechsel	--	-	+	+
1 Arbeitsgang mehrmals im Arbeitsplan	-	-	-	+
Splitting im Arbeitsplan	-	-	+	+
1 Arbeitsplan für mehrere Teile	o	o	o	-
Fertigungsüberwachung	o	o	+	++
Aufwand sequent. Arbeitsplan	+	-	-	o
Aufwand alternative Reihenfolge	o	+	+	o
Aufwand alternativer Arbeitsgang	o	o	+	+
Aufwand Bearbeitungspfadwechsel	--	-	+	+
mehrere Vorgänger-Arbeitsgänge	+	--	--	--
Bewertungsskala:	(++) = sehr gut; (+) = gut; (o) = mittel; (-) = schlecht; (--) = nicht möglich			

Abb. 96: Bewertung der Verfahren zur Strukturierung von Arbeitsplänen

Die Darstellung der Arbeitspläne mit Strukturknoten beinhaltet vollständig das Verfahren der notwendigen Vorgänger, da die Präzedenzmatrix den gleichen Informationsgehalt bietet wie der Beziehungstyp *notwVorg*. Zusätzlich stellt das Verfahren noch zwei weitere Matrizen bereit, so daß das Verfahren der notwendigen Vorgänger als Teilverfahren der Strukturknoten betrachtet werden kann.

Die Darstellung der **alternativen Abarbeitungsreihenfolge** ist prinzipiell in jedem Verfahren möglich.

Alternative Arbeitsgänge oder Arbeitsgangsequenzen können im Verfahren der notwendigen Vorgänger nur über einen neuen Arbeitsplan abgebildet werden. Da dies einen Methodenwechsel innerhalb des Verfahrens darstellt, wird das Kriterium mit der Note "mittel" (o) bewertet.

Nach Beendigung eines Teils der Arbeitsgänge wird ein **Wechsel des Bearbeitungspfades** durch isolierte Arbeitspläne nicht unterstützt. Eine Unterstützung durch das Verfahren der Definition notwendiger Vorgänger erfolgt nur für die Bearbeitungspfade innerhalb eines Arbeitsplans, d. h. bei alternativen Reihenfolgen.

Soll ein **Arbeitsgang mehrmals in einem Arbeitsplan** enthalten sein, wie z. B. ein Arbeitsgang "waschen", so ist dies mit isolierten Arbeitsplänen, dem Verfahren der Definition notwendiger Vorgänger sowie der Strukturknoten nur über ein Duplizieren des Arbeitsgangs möglich. Bei der zustandsorientierten Darstellung kann ein Arbeitsgang von mehreren Zustandsübergangskanten (Beziehungstyp *TZKante*) referenziert werden.

Ein **Splitting** von Arbeitsgängen als Standardvorgabe im Arbeitsplan ist nur über Strukturknoten und zustandsorientierte Darstellung möglich.

Die Zuordnung von **einem Arbeitsplan zu mehreren Teilen** ist bei der zustandsorientierten Darstellung nur möglich, wenn die Zustandsdefinition auch für mehrere Teile Gültigkeit besitzt. Auch bei den anderen Verfahren müssen, wie später noch gezeigt wird, die Teile mit gleichen Arbeitsplänen auch gleiche Stücklisten besitzen.

Die **Überwachung des Fertigungsfortschritts** ist bei den Verfahren der isolierten Arbeitspläne und über die Definition notwendiger Vorgänger möglich. Bei dem Verfahren der Strukturknoten kann mit Hilfe der drei beschriebenen Matrizen, einem Anforderungs- und einem Ausführungsvektor für jeden Auftrag eine detaillierte Fertigungsüberwachung erfolgen (Zörntlein 88). Durch die Zustandsdefinition des zustandsorientierten Verfahrens kann der Bearbeitungsfortschritt eines jeden einzelnen Teils sehr gut dokumentiert werden, was insbesondere für die Zwischenlagerung von unvollständig bearbeiteten oder geteilten Auftragslosen nützlich ist.

Der Aufwand für die Speicherung eines einzelnen, **sequentiell abzuarbeitenden Arbeitsplans** ist bei dem Verfahren mit isolierten Arbeitsplänen gering. Bei der zustandsorientierten Darstellung muß neben dem Endzustand noch pro Arbeitsgang ein Ausgangszustand definiert werden. Die Definition der notwendigen Vorgänger und analog dazu die Präzedenzmatrix der Strukturknoten führen dagegen bei langen sequentiellen

Arbeitsplänen zu einem erheblichen Aufwand. So müssen beispielsweise bei einem Arbeitsplan mit 15 Arbeitsgängen 105 Beziehungen des Typs *notwVorg* bzw. Matrix-Elemente für die Präzedenzmatrix mit den PWert von 1 gespeichert werden.

Der Aufwand für die Speicherung einer **alternativen Reihenfolge** ist bei isolierten Arbeitsplänen relativ hoch, eine Redundanz der Arbeitsgangbeschreibungen kann aber durch die Komplexität von (0,n) vom Typ *AG* zu dem Beziehungstyp *AGZuo* vermieden werden. Die zustandsorientierte Darstellung benötigt pro Knoten einer vertauschten Reihenfolge eine neue Zustandsdefinition.

Bei dem Verfahren der isolierten Arbeitspläne und der Definition notwendiger Vorgänger führt ein **alternativer Arbeitsgang** jeweils zu einem neuen Arbeitsplan, was einen gewissen Aufwand bedeutet.

Der **Wechsel von Bearbeitungspfaden** erfordert bei keinem Verfahren einen nennenswerten Zusatzaufwand, so daß sich die gleiche Bewertung ergibt wie bei der Möglichkeit des Bearbeitungspfadwechsels.

Bei isolierten Arbeitsplänen können in einem Arbeitsplan mehrere parallele Bearbeitungspfade abgebildet werden, die in einem Arbeitsgang (Montage) zusammenlaufen (**mehrere Vorgänger**). Dieser Sachverhalt wird aber üblicherweise über die Stückliste abgebildet.

Zusammenfassend läßt sich sagen, daß das Verfahren mit Strukturknoten und die zustandsorientierte Darstellung geeignete Methoden zur Abbildung flexibler Arbeitspläne bereitstellen. Die Flexibilität wird allerdings durch einen zusätzlichen Aufwand für die Definition der Matrizen bzw. Teilezustände erkauft. Da die Teilezustandsdefinitionen aber auch für die Materialverfolgung genutzt werden können, soll im Referenzmodell die zustandsorientierte Darstellung angewandt werden.

7.5 Chargendaten

Der Chargennachweis dient der Verfolgung des Materialflusses und der Dokumentation der Materialzusammensetzung eines Erzeugnisses. Eine Charge repräsentiert immer eine Menge eines bestimmten Materials des Entitytyps *Teil*. Damit hat die Charge als Ausprägung eines

Teils den Charakter einer Einzelressource im Gegensatz zu dem zur Ressourcengruppen generalisierten Entitytyp *Teil*.

Einzelne Stücke eines Teils können dann zu einer Charge zusammengefaßt werden, wenn sie als Rohteil aus der gleichen Lieferung eines Lieferanten oder als Baugruppe bzw. Endprodukt aus der gleichen Lieferung der Rohteile und dem gleichen Fertigungsprozeß stammen. Aus der eindeutigen Zuordnung einer Charge zu einem Teil ergibt sich für die Dokumentation der Materialzusammensetzung die Notwendigkeit des Verwendungs- bzw. Zusammensetzungsnachweises analog zu der Stücklistenstruktur für die Teile. Abbildung 97 a zeigt die Darstellung eines Gozintographen für ein Endprodukt P. Für die Produktion eines Stücks des Teils P werden 1 Stück des Teils B und 2 Stücke des Teils E benötigt. Zur Herstellung eines Stücks der Baugruppe B wird wiederum ein Stück des Teils E benötigt. Die Mengenangabe wird als Produktionskoeffizient bezeichnet.

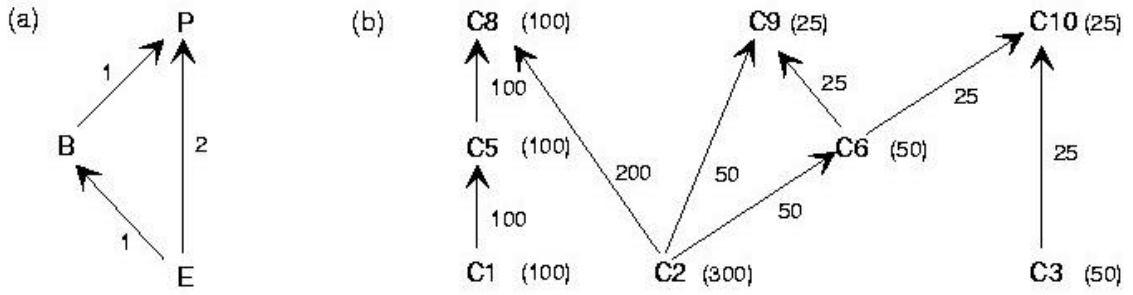


Abb. 97: Darstellung eines Chargenachweises im Vergleich zu einem Gozintographen

Es sollen 150 Stück des Endprodukts P hergestellt werden. Als Ausgangsmaterial stehen drei Chargen des Teils E mit 100 Stück (C1), 300 Stück (C2) und 50 Stück (C3) zur Verfügung. Abbildung 97 b zeigt eine mögliche Verwendung der Rohteilechargen mit der daraus folgenden Konsequenz für die Chargendifferenzierung auf der Baugruppen- und Endproduktebene. Für die Baugruppe B werden zwei Chargen C5 und C6 aus den Rohteilchargen C1 und C2 hergestellt. Diese werden im folgenden Bearbeitungsschritt zusammen mit der Rohteilcharge C3 und den Rest der Charge C2 zu den Endproduktchargen C8, C9 und C10 weiterverarbeitet.

Abbildung 98 zeigt die Darstellung im entsprechende Expanded ERM. Der bereits eingeführte Entitytyp *Teil* geht über den Beziehungstyp *Struktur* eine Verbindung mit sich selbst ein und stellt damit die Datenstrukturen zur Abbildung von Stücklisten bereit. Auf

eine Integritätsbedingung zum Ausschluß von rekursiven Teilebeziehungen über den Beziehungstyp *Struktur* wird verzichtet, da es bei bestimmten Produktionen, z. B. bei chemischen Prozessen, zu Zyklen in der Stückliste kommen kann (Scheer 90f, S. 111 - 113). Chargen werden als Entitytyp *Chg* abgebildet und über den Beziehungstyp *ChgTeil* eindeutig einem Typ *Teil* zugeordnet.

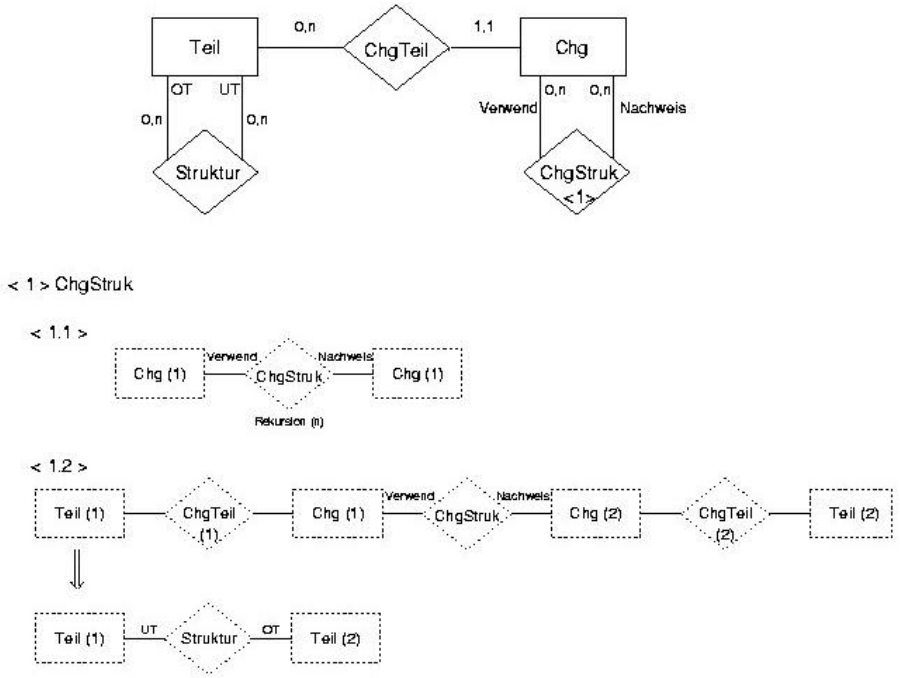


Abb. 98: PERM zu Chargendaten

Chargen sind ähnlich der Stückliste über einen Beziehungstyp *ChgStruk* (*Ch*arg*ens*t*ruk*tur) miteinander verbunden. Die Kante *Verwend* beschreibt die Verwendung einer Charge für die Produktion anderer Chargen, die Kante *Nachweis* die entsprechenden Ausgangschargen. Bezüglich einer Mengeneinheit einer Charge sind alle Beziehungen über die Kante *Nachweis* und-verknüpft, d. h. die Herstellung einer Mengeneinheit benötigt entsprechend dem Produktionskoeffizienten der korrespondierenden Stückliste Mengeneinheiten von **allen** referenzierten Chargen. Die Beziehungen der Kante *Verwend* sind oder-verknüpft, d. h. eine Mengeneinheit wurde nur für die Herstellung **einer** der referenzierten Chargen verwendet. Integritätsbedingung <1.1> verhindert Zyklen in der Chargenstruktur, die, im Gegensatz zu den Stücklisten, immer ausgeschlossen sind. Bedingung <1.2> stellt sicher, daß für alle verbundenen Chargen auch eine entsprechende Stückliste existiert.

7.6 Lager- und Transportdaten

Alle nichtstationären Produktionsressourcen müssen für die Fertigung bewegt, gehandhabt, gelagert und gepuffert werden. In der automatischen Fertigung werden diese Funktionen durch Lager- und Transportsysteme übernommen. Neben der Steuerung und Überwachung der Lagerung und des Transports können die Systeme auch gleichzeitig den Funktionen der Materialflussverfolgung dienen. Mit Hilfe dieser Systeme kann festgestellt werden, wo sich welches Material (Teile, Chargen, Werkzeuge, Vorrichtungen usw.) zur Zeit befindet.

Mit einem Transport wird Material von einer Station zu einer anderen bewegt. Aus Sicht der Materialflußverfolgung befindet sich das Material an einer Lokalität, d. h. entweder an einer Station oder auf einem Transport zwischen zwei Stationen. Eine Station kann eine Bearbeitungsstation (Maschine, Arbeitsplatz) oder ein Lager (z. B. Lagerplatz in einem Hochregallager, Bahnhof in der Werkstatt, Puffer vor der Maschine) sein. Damit ergibt sich eine Begriffshierarchie entsprechend Abbildung 99.

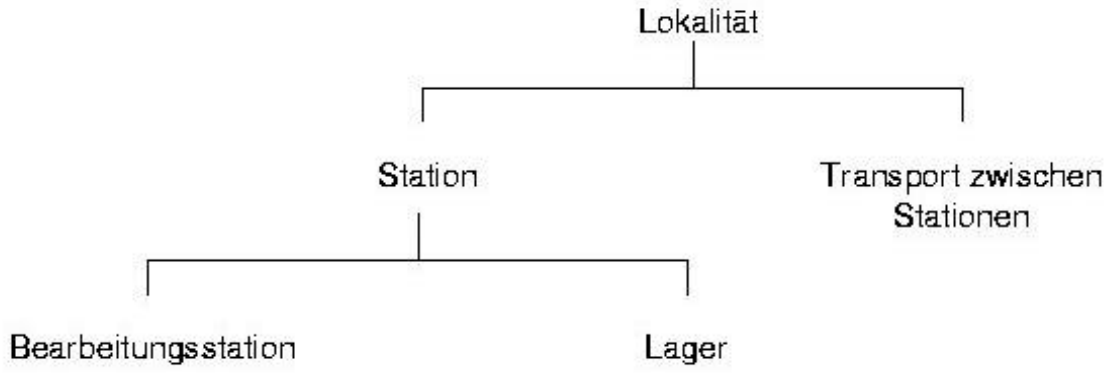


Abb. 99: Klassifizierung der Begriffe Lokalität, Station und Lager

Aufgrund des engen Zusammenhangs zwischen Lagerdaten und Transportdaten sollen die Datenstrukturen gemeinsam entwickelt werden. Entsprechend der Begriffshierarchie wird der Entitytyp *Station* als Generalisierung der Typen *Maschine*, *Pufferplatz* und *Lagerplatz* im Expanded ERM dargestellt (vgl. Abbildung 100).

Der Typ *Maschine* wurde bereits bei den Grunddaten eingeführt (vgl. Kapitel "Ressourcen", S. 125) und bezeichnet die möglichen Bearbeitungsstationen. Über den Typ *Pufferplatz* werden alle Lagerungsmöglichkeiten abgebildet, die nicht direkt zu einem Lagersystem gehören wie z. B. Pufferplätze für Warteschlangen vor der Maschine, Bahnhöfe in den Werkstätten usw.

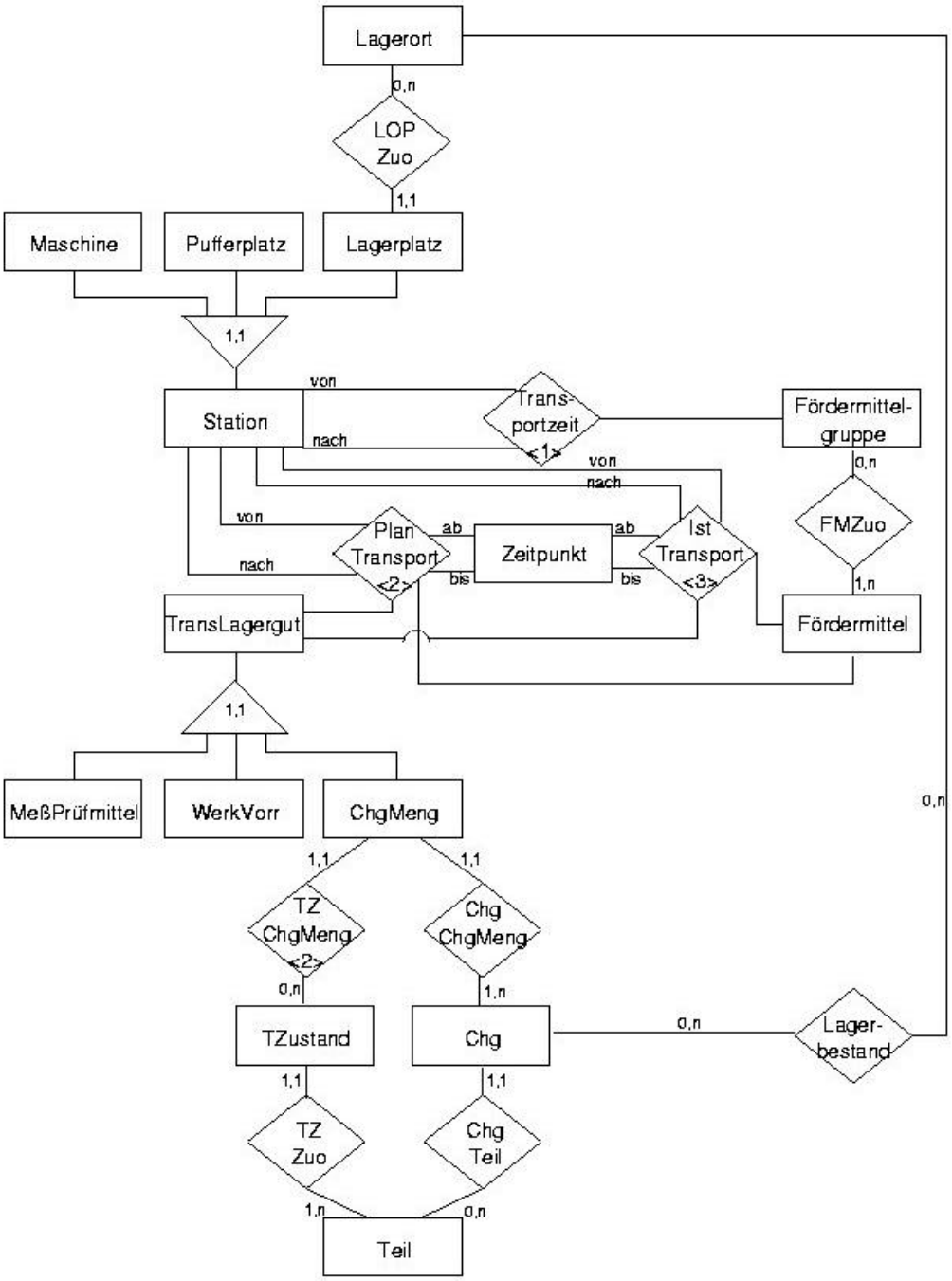
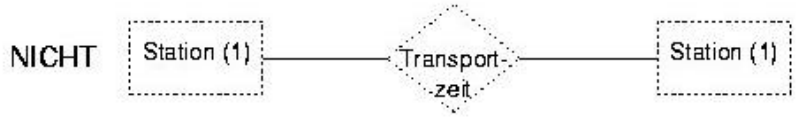


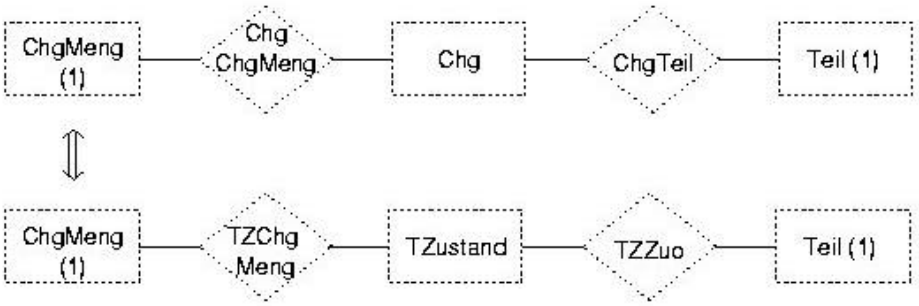
Abb. 100-1: PERM zur Lager- und Transportverwaltung

< 1 > Transportzeit

< 1.1 > von, nach Fördermittelgruppe -> ∅



< 2 > TZChgMeng



< 3 > PlanTransport

< 3.1 > TransLagergut, ab -> bis, von, nach, Fördermittel
Fördermittel, ab -> bis, von, nach, TransLagergut

< 3.2 > ab Zeitpunkt < bis Zeitpunkt

< 3.3 > ab Zeitpunkt ≥ max (bis Zeitpunkt) WHERE
ab Zeitpunkt < :ab Zeitpunkt AND
TransLagergut = :TransLagergut

< 3.4 > bis Zeitpunkt ≤ min (ab Zeitpunkt) WHERE
bis Zeitpunkt > :bis Zeitpunkt AND
TransLagergut = :TransLagergut

< 3.5 > ab Zeitpunkt ≥ max (bis Zeitpunkt) WHERE
ab Zeitpunkt < :ab Zeitpunkt AND
Fördermittel = :Fördermittel

< 3.6 > bis Zeitpunkt ≤ min (ab Zeitpunkt) WHERE
bis Zeitpunkt > :bis Zeitpunkt AND
Fördermittel sonst = :Fördermittel

< 3.7 > NICHT

Abb. 100-2: PERM zur Lager- und Transportverwaltung

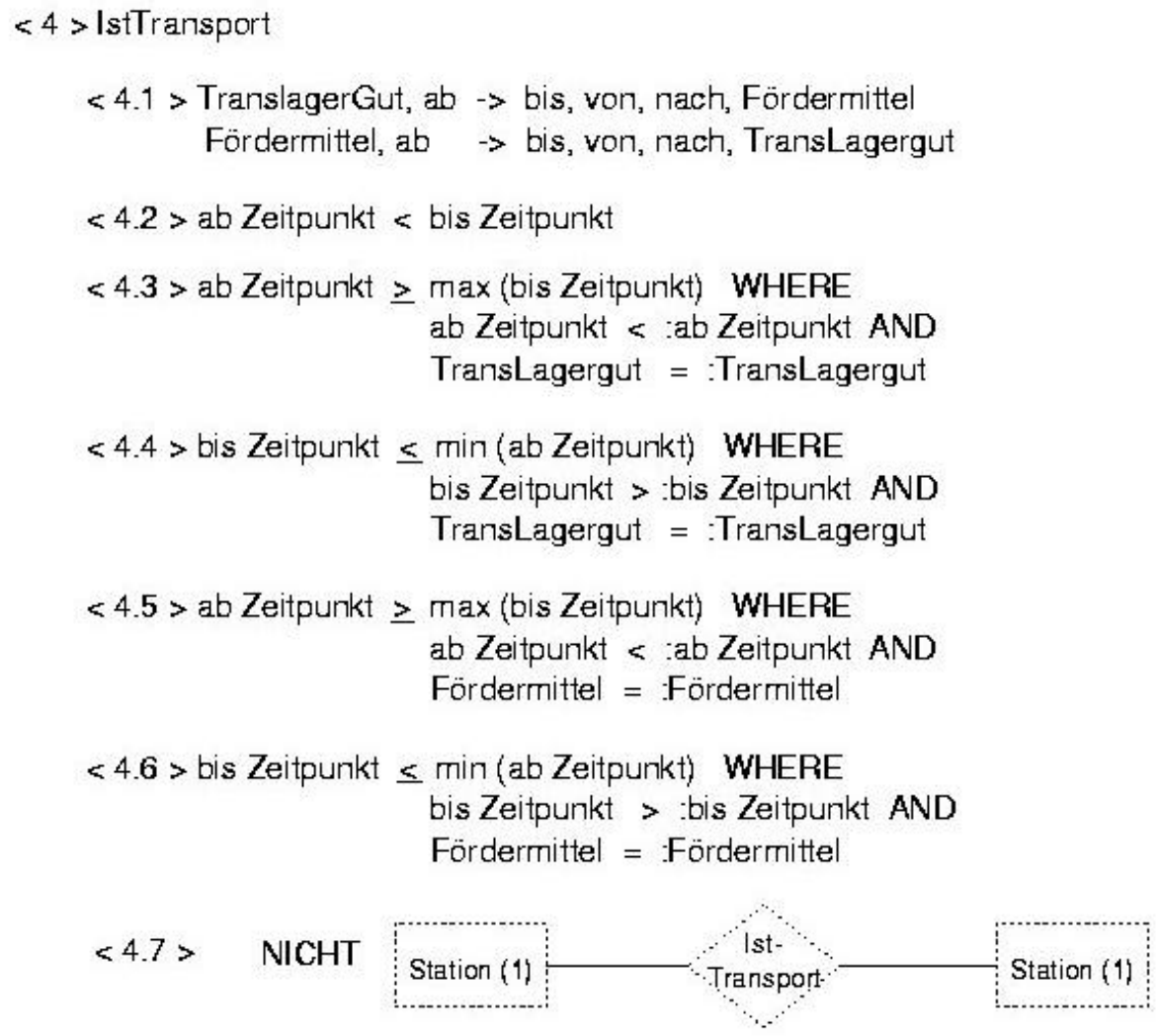


Abb.100 -3: PERM zur Lager- und Transportverwaltung

Lagerplatz repräsentiert die kleinste Lagerungseinheit in einem Lagersystem z. B. ein Fach oder einen Palettenplatz. Diese Lagerungseinheiten können über mehrere Hierarchiestufen zusammengefaßt werden, z. B. mehrere Plätze zu einem Lagerort, mehrere Orte zu einem Lagerraum usw. In Abbildung 100 wird eine Hierarchiestufe über den Beziehungstyp *LOPZuo* (*L*agerort-*P*latz*zuo*rdnung) zu dem Typ *Lagerort* dargestellt. Für den Transport zwischen zwei Stationen ist in Abhängigkeit des eingesetzten Transportmittels eine bestimmte Transportzeit notwendig. Die Transportzeiten werden u. a. für die Berechnung der Übergangszeiten zwischen zwei Bearbeitungsschritten oder der Planung von Transporten benötigt. Sie werden als Beziehungstyp *Transportzeit* zwischen den Typen *Station* mit der Kante *von* und der Kante *nach* sowie dem Entitytyp *Fördemittelgruppe* dargestellt. Integritätsbedingung <1> definiert zum einen die Art der Beziehung, zum anderen werden Transportzeiten zwischen der gleichen Station ausgeschlossen.

In dem Entitytyp *TransLagergut* (*Trans*port- und *Lagergut*) werden alle beweglichen Materialien als konkrete Transportstücke zusammengefaßt. Er bildet daher eine Generalisierung der bereits eingeführten Typen *WerkVorr* und *MeßPrüfmittel* sowie des Typs *ChgMeng*, der ein Teillos einer getrennt zu transportierenden oder zu lagernden Charge darstellt. Der Typ *ChgMeng* wird eindeutig einer Charge (Typ *Chg*) zugeordnet, eine Charge muß aus mindestens einem Teillos bestehen. Weiterhin muß die Teilmenge einer Charge sich genau in einem Teilezustand befinden. Dies ist z. B. notwendig, um eine ausgelagerte Menge einer teilfertigen Baugruppe eindeutig dem nächsten Bearbeitungsschritt zuführen zu können. Da sowohl der Teilezustand als auch die Charge ihrerseits eindeutig einem Teil zugeordnet sind, stellt Integritätsbedingung <2> sicher, daß eine Chargenmenge nur einer Charge und einem Teilezustand desselben Teils zugeordnet werden.

Ein geplanter Transport eines Transport- und Lagerguts wird über den Beziehungstyp *PlanTransport* abgebildet. Dazu werden neben dem Entitytyp *TransLagergut* die Typen *Fördermittel*, *Station* über die Kanten *von* und *nach* sowie *Zeitpunkt* über die Kanten *ab* und *bis* aggregiert. Sowohl aus Sicht des Entitytyps *TransLagergut* als auch des Typs *Fördermittel* ist der Beziehungstyp *PlanTransport* eine unstetige, überlappungsfreie Zeitdauer, da ein Gut zu einem Zeitpunkt nur einmal transportiert werden kann, und ein Fördermittel zu einem Zeitpunkt nur einmal befördern kann. Dies führt zu den Integritätsbedingungen <3>. Bedingung <3.1> zeigt, daß die Beziehung entsprechend der beiden Sichten zwei Determinanten besitzt. Die Bedingungen <3.2> bis <3.6> definieren die unstetigen, überlappungsfreien Zeitdauern der beiden Sichten; <3.7> verbietet geplante Transporte innerhalb der gleichen Station. Soll bei der Planung eines Transports von unbegrenzten Kapazitäten der Fördermittel ausgegangen werden, so würde die zweite Determinante sowie die Bedingungen <3.5> und <3.6> entfallen. Bereits realisierte Transporte werden über den Beziehungstyp *IstTransport* abgebildet, für die Integritätsbedingungen analog zu *PlanTransport* gelten (Integritätsbedingung <4>). Über den Beziehungstyp *IstTransport* kann sowohl die aktuelle Lager- als auch Umlaufbestandsmenge eines Teils ermittelt werden. Für die Bestandsführung ist es allerdings wünschenswert, gezielt auf die gesamte Lagermenge eines Teils zugreifen zu können. Dies kann über ein Attribut *Lagermenge* am Entitytyp *Teil* dargestellt werden, was bei jeder Lagerbuchung zu einer Änderung des Stammdatums führen würde. Deshalb wird ein Beziehungstyp *Lagerbestand* mit einem Attribut *Menge* zwischen den Typen *Chg* und *Lagerort* gebildet, mit der der Lagerbestand nach Chargen und Lagerort differenziert geführt werden kann.

7.7 Auftragsdaten

Entsprechend den Ausführungen im Kapitel "Arbeitsplandaten" (s. S. 134) werden Aufträge in einer zustandsorientierten Form dargestellt. Neben den bereits eingeführten Entitytypen *Teil*, *APL*, *AG* und *TZustand* und deren verbindenden Beziehungstypen wird der Entitytyp *TechnPrüfVerf* (*Techn*ische und *Prüf*ende *Verf*ahren) in das Expanded ERM der Abbildung 101 eingeführt, der alle im Betrieb zur Verfügung stehenden Fertigungs- und Prüfverfahren beschreibt (vgl. auch Scheer 90f, S. 164). Jeder Arbeitsgang wird über den Beziehungstyp *AGVerf* genau einem Entity des Typs *TechnPrüfVerf* zugeordnet, d. h. ein Arbeitsgang ist entweder ein Arbeitsschritt der Herstellung oder ein Prüfschritt der Qualitätssicherung.

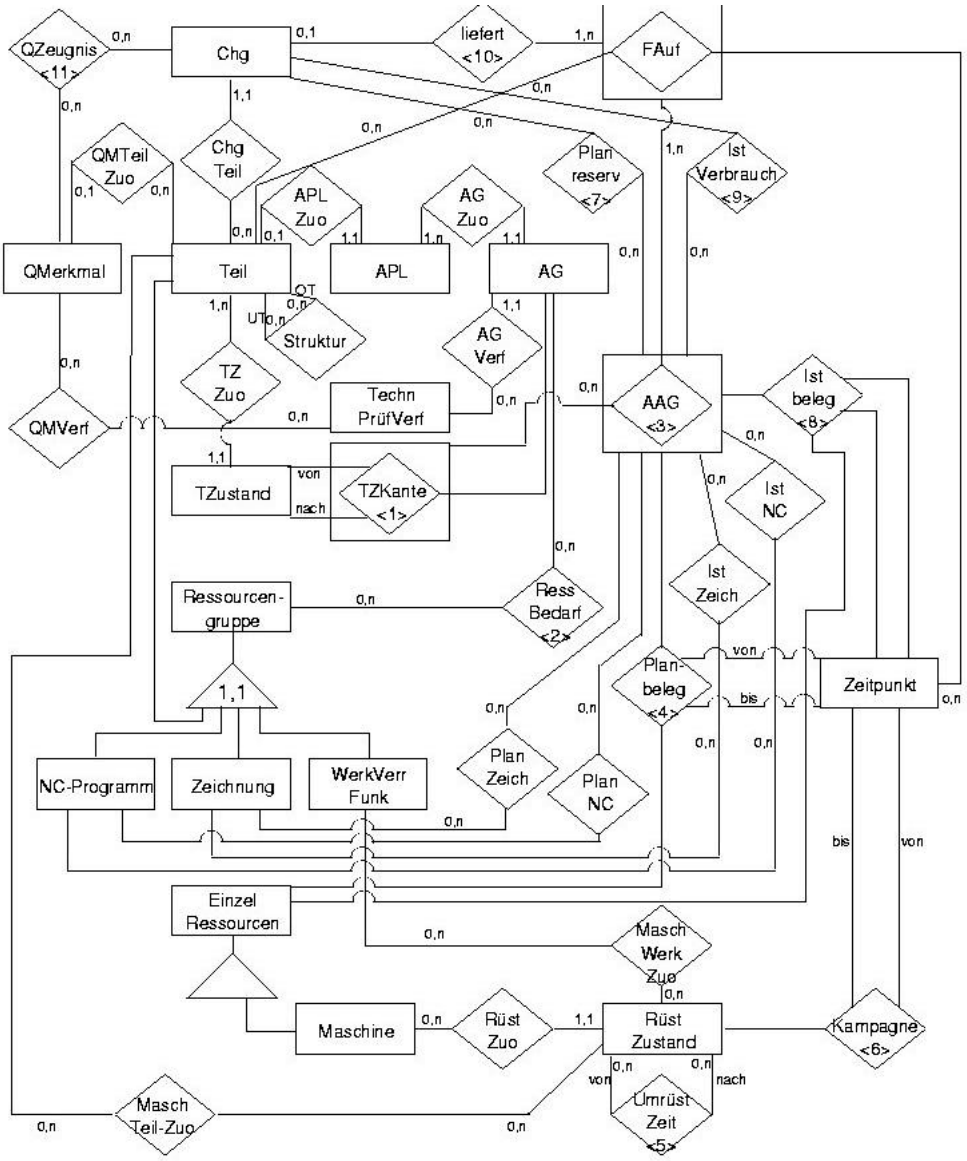


Abb. 101-1: PERM der Auftragsdaten

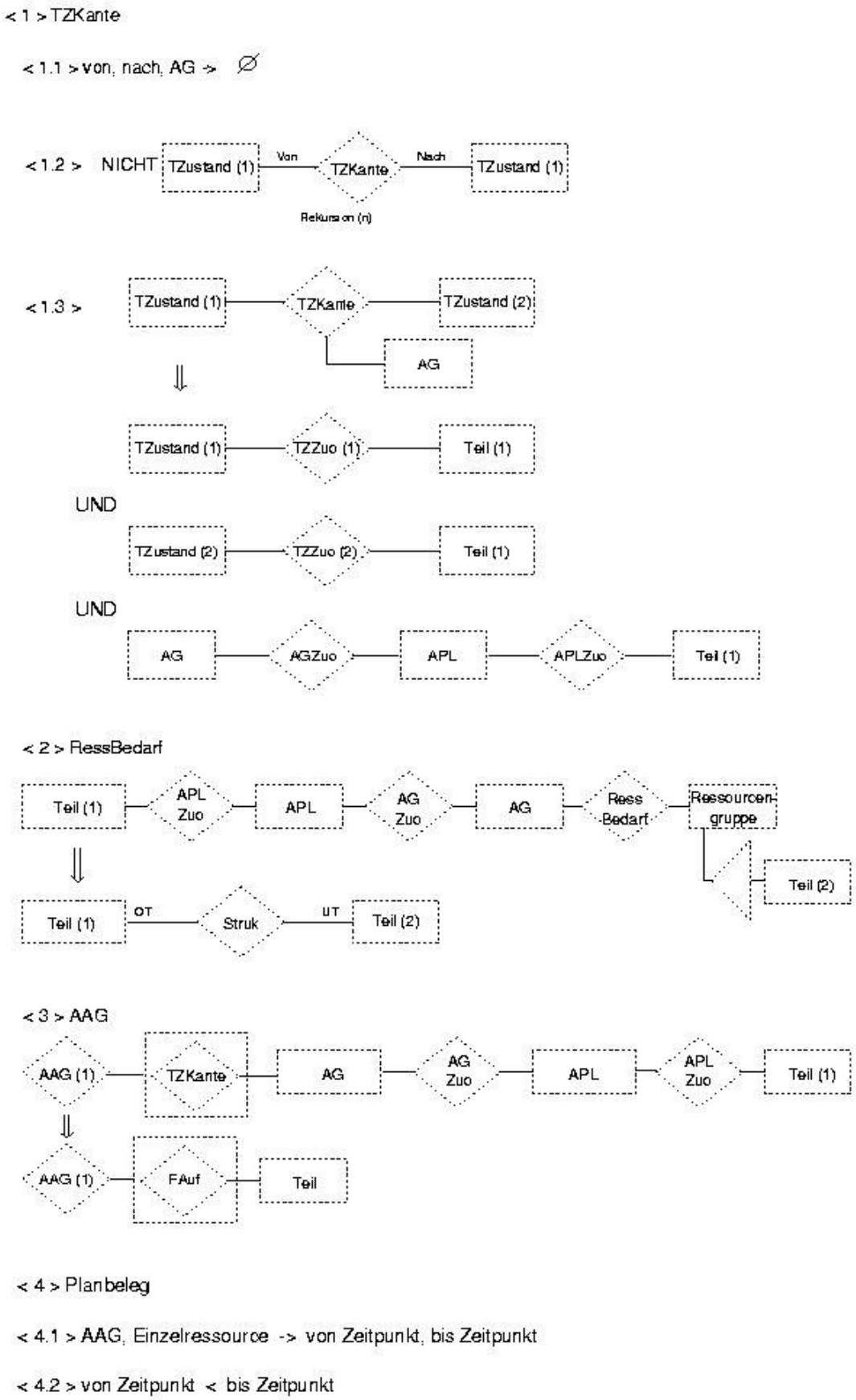


Abb. 101-2: PERM der Auftragsdaten

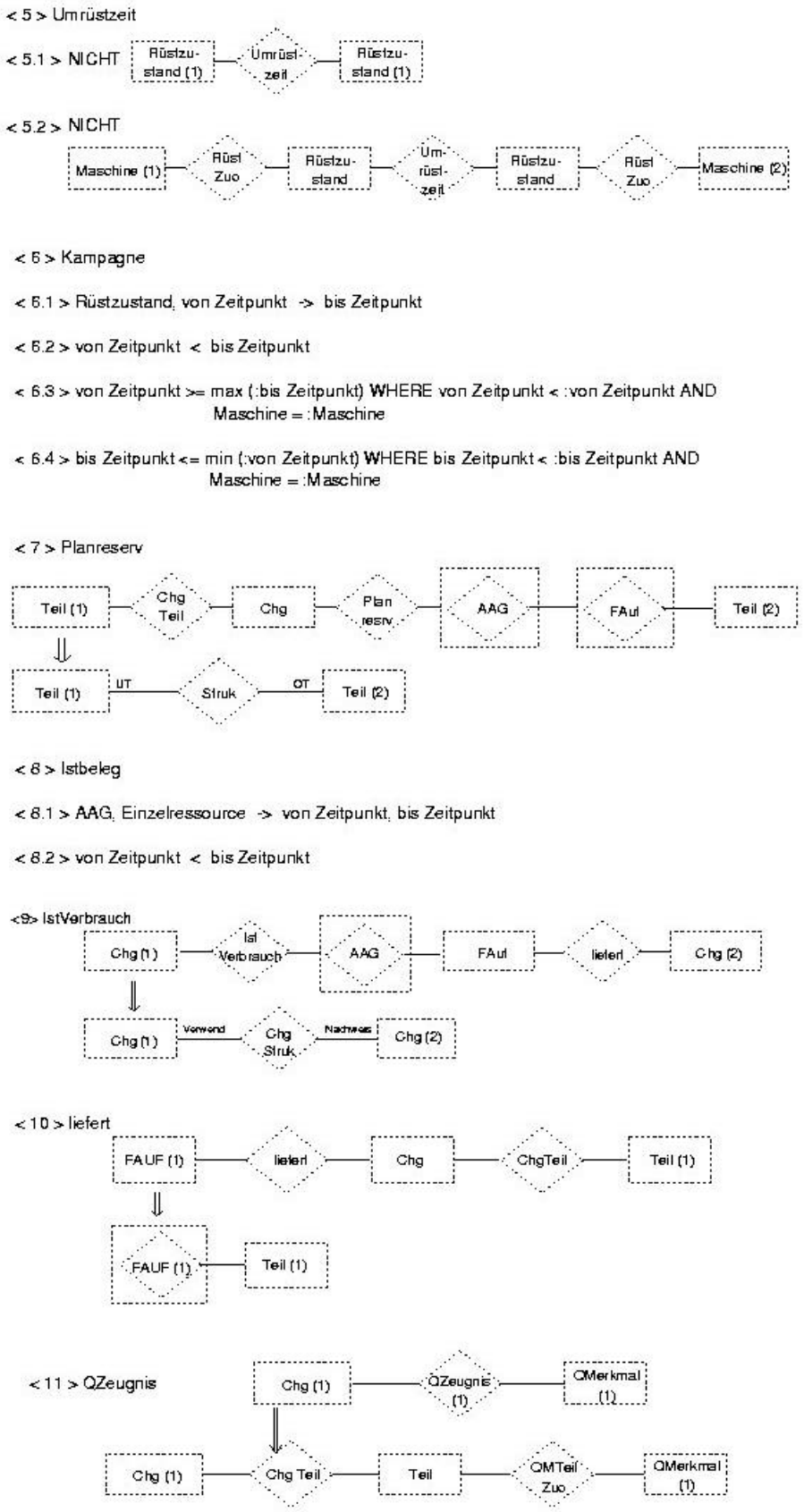


Abb. 101-3: PERM der Auftragsdaten

Qualitätsdaten

Für die Qualitätssicherung werden Qualitätsmerkmale, z. B. Toleranzen, Oberflächengüte, Gewicht, usw. definiert, die als Entitytyp *QMerkmal* (*Q*ualitäts*merkmal*) dargestellt werden. Jedes Qualitätsmerkmal kann über den Beziehungstyp *QMTeilZuo* (*Q*ualitäts*m*erkmal-*Teil*e-*Zuo*rdnung) mehreren Teilen zugeordnet, andererseits können jedem Teil auch mehrere Qualitätsmerkmale zugeordnet werden. Jedem Verfahren des Typs *TechnPrüfVerf* können über den Beziehungstyp *QMVerf* (*Q*ualitäts*m*erkmal-*Verf*ahrens-Zuordnung) Qualitätsmerkmale zugeordnet werden. Für den Beziehungstyp *TZKante* gelten die bereits in Abbildung 95 formulierten Integritätsbedingungen <1>.

Fertigungsaufträge

Jeder Arbeitsgang benötigt für die Ausführung bestimmte Ressourcen. Da als Stammdatum nur die Funktionalität der Ressource, nicht aber die konkrete Ressource von Interesse ist, wird der Typ *AG* über den Beziehungstyp *RessBedarf* mit dem Typ *Ressourcengruppe* verbunden. Üblicherweise wird ein Arbeitsgang als ein Arbeitsschritt an einer Maschinengruppe definiert, bei der ein oder mehrere Werkzeuge, Vorrichtungen, NC-Programme, Materialien usw. benötigt werden, was zu einer Komplexität von (0,n) für die Kante zwischen *AG* und *RessBedarf* führt. Über ein Attribut *Vorlaufzeit* des Beziehungstyps *RessBedarf* kann bestimmt werden, wann die entsprechenden Ressourcen vor dem eigentlichen Start des Arbeitsschritts benötigt werden. Für die Ressource *Maschine* ist dies üblicherweise die Rüstzeit, für die Ressourcen Werkzeug, Vorrichtung, NC-Programme, Zeichnungen sind dies die Bereitstellungszeiten. Auch für die benötigten Teile kann hier die Vorlaufzeit, in Gegensatz zu den eher groben Daten der Materialbedarfsplanung, arbeitsganggenau spezifiziert werden. Gleichzeitig muß über eine Integritätsbedingung sichergestellt werden, daß nur solche Teile als Ressourcenbedarf angegeben werden können, die über die Stücklistenstruktur als untergeordnete Teile definiert sind (Integritätsbedingung <2>).

Fertigungsaufträge werden im Rahmen der Materialwirtschaft für alle Primär- und Sekundärbedarfe gebildet, für die keine entsprechenden Lagermengen zur Verfügung stehen (Loos et al. 89). Sie werden als Beziehungstyp *FAuf* zwischen dem Typ *Teil* und *Zeit* abgebildet (vgl. auch Kapitel "Modellierung der Zeit", S. 119). Ein Fertigungsauftrag bezieht sich damit auf ein bestimmtes Teil, das zu einem bestimmten Zeitpunkt benötigt wird.

Ein Fertigungsauftrag wird nach einem Arbeitsplan abgearbeitet. Da bei der zustandsorientierten Arbeitsplandarstellung alle Abarbeitungsalternativen in einem Arbeitsplan hinterlegt sind, werden alle aus dem Arbeitsplan benötigten Arbeitsgänge über den Beziehungstyp *AAG* (*a*uftragsbezogener *A*rbeits*g*ang) zugeordnet. Um auch die Reihenfolge der Bearbeitungsschritte im Fertigungsauftrag zu führen, wird der Beziehungstyp *AAG* nicht zwischen dem Entitytyp *FAuf* und dem Entitytyp *AG*, sondern zwischen dem Entitytyp *FAuf* und dem zum Entitytyp uminterpretierten Beziehungstyp *TZKante* gebildet. Ein Fertigungsauftrag muß mindestens eine, *TZKante* kann eine beliebige Anzahl von Beziehungen *AAG* eingehen. Dabei muß allerdings sichergestellt sein, daß nur solche Arbeitsgänge einem Fertigungsauftrag zugeordnet werden, die für die Herstellung oder zur Qualitätssicherung des gewünschten Teils über den Arbeitsplan definiert wurden (Integritätsbedingung <3>).

Planungsdaten und Rüstzustände

Die im Rahmen der Stammarbeitspläne als *RessBedarf* hinterlegten Bedarfe an Ressourcengruppen, werden für Fertigungsaufträge auf Einzelressourcen spezifiziert, d. h. entsprechend der Bedarfe werden konkrete Maschinen, Werkzeuge, Vorrichtungen usw. zugeordnet. Dazu wird ein Beziehungstyp *Planbeleg* (*Planbeleg*ung) zwischen den auftragsbezogenen Arbeitsgängen und den Einzelressourcen gebildet. Desweiteren gehen die Anfangs- und Endzeitpunkte des Bedarfs als Kanten *von* und *nach* von dem Entitytyp *Zeitpunkt* aus in die Beziehung ein (Integritätsbedingung <4>). Da die Ressourcenbedarfe des Beziehungstyps *Bedarf* nicht direkt in den Beziehungstyp *Planbeleg* eingehen, ist es auch möglich, Belegungen von Ressourcen zu planen, die nicht im Stammarbeitsplan hinterlegt sind. Dies ist bsw. für die Ressourcenart Mitarbeiter wichtig, die aus betriebsrechtlichen Gründen in den wenigsten Fällen in den Arbeitsplänen geführt wird.

Die gewählte Abbildungsart der Fertigungsaufträge läßt eine weitestgehende Flexibilität der Planungsstrategie des Fertigungsablaufes zu:

(1) Gegenüber der Darstellung, in der ein Arbeitsgang genau einer Maschine bzw. Maschinengruppe zugeordnet ist, und andere Ressourcenarten als Komponentenbedarfe angefügt sind, sind in der gewählten Abbildungsart alle Ressourcenarten gleichberechtigt. Dies erleichert zum einem die Simultanplanung aller Ressourcen, und läßt zum anderen eine beliebige Auswahl der zu verplanenden Ressourcenarten zu. Bei der Simultanplanung werden neben der Planung der Maschinenkapazitäten auch gleichzeitig andere Ressourcenarten wie Werkzeuge und

Mitarbeiter verplant. Andererseits können bei bestimmten Fertigungsorganisationen wie Fertigungsinseln nicht die Maschinen die zu verplanenden Kapazitätseinheiten sein, sondern die Mitarbeiter, so daß zwischen den zu verplanenden Primärressourcen und den daraus abgeleiteten Sekundärressourcen unterschieden wird (vgl. Ruffing 90, S. 110 ff).

(2) Eine Einzelressource kann in der Regel zu einem Zeitpunkt einen Auftrag bearbeiten. Die Planbelegung wäre deshalb aus Sicht der Einzelressource eine unstetige, überlappungsfreie Zeitdauer, was zu entsprechenden Integritätsbedingungen führen würde. Es kann allerdings notwendig sein, eine Einzelressourcen gleichzeitig mit mehreren, unabhängige Aufträgen zu belegen:

- Aufgrund bestimmter Planungsstrategien kann eine Einzelressource ohne Kapazitätsbeschränkung verplant werden. Dies kann dazu führen, daß mehrere, sich zeitlich überschneidende Planbelegungen auf der Einzelressource entstehen. Diese können anschließend in einem Kapazitätsabgleich (vgl. auch Kinzer 71) geglättet oder über eine Freigabefunktion (vgl. auch Wiendahl 87) zugeteilt werden.

- Aufgrund technologischer Gegebenheiten sind bestimmte Einzelmaschinen in der Lage, mehrere Aufträge gleichzeitig zu bearbeiten. So kann z. B. eine Mehrspindeldrehmaschine entsprechend der Spindelanzahl mehrere Aufträge bearbeiten oder ein Glühofen entsprechend der Volumenkapazität verschiedene Teile unterschiedlicher Aufträge aufnehmen. Bei der Verplanung solcher Aggregate muß der Planungsalgorithmus prinzipiell zwischen der Belegungsdauer und der kapazitativen Möglichkeit unterscheiden.

Deshalb wird bei der allgemeinen Formulierung der Integritätsbedingung für den Beziehungstyp *Planbeleg* keine Periodenbedingung berücksichtigt (vgl. Integritäts-bedingung <4>).

(3) Das Splitten von auftragsbezogenen Arbeitsgängen ist durch die Bildung mehrerer zeitlich paralleler Planbelegungen auf unterschiedlichen Maschinen möglich.

(4) Ausweichmaschinen, alternative Arbeitsgänge oder alternative Abarbeitungsfolgen sind über die zustandsorientierte Darstellung jederzeit möglich, auch nachdem bereits ein Teil der Arbeitsgänge abgearbeitet worden ist. Desweiteren ist ein Splitten

der Auftragsmenge und Bearbeiten der Teillose über verschiedene Bearbeitungspfade oder -alternativen möglich. Dazu werden aus dem Stammarbeitsplan die entsprechenden Arbeitsgänge und Teilezustandskanten herangezogen.

(5) Die Abbildung überlappender Fertigung, d. h. das zeitliche Übereinanderschieben von auftragsbezogenen Arbeitsgängen eines Fertigungsauftrags, ist über die Anfangs- und Endzeitpunkte (Kanten *von* und *bis* des Beziehungstyps *Planbeleg*) unter Berücksichtigung der Transportzeiten (Beziehungstyp *Transportzeit*) möglich.

(6) Das Raffen von auftragsbezogenen Arbeitsgängen dient der Einsparung von Rüstzeiten an einer Maschine. Dazu werden solche Aufträge hintereinander gefertigt, die ein ähnliches Rüsten erfordern. Verschiedene Rüstzustände können in dem Entitytyp *Rüstzustand* abgebildet werden. Ein Rüstzustand muß eindeutig einer Maschine zugeordnet werden. Rüstzustände sind gekennzeichnet durch bestimmte Werkzeuge bzw. Vorrichtungen oder Teile, für die die Maschine vorbereitet wird. Dies wird über die Beziehungen *MaschWerkZuo* (*Masch*inen-*Werk*zeug-*Zuo*rdnung) zu dem Entitytyp *WerkVorr* und *MaschTeilZuo* (*Masch*inen-*Teil*-*Zuo*rdnung) zu dem Entitytyp *Teil*, wobei jeweils beliebig viele Zuordnungen möglich sind, abgebildet. Die Umrüstzeiten von einen Rüstzustand in einen anderen werden als Umrüstmatrix in dem Beziehungstyp *Umrüstzeit* mit den Kanten *von* und *nach* hinterlegt. Integritätsbedingung <5> stellt sicher, daß nur Umrüstzeiten für Zustände der gleichen Maschine definiert werden. Zum anderen werden Rüstzeiten zwischen den gleichen Zuständen ausgeschlossen.

(7) Für die Serienfertigung ist das Fertigen in Kampagnen üblich. Dabei werden bestimmte Serien (d. h. ein Teilespektrum) für eine gewisse Zeitdauer aufgelegt, die danach von anderen Serien abgelöst werden. Eine Serie kann als Rüstzustand interpretiert werden, dem alle zu einer Serie gehörenden Teile zugeordnet sind. Die Kampagne kann damit als Beziehung zwischen einem Rüstzustand und dem Entitytyp *Zeitpunkt* mit den Kanten *von* und *nach* dargestellt werden, wobei eine Kampagne aus der Sicht der Maschine (bzw. eines Rüstzustands) eine unstetige, überlappungsfreie Zeitdauer darstellt (Integritätsbedingung <6>).

(8) Ein Auftragsmix an einer Maschine kann über die Möglichkeiten des Splittens und Raffens abgebildet werden. Dabei werden mehrere auftragsbezogene Arbeitsgänge in kleine Bearbeitungseinheiten aufgeteilt und in einem Mix hintereinandergelegt, wobei für die Maschine nur zu Beginn unproduktive Rüstzeiten anfallen.

Die Planbelegungen für einen auftragsbezogenen Arbeitsgang erfassen nur die Ressourcen, die zu dem Typ *Einzelressource* generalisiert wurden. NC-Programme und Zeichnungen werden jeweils über die Beziehungen *PlanNC* (ge*plan*tes *NC*-Programm) und *PlanZeich* (ge*plan*te *Zeich*nung) dem Entitytyp *AAG* zugeordnet. Für den Teilebedarf werden über den Beziehungstyp *Planreserv* (*Planreserv*ierung) bestimmte Chargen für einen Arbeitsgang reserviert. Hierbei gelten die gleichen Integritätsbedingungen wie für den Ressourcenbedarf, d. h. die Planreservierung einer Charge ist nur dann möglich, wenn entsprechende Teile in der Baukastenstückliste des zu fertigenden Teiles enthalten sind (Integritätsbedingung <7>).

Die beschriebenen Beziehungen der Planung können durch eine Planungshierarchie erweitert werden, so daß für jede Hierarchiestufe eigene Planbelegungsbeziehungen angelegt werden könnten, z. B.:

- In Rahmen einer mittelfristigen Planung werden nur Maschinenkapazitäten und Teile verplant, wobei die Maschinen nur auf Gruppenebene geplant werden.

- In einer Feinplanung werden alle Ressourcenarten, also auch Personal und Werkzeug auf Ebene der Einzelressourcen geplant.

- In einer Feinstplanung werden die auftragsbezogenen Arbeitsgänge endgültig den Ressourcen zugeordnet. Auf dieser Ebene könnte auch die Transportplanung sowie die Bildung von Auftragsmix für ein Bearbeitungszentrum erfolgen.

Istdaten

Der tatsächlich realisierte Fertigungsablauf wird in eigenen Beziehungen dokumentiert. Für die Einzelressourcen wird ein Beziehungstyp *Istbeleg* (*Istbeleg*ung) zwischen den Typen *AAG*, *Einzelressource* sowie *Zeitpunkt* mit den Kanten *von* und *bis* mit den Integritätsbedingungen <8> analog zu dem Beziehungstyp *Planbeleg* angelegt. Die Istbelegung kann entsprechend der Ressourcenart u.a. für folgende Zwecke dienen:

- Mit der Angabe der Maschine, auf der tatsächlich gefertigt wurde, sowie der tatsächlich benötigten Belegungsdauer kann über den entsprechenden Kostensatz eine durchgeführt werden.

- Die Istbelegung der Mitarbeiter kann für die Bruttolohnfindung genutzt werden.

- Ressourcen wie Maschinen oder Werkzeuge unterliegen einem Verschleiß. Über die Summe aller Istbelegungen und der realisierten Leistungsgrade kann die genutzte Kapazität ermittelt werden. Daraus können Wartungszeitpunkte für Maschinen bzw. en von Werkzeugen abgeleitet werden.

Die bei der Fertigung genutzten NC-Programme und Zeichnungen können über den Beziehungstyp *IstNC* und *IstZeich* dem Arbeitsgang zugeordnet werden. Für die genutzten Rohstoffe wird ein Beziehungstyp *IstVerbrauch* zwischen den Typen *AAG* und *Chg* gebildet. Entsprechend der Planreservierung wird die Integritätsbedingung <9> formuliert. Ein fertiggestellter Auftrag liefert als Ergebnis eine oder mehrere Chargen (Beziehungstyp *liefert*), wobei wegen der zugrundeliegenden Chargendefinition eine Charge aus höchstens einem Fertigungsauftrag entstammen kann. An die Beziehung wird zum einen die Bedingung gestellt, daß die gelieferten Chargen auch dem herzustellenden Teil des Fertigungsauftrags entsprechen, zum anderen muß die Chargenstruktur der Chargenverfolgung (Beziehungstyp *ChgStruk*) mit dem Istverbrauch an Chargen für das Rohmaterial und den Zielchargen des Fertigungsauftrags übereinstimmen (Integritätsbedingungen <9> und <10>).

Die bei der Fertigung realisierten Qualitäten werden für jede Charge in einem Qualitätszeugnis dokumentiert. Ein Zeugnis stellt damit eine Bewertung einer Charge entsprechend der für das Teil definierten Qualitätsmerkmale dar. Es wird als ein Beziehungstyp *QZeugnis* (*Q*ualitäts*zeugnis*) zwischen den Typen *Chg* und *QMerkmal* abgebildet. Über Integritätsbedingung <11> wird sichergestellt, daß nur solche Qualitätsmerkmale in das Zeugnis aufgenommen werden, die für das Teil der Charge gültig sind.

Die streng formulierten Integritätsbedingungen lassen nur die Abbildung von Auftragsdaten und realisierten Werten in den *Ist*-Beziehungen zu, die den Stammdaten entsprechen. Im Fertigungsbereich kann aber davon ausgegangen werden, daß der tatsächliche Ablauf nicht korrekt entsprechend den vorgegebenen Werten erfolgt. So ist es möglich, daß für die Produktion einer Charge Rohstoffe verwendet werden, deren Teile nicht in der Stückliste definiert sind. Um solche Zustände abbilden zu können, kann es sinnvoll sein, bestimmte Integritätsbedingungen zu lockern.

Auftragsnetze

Die als Ausgangsmaterial für die Bearbeitung eines Fertigungsauftrags notwendigen Rohstoffe oder Baugruppen sind als Ressourcenbedarf in neutraler Form im Stammarbeitsplan hinterlegt bzw. als Planreservierung konkreter Chargen den auftragsbezogenen Arbeitsgängen zugeordnet. Die Materialien der Planreservierungen können entweder als Rohstoff eingekauft werden oder als Baugruppe selbst wieder über Fertigungsaufträge gedeckt werden. Häufig wird dieser Zusammenhang über eine Lagerzugangsbuchung nach Fertigstellung der untergeordneten Baugruppe und eine Lagerausgangsbuchung als Kommisionierung für den Bedarf der übergeordneten Baugruppe abgewickelt. Dabei geht der Zusammenhang zwischen den untergeordneten und den übergeordneten Fertigungsaufträgen verloren, obwohl die übergeordneten Fertigungsaufträge als Bedarfsverursacher ursächlich die Generierung der untergeordneten Fertigungsaufträge bewirkt haben. Bei Kundeneinzelfertigung und Kleinserienfertigung mit großer Fertigungstiefe ist das Erkennen der Zusammengehörigkeit für die optimale Steuerung der Fertigung wichtig (vgl. auch Scheer 90f, S. 139):

- Für eine kurzfristige und dennoch fristgerechte Herstellung ist eine einheitliche Terminierung über alle Fertigungsstufen wünschenswert. So können die exakten Bedarfstermine der Materialbedarfe eines übergeordneten Fertigungsauftrags Ausgangspunkt für die Terminierung der Fertigungsaufträge der untergeordneten Teile bilden.

- Bei Störungen und Verzug von Fertigungsaufträgen können unmittelbar die betroffenen übergeordneten Fertigungsaufträge und damit auch die dahinterstehenden Kundenaufträge ermittelt werden.

- Nach Fertigstellung eines Fertigungsauftrags kann die hergestellte Baugruppe direkt dem Bedarfsverursacher zugesteuert werden. Damit entfällt der Umweg über das Lager. Durch diese zeitgerechte Materialzusteuerung kann die Durchlaufzeit verringert werden.

Die Zusammengehörigkeit der Fertigungsaufträge läßt sich aus der Stücklistenstruktur der Teile ableiten. Abbildung 102 zeigt die Struktur der Auftragsbeziehungen, die als Auftragsnetz bezeichnet werden soll. In Teil a ist eine Stücklistenstruktur mit Produktionskoeffizienten dargestellt. Teil b zeigt Fertigungsaufträge mit Auftragsmengen zu den einzelnen Teilen aus der Stückliste sowie die Beziehungen als Netz. Die an die Netzstrukturen angefügten Mengenangaben verdeutlichen, wieviel Mengeneinheiten von

dem vorgelagerten Fertigungsauftrag in den nachgelagerten Fertigungsauftrag übernommen werden.

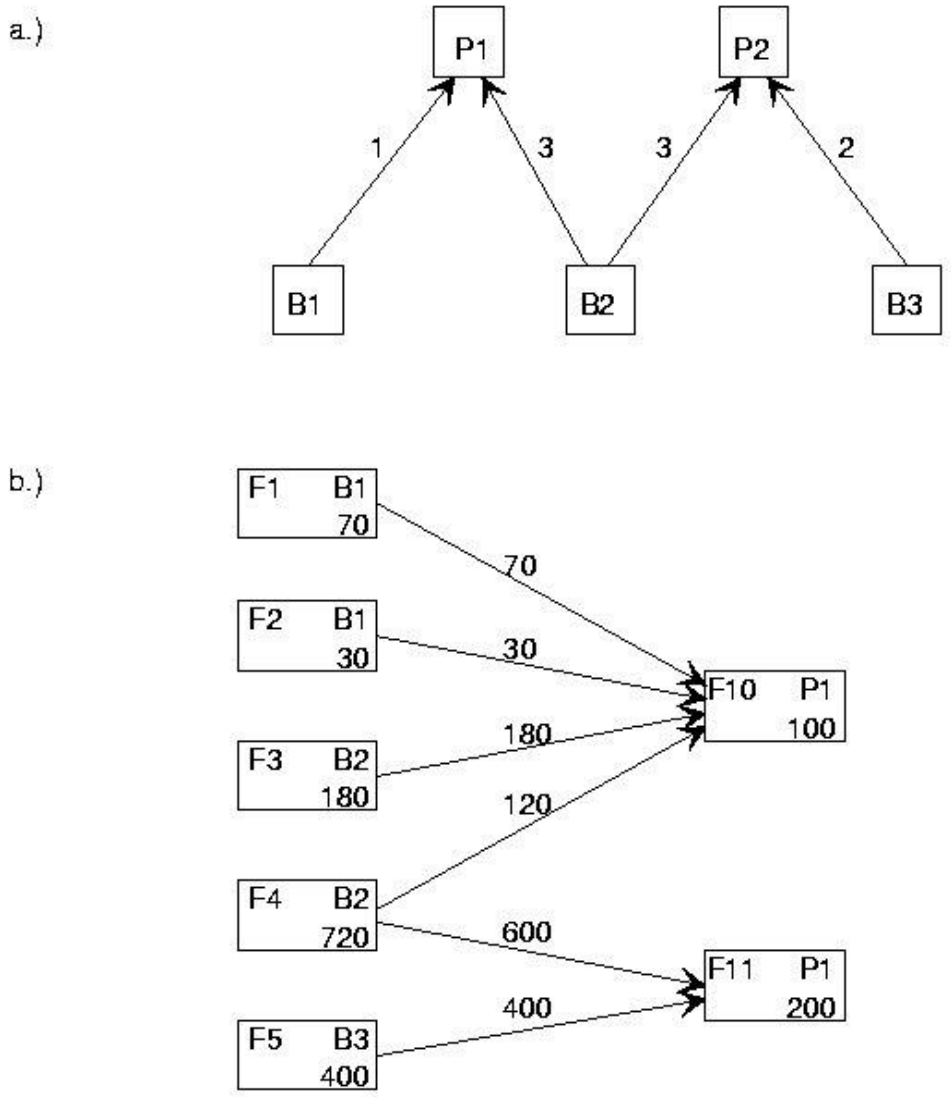


Abb. 102: Ableitung des Auftragsnetze aus der Stückliste

Die Auftragsstrukturen sind wie folgt zu interpretieren:

- Aus Sicht eines vorgelagerten Auftrages werden die Mengen auf die nachfolgenden Aufträge aufgeteilt. Auftrag F4, der das Teil B2 in einer Menge von 720 produziert, versorgt den Fertigungsauftrag F10 mit 120 Mengeneinheiten und Fertigungsauftrag F11 mit 600 Mengeneinheiten.

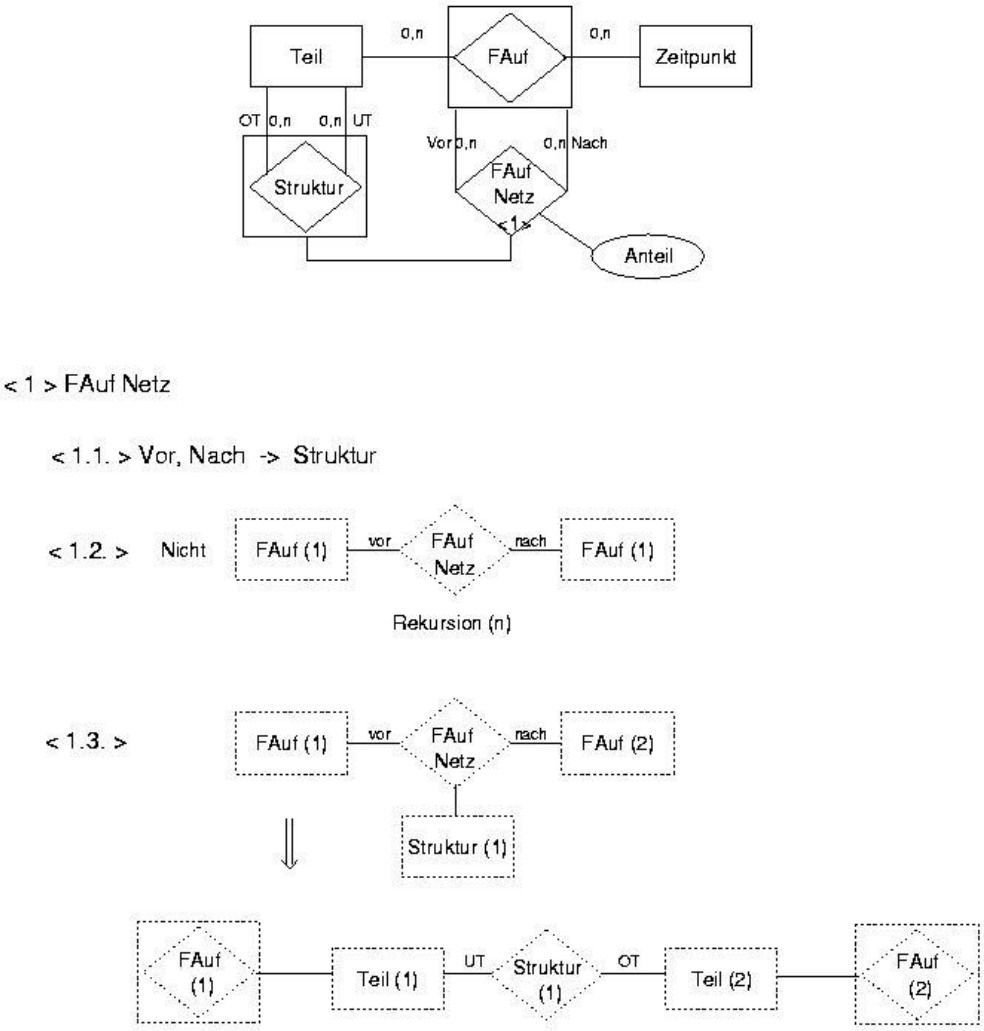


Abb. 103: Auftragsnetze im PERM

- Aus Sicht eines nachgelagerten Auftrages können die Mengen der vorgelagerten Aufträge bezüglich einer Mengeneinheit und- bzw. oder-verknüpft sein:

  -- Oder-verknüpfte Beziehungen decken den gleichen Teilebedarf. Daraus folgt, daß die vorgelagerten Aufträge das gleiche Teil liefern. Auftrag F1 sowie Auftrag F2 liefern das Teil B1 an den Auftrag F10. Für die Fertigung eines Teils P1 wird entweder ein Teil B1 aus den Auftrag F1 oder ein Teil B1 aus dem Auftrag F2 benötigt.

  -- Und-verknüpfte Beziehungen decken unterschiedliche Teilebedarfe entsprechend der Stückliste. Dies bedeutet, daß unterschiedliche Materialien

zusammengeführt werden. Auftrag F4 liefert das Teil B2, Auftrag F5 das Teil B3 an den Auftrag F11. Solche Vorgänge sind typisch für Montageprozesse.

Über die Strukturbeziehung der Teile der Fertigungsaufträge können die Und- sowie Oder-Verknüpfungen differenziert werden. Oder-verknüpfte Auftragsbeziehungen verweisen auf die gleiche Strukturbeziehung, während und-verknüpfte Auftragsbeziehungen auf verschiedene Strukturbeziehungen des gleichen übergeordneten Teils verweisen.

Abbildung 103 zeigt die Darstellung im Expanded ERM. Der Beziehungstyp *FAufNetz* bildet die Auftragsbeziehung als Netz ab, wobei die Kante *Vor* die vorgelagerten und die Kante *Nach* die nachgelagerten Fertigungsaufträge referenzieren. Gleichzeitig verweist eine Auftragsbeziehung auf eine Stücklistenstruktur, mit der die Art der Verknüpfung ausgedrückt wird. Alle Kanten des Beziehungstyps *FAufNetz* besitzen eine Komplexität von (0,n). Das Attribut *Anteil* beschreibt den Anteil der Menge des vorgelagerten Auftrags, der zur Deckung des nachgelagerten Auftrags genutzt wird. Integritätsbedingung <1.1> beschreibt die Determinante des Beziehungstyps, und Bedingung <1.2> schließt Rekursionen im Auftragsnetz aus, die im Gegensatz zur Stückliste auf jeden Fall unzulässig sind. Bedingung <1.3> stellt sicher, daß nur solche Fertigungsaufträge im Netz verbunden sind und auf eine Stückliste verweisen, deren herzustellende Teile auch über den gleichen Beziehungstyp *Struktur* als Stücklistenstufe definiert sind.

7.8 Instandhaltungsdaten

Die Bedeutung der Wartung des Maschinenparks wird mit wachsendem Automatisierungsgrad wichtiger, da sich die menschliche Arbeitsleistung in der Fertigung zunehmend von der reinen Produktion hin zur Überwachung und Instandhaltung des Fertigungsablaufs entwickelt. Neben der Beseitigung von Betriebsstörungen gehört die vorbeugende Planung und Durchführung von Wartungsmaßnahmen zum Aufgabengebiet der Instandhaltung (Scheer 74).

Obwohl die Planung der präventiven Instandhaltung starke Parallelen zur Planung von Fertigungsaufträgen aufweist, werden aufgrund einiger Abweichungen in dem vorliegenden Referenzmodell eigene Datenstrukturen entworfen:

- Bei der Definition von Arbeitsplänen wurde, aufgrund der günstigen Darstellungsmöglichkeiten der Materialflußproblematik, eine zustandsorientierte

Darstellungsform gewählt. Eine entsprechende Definition von Wartungsabarbeitungszuständen der Maschinen, nicht zu verwechseln mit dem Beziehungstyp *Zustand* zwischen den Entitytypen *Einzelressource* und *Status* (s. S. 134), erscheint wenig sinnvoll.

- Ein Fertigungsauftrag wird in der Regel auf mehreren Maschinen abgearbeitet. Bei der Generierung des Fertigungsauftrages ist noch nicht bekannt, auf welchen Einzelmaschinen die Bearbeitungsschritte (Arbeitsgänge) gefertigt werden sollen. Die exakte Zuordnung erfolgt erst im Rahmen eines (Fein-) Planungsschritts. Dagegen bezieht sich ein gesamter Wartungsauftrag auf eine Einzelmaschine, deren Identität bereits bei der Generierung des Auftrags bekannt ist.

- Bei der Definition von Fertigungsaufträgen wurde die Chargenproblematik berücksichtigt und in die Datenstrukturen integriert. Für Wartungsaufträge existieren hierfür keine entsprechenden Komponenten.

- Einzelne Wartungsschritte innerhalb einer Wartungsmaßnahme können häufig unabhängig voneinander parallel durchgeführt werden, z. B. die Wartung eines Getriebes und der Elektronik.

Abbildung 104 zeigt das Expanded ERM mit den Datenstrukturen zur Instandhaltung. Eine *Maschine* kann über einen Beziehungstyp *MTypzuo* (*M*aschinen*typzuo*rdnung) eindeutig einem *Maschinentyp* zugeordnet werden. Ein Maschinentyp repräsentiert alle konstruktionsgleichen (nicht funktionsgleichen!) Maschinen, für die gleiche Wartungsmaßnahmen definiert werden können. Jede Maschine besteht aus Maschinenteilen (Entitytyp *Maschinenteil*), die über eine Stückliste als Beziehungstyp *MTStruk* (*M*aschinen*t*eile*struk*tur) mit den Kanten *OMT* (*O*ber-*M*aschinen*t*eil) und *UMT* (*U*nter-*M*aschinen*t*eil) analog zur Teilestückliste verbunden sind (vgl. Scheer 90f, S. 303). Über Integritätsbedingung <1> wird ein Zyklus innerhalb der Stückliste ausgeschlossen. Jeder Maschinentyp wird über den Beziehungstyp *MTypTeil* einem Maschinenteil zugeordnet, wobei nur solche Maschinenteile zulässig sind, die am Anfang einer Stücklistenstruktur stehen (Integritätsbedingung <2>). Für jeden Maschinentyp können Wartungspläne (*WPL*) definiert werden, die sich im Wartungsintervall, den betroffenen Maschinenteilen usw. unterscheiden können. Zur Durchführung der Wartungsmaßnahmen stehen, analog zu den technischen und prüfenden Verfahren, Wartungverfahren (*WVerfahren*) wie Schmieren, Toleranzprüfung, Funktionsprüfung oder Verschleißteilwechsel zur Verfügung. Über den Beziehungstyp *WSchritt* (*W*artungs*Schritt*) werden einem Wartungsplan im Zusammenhang mit den betroffenen Maschinenteilen die

Wartungsverfahren zugeordnet. Neben den funktionalen Abhängigkeiten der Beziehung stellt Integritätsbedingung <3> sicher, daß nur solche Maschinenteile von einem Wartungsschritt referenziert werden, die in der Stückliste der Maschine des Wartungsplans enthalten sind. Die Reihenfolge der einzelnen Wartungsschritte innerhalb eines Wartungsplans wird über den Beziehungstyp *WSchrittNetz* mit den Kanten *VWS* (*V*orheriger *W*artungs*S*chritt) und *NWS* (*N*achfolgender *W*artungs*S*chritt) definiert. Durch die Definition der Reihenfolge als Netz können in einem Wartungsplan voneinander unabhängige Folgen von Wartungsschritten definiert werden. Innerhalb des Netzes dürfen keine Rekursionen auftreten (Integritätsbedingung <4>). Über den Beziehungstyp *WBedarf* (*W*artungs*Bedarf*) werden alle für die Wartung notwendigen Ressourcen wie bsw. Werkzeuge auf Gruppenebene zugeordnet. Aufgrund der entsprechenden Wartungsintervalldefinitionen im Wartungsplan werden im Zeitverlauf Wartungsaufträge (*WAuf*) als Beziehung zwischen den Typen *Maschine* und *Zeitpunkt* generiert. Mit einem Wartungsauftrag können gleichzeitig Maßnahmen aus mehreren Wartungsplänen durchgeführt werden. Durch die Zuordnungsmöglichkeit mehrerer Wartungspläne müssen bsw. die Wartungsschritte einer kleinen Inspektion, die auch bei einer großen Inspektion durchzuführen sind, nur einmal definiert werden. Abgebildet wird dies über den Beziehungstyp *WPLAuf* (*W*artungs*pl*an-Wartungs*auf*trag-Zuordnung), der aus Sicht des Entitytyps *WAuf* einen Komplexitätsgrad von (1,n) besitzt. Mit Integritätsbedingung <5> wird gewährleistet, daß nur solche Wartungspläne für einen Wartungsauftrag einer Maschine referenziert werden, die für den Maschinentyp der Maschine Gültigkeit besitzen. Mit Hilfe des Beziehungstyps *WAG* (*W*artungs*a*rbeits*g*ang) zwischen den Entitytypen *WAuf* und *WSchritt* werden konkret in der Wartungsmaßnahme durchzuführende Wartungsarbeitsgänge festgelegt. Integritätsbedingung <6> stellt sicher, daß nur solche Wartungsarbeitsgänge angelegt werden, für die in den von dem Wartungsauftrag referenzierten Wartungsplänen Wartungschritte vorhanden sind.

Analog zu den Fertigungsaufträgen werden für die Planung der Wartungsmaßnahmen die Beziehungstypen *WPlanbeleg* (*W*artungsbedingte *Planbeleg*ung) und *WPlanReserv* (*W*artungsbedingte *Planreserv*ierung) sowie für die Dokumentation der Wartungsdurchführung die Beziehungstypen *WIstbeleg* (*W*artungsbedingte *Istbeleg*ung) und *WIstVerbrauch* (*W*artungsbedingte *Istverbrauch*) mit entsprechenden Integritätsbedingungen <7> und <8> angelegt. Zu beachten ist dabei, daß die zu wartende Maschine während der gesamten Wartungsdauer nicht dem übrigen Fertigungsprozeß zur Verfügung steht. Dies kann über eine Planbelegung der entsprechenden Maschine oder über eine Zustandsdefinition für die gesamte Zeitdauer des Wartungsauftrags abgebildet werden.

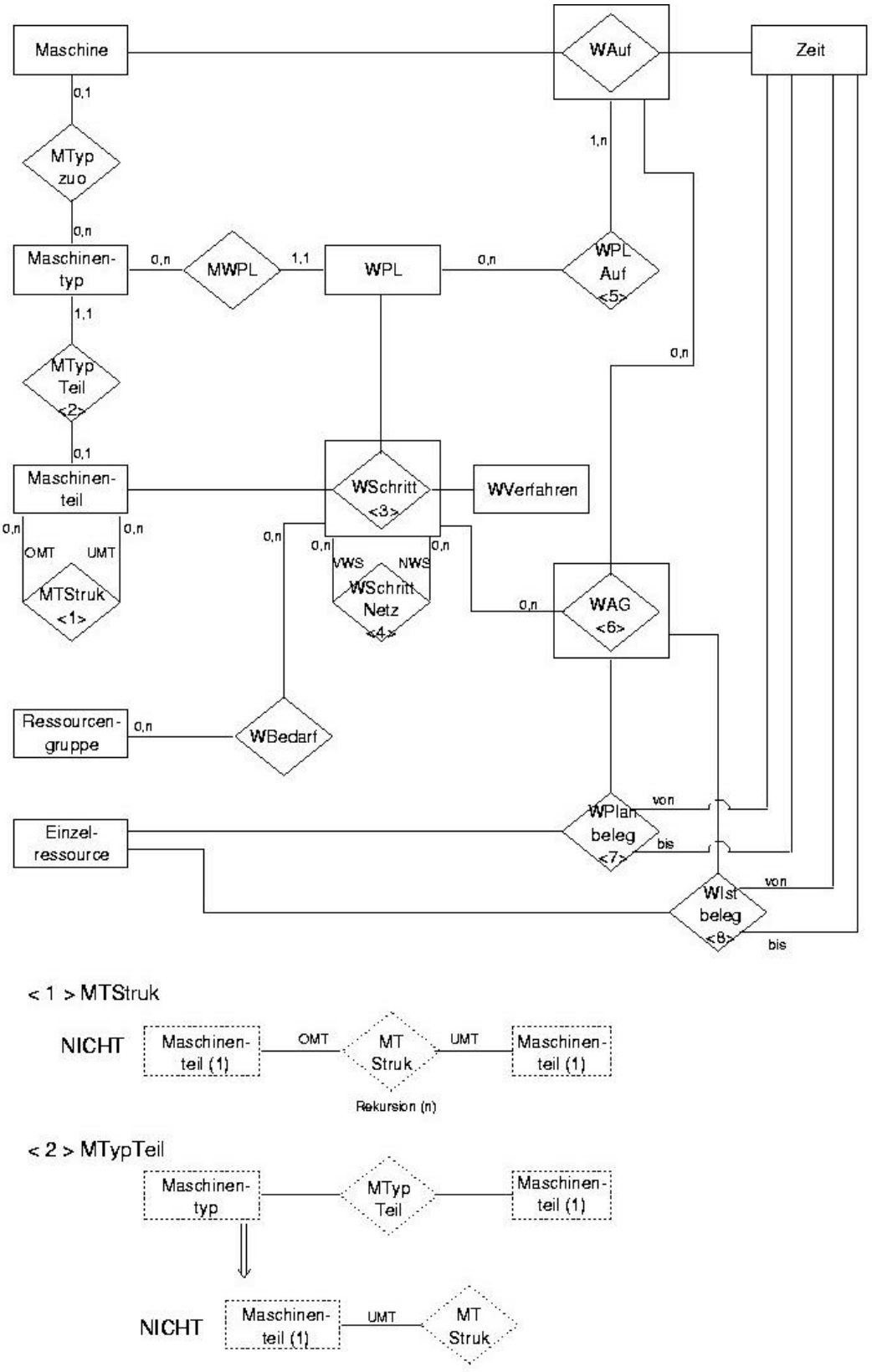


Abb. 104-1: PERM zur Instandhaltung

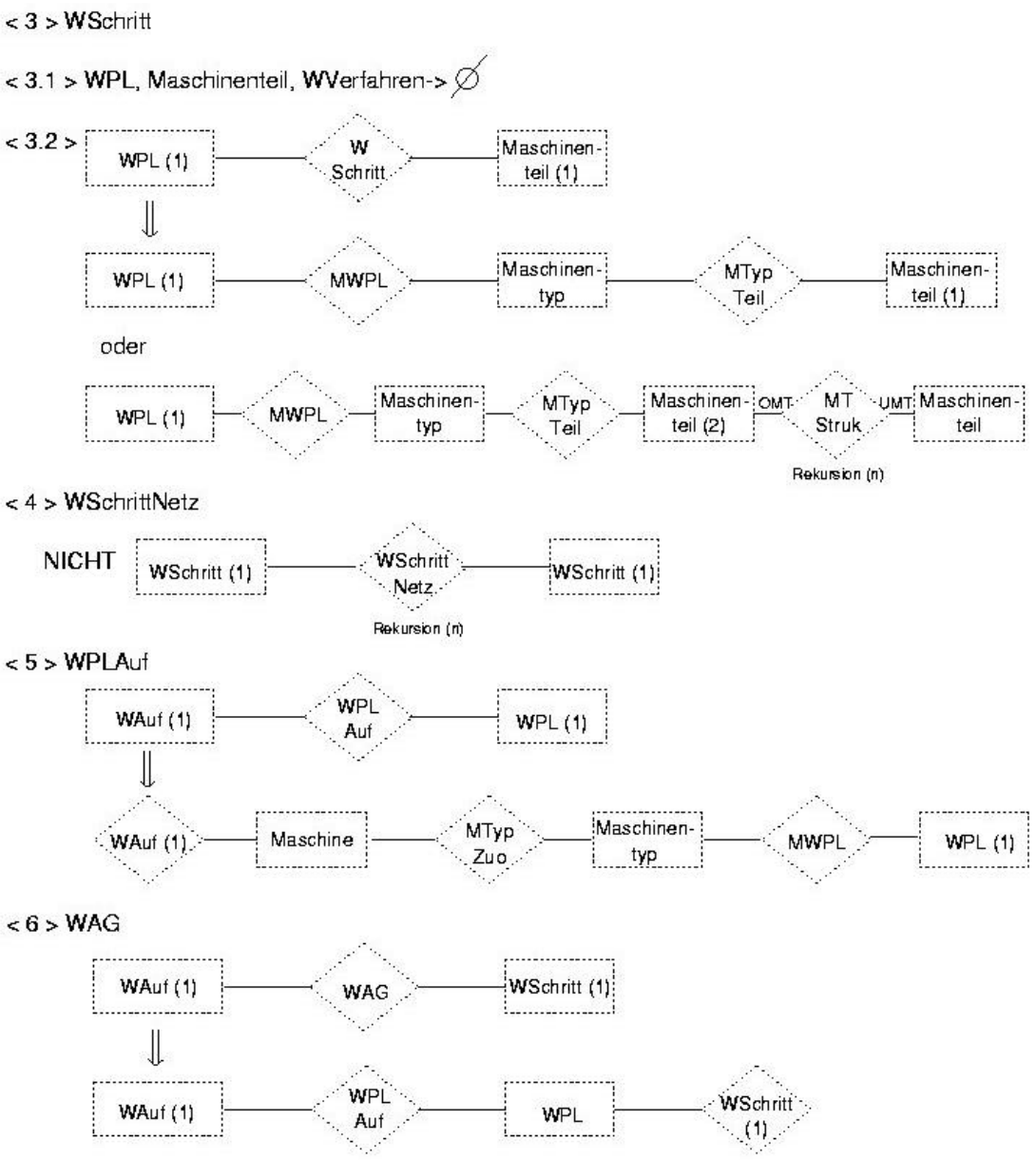


Abb. 104-2: PERM zur Instandhaltung

Durch die Analogie der *Plan*- und *Ist*-Beziehungen zwischen den Fertigungs- und Wartungsaufträgen kann man die Planbelegungs- und Istbelegungsbeziehungen zu einem Beziehungtyp alternativer Entitytypen zusammenfassen. Abbildung 105 zeigt dies am Beispiel der Beziehungen *Planbeleg* der Fertigungsaufträge und *WPlanbeleg* der Wartungsaufträge. Der resultierende Beziehungstyp *PlanWPlanbeleg* besitzt alternative Kanten zu den Entitytypen *AAG* und *WAG*. Die Darstellung in einem Beziehungstyp ist sinnvoll, da die Funktionen der Planung Beziehungen beider Typen unter gleichzeitiger Beachtung bereits vorhandener Beziehungen erzeugen.

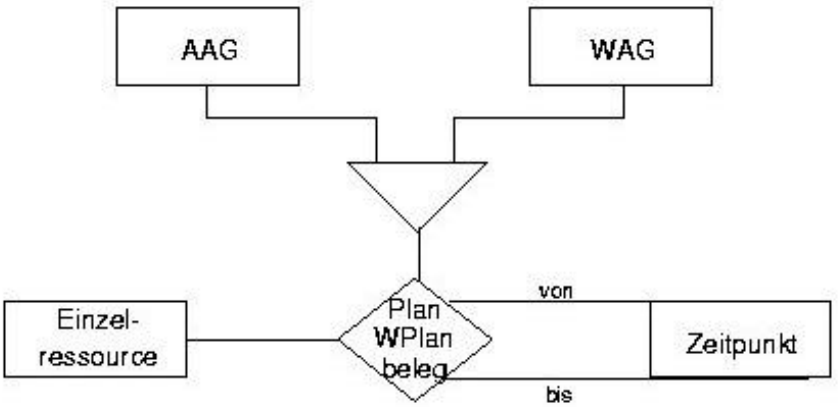


Abb. 105: Planbelegung als Beziehung alternativer Entitytypen

8. Umsetzung des Expanded Entity-Relationship-Modells in Datenbanksysteme

Bei der Implementierung von Informationssystemen müssen die konstruierten Datenstrukturen in die Beschreibungssprachen von Datenbanksystemen umgesetzt werden. Das dem Datenbanksystem zugrundeliegende Modell bestimmt die Möglichkeit, inwieweit die verwendeten Datenstrukturen abbildbar sind.

Aufgrund mangelnder semantischer Ausdruckskraft eines Modells kann es notwendig werden, Datenstrukturinformationen im Programmcode einer Applikation abzubilden (Funktionssicht der Umsetzungsebene). Dies führt dazu, daß verschiedene Programme, die die Datenbank benutzen, die gleiche Logik mehrmals abbilden müssen. Dadurch kann es leicht zu unerwünschten und inkonsistenten Datenbankzuständen kommen.

8.1 Modelle der Datenbanksysteme

Die heute verfügbaren Datenbanksysteme basieren fast ausschließlich auf dem hierarchischen Modell, dem Netzwerkmodell oder dem Relationenmodell (vgl. hierzu Date 81; Schlageter/Stucky 83; Martin 87; Zehnder 87; Vetter 89; Scheer 90f).

Im **hierarchischen Modell**, dem ersten und ältesten Modell, werden alle Datenstrukturen als geordnete Bäume dargestellt. Jedes Objekt (Element) hat maximal einen Vater und kann mehrere Söhne besitzen. Elemente ohne Vater werden Wurzelelemente genannt (Root Segments). Die Existenz eines Elements ist von der Existenz aller hierarchisch übergeordneten Elemente einschließlich des Wurzelelements abhängig. Die Suche nach einem Objekt muß immer bei einem Wurzelelement beginnen. Das hierachische Modell eignet sich gut für die Abbildung binärer Beziehungen von Typ (5) und Gruppierungen.

Das **Netzwerkmodell** stellt eine Erweiterung des hierarchischen Modells dar und geht zurück auf eine Empfehlung der Database Task Group der Conference on Data System Language (CODASYL-DBTG). Im Netzwerkmodell besteht die Möglichkeit, daß ein Objekt (Record) neben mehreren Söhnen (Member) auch mehrere Väter (Owner) besitzen kann. Die Verknüpfung zwischen einem Owner und einem Member wird genannt. Im Netzwerkmodell können damit auch binäre oder Mehrfachbeziehungen unterschiedlicher Typen abgebildet werden, indem bsw. bei einer binären Beziehung von Typ (8) die beiden Records der Entitytypen Owner zu dem Record des Beziehungstyps sind. Die Suche nach

Datenelementen kann bei jedem Record beginnen, unabhängig davon, ob der Record keinen, einen oder mehrere Owner besitzt.

Die auf diesen Modellen aufbauenden Datenbanksysteme sind zwar in der Praxis noch häufig im Einsatz (z. B. das hierarchische System von IBM oder die Netzwerkdatenbanksysteme UDS von Siemens und von Cullinet), verlieren aber wegen ihrer Unflexibilität zunehmend an Bedeutung und werden seitens der Anbieter in der Regel nicht mehr als strategische Produkte angesehen. Dagegen setzen sich Datenbanksysteme nach dem **Relationenmodell** immer mehr durch.

8.2 Relationenmodell

In dem bereits 1970 von Codd formulierten Relationenmodell (auch relationales Datenmodell genannt) werden Informationen mit Hilfe zweidimensionaler Tabellen oder Relationen dargestellt (Codd 70). Eine Relation ist als kartesisches Produkt der Wertebereiche der sie beschreibenden Attribute definiert. Es gilt:

$$R(a_1, a_2, \ldots a_n) \subseteq dom(a_1) \times dom(a_2) \times \ldots dom(a_n)$$

Die Attribute einer Relation bilden die Spaltendefinitionen (columns). Ein einzelner Datensatz wird als Tupel bezeichnet. Da eine Relation als die Menge des kartesischen Produktes definiert ist, gibt es zum einen keine definierte Reihenfolge der Tupel, zum anderen müssen sich die Tupel einer voneinander unterscheiden.

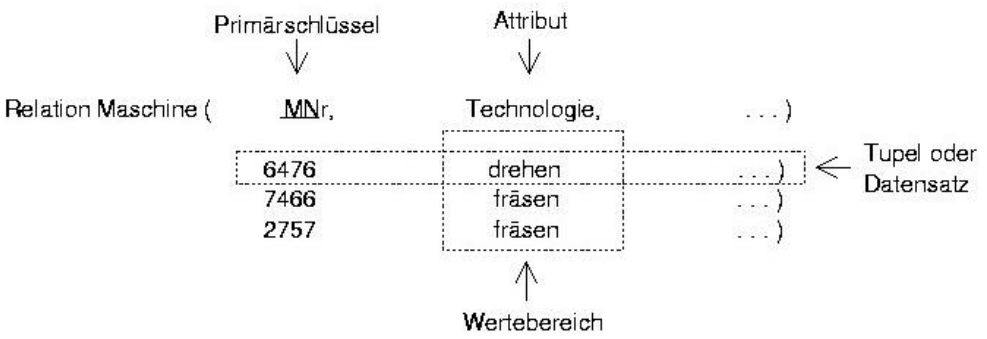


Abb. 106: Aufbau einer Relation

Diese Eindeutigkeit wird üblicherweise bereits von einer Untermenge der Attribute bzw. von einem einzige Attribut erfüllt und als Primärschlüssel bezeichnet (in Abbildung 106 unterstrichen dargestellt).

Als Datenbankbeschreibungssprache(Data Description Language, DDL) und Datenmanipulationssprache(Data Manipulation Language, DML) für relationale Datenbanksysteme hat sich die Sprache SQL durchgesetzt, die bereits als ISO-Norm (ISO= International Organization for Standardization) vorliegt (ISO 89). Für die weitere Betrachtung sind folgende Elemente der Sprache SQL wichtig:

- Die **CREATE TABLE**-Anweisung dient zur Definition einer Relation.

- Über die **NOT NULL**-Anweisung können undefinierte Werte für ein Attribut ausgeschlossen werden (Wedekind 88). Der undefinierte Wert NULL ist nicht gleichbedeutend mit dem numerischen Wert Null oder einem leeren String.

- Die **UNIQUE**-Anweisung bestimmt für ein Attribut oder eine Attributkombination, daß deren Ausprägung in der Relation einmalig ist. Gleichzeitig impliziert die UNIQUE-Bedingung, daß alle beteiligten Attribute keine NULL-Werte annehmen dürfen. Innerhalb einer Relation können mehrere UNIQUE-Bedingungen formuliert werden.

- Die **PRIMARY KEY**-Bedingung definiert das Attribut bzw. die Attributkombination des Primärschlüssels. An den Primäschlüssel werden die gleichen Anforderungen gestellt wie an die UNIQUE-Bedingung. Im Gegensatz zu dieser ist allerdings nur ein Primärschlüssel pro Relation erlaubt.

- Mit Hilfe der **FOREIGN KEY**-Bedingung kann der Wert eines Attributs oder einer Attributkombination in einer Relation von der Existenz des gleichen Wertes innerhalb eines Attributs oder einer Attributkombination einer anderen Relation abhängig gemacht werden (Fremdschlüssel). Dabei müssen der Datentyp und die Domäne in der referenzierenden und in der referenzierten Relation identisch sein. Desweiteren müssen die betroffenen Attribute der referenzierten Relation als UNIQUE oder PRIMARY KEY definiert sein.

- Die **CHECK**-Bedingung definiert Integritätsregeln für ein oder mehrere Attribute einer Relation. Die Regeln können Vergleichs-, Grenz-, Aufzählungs-, Ähnlichkeits-

oder Existenzbedingungen enthalten. Es können innerhalb der Relation nur Attribute eines Tupels einbezogen werden.

- Mit Hilfe der **CREATE VIEW**-Anweisungkönnen virtuelle Relationen definiert werden, deren Tupel aus anderen echten oder virtuellen Relationen über Operationen der Relationenalgebra wie Projektion, Selektion, Join, Vereiningung, Durchschnitt oder Differenz(vgl. hierzu Schlageter/Stucky 83) abgeleitet werden.

Obwohl das Relationenmodell wie in Abbildung 106 beschrieben ein allgemeines Modell für Datenbanksysteme darstellt, die Sprache SQL dagegen die konkrete Umsetzung in eine Data Description Language widerspiegelt, soll im folgenden aufgrund der engen Verknüpfung und der Standardisierung nicht dazwischen unterschieden werden (vgl. auch Scheer 90f, S. 24).

8.3 Abbildung im Relationenmodell

Die Abbildungsmöglichkeiten von Datenstrukturen im Relationenmodell sollen anhand der für das Expanded Entity-Relationship-Modell definierten Konstruktionsoperatoren erfolgen (zur Überführung von ERM-Strukturen in das Relationenmodell bzw. in relationale Datenbanksysteme vgl. auch Wong/Katz 80; Teorey et al. 86; Markowitz/Shoshani 89; Batra et al. 90; Loos 90a; Scheer 90f).

Auf die Zerlegung von Datenstrukturen im Rahmen der Normaliesierung, wie sie für das Relationenmodell entwickelt wurde (Codd 70; Date 81; Kent 83), soll bei der Überführung nicht weiter eingegangen werden. Werden bei der Modellierung im Expanded Entity-Relationship-Modell nur einwertige bzw. skalare Attribute zugelassen, so befindet sich jedes Objekt zumindest in der ersten Normalform (1NF). Aufgrund der Attributierung der Objekttypen kann es aber notwendig sein, für die Beseitigung von Teilschlüsselabhängigkeiten (tweite Normalform, 2NF) oder transitiven Abhängigkeitenen (dritte Normalform, 3NF) Objekttypen zu zerlegen.

Entitytypen und Beziehungstypen

Ein Entitytyp kann direkt als Relation abgebildete werden. Die beschreibenden Attribute des Entitytyps werden durch die Columns der Relation ausgedrückt. Die Determinante des Entitytyps wird zum Primärschlüssel der Relation. In der Regel wird aus jedem Entitytyp

eine eigene Relation gebildet. In besonderen Ausnahmefällen können mehrere Entitytypen aufgrund der Art der sie verbindenden Beziehungstypen zusammen eine Relation bilden.

Die Abbildung von Beziehungstypen erfolgt ebenfalls in Relationen, wobei in Abhängigkeit von der Art des Beziehungstyps eigene Relationen gebildet werden oder der Beziehungstyp mit der Relation eines Entitytyps zusammengefaßt wird. Daraus folgt, daß eine Relation sowohl die Eigenschaft eines Entitytyps als auch die Eigenschaft eines Beziehungstyps besitzen kann. Bei der Überführung müssen folgende Regeln beachtet werden:

- Die Relation erhält als Primärschlüssel das Attribut bzw. die Attributkombination, die der Determinante entspricht.

- Hat ein Beziehungstyp aufgrund der funktionalen Abhängigkeiten mehrere mögliche Determinanten, so wird eine Determinante zum Primärschlüssel. Die Attribute bzw. Attributkombinationen der anderen Determinanten werden als UNIQUE gekennzeichnet.

- Alle Attribute der funktionalen Abhängigkeiten sind Fremdschlüssel und müssen über Referenzbedingungen auf die Primärschlüssel der Relationen der Entitytypen verweisen.

- Verschieden Typen können in einer Relation zusammengefaßt werden, wenn sie über gleiche Determinanten und damit über die gleichen Primärschlüssel verfügen.

- Werden in einer Relation verschiedene Typen zusammengefaßt, z. B. ein Entitytyp und ein Beziehungstyp, so werden die Attribute aller Typen in die Relation übernommen.

- Wird ein Beziehungstyp in eine eigene Relation überführt, so muß jedes Tupel über die Fremdschlüsselbeziehungen alle aggregierten Entitytypen referenzieren. Dies wird durch die NOT NULL-Implizierung der Primärschlüssel- oder UNIQUE-Bedingungen der Fremdschlüssel innerhalb der Relation des Beziehungstyps sichergestellt.

- Wird ein Entity in eine eigene Relation überführt und hat der Entitytyp Beziehungen mit Kanten, deren Komplexitätsgrade einen (min)-Wert von größer null besitzen, so hat der Primärschlüssel der Relation gleichzeitig Fremdschlüsseleigenschaft. Da ein Fremdschlüssel allerdings nur Primärschlüssel oder UNIQUE-Attribute bzw.

-Attributkombinationen referenzieren kann, hängt die Abbildungsmöglichkeit von der referenzierten Relation ab.

- Ist eine Relation aus einer Zusammenfassung eines Entitytyps und eines Beziehungstyp entstanden, und ist die Beziehung des Entitytyps optional, d. h. der (min)-Wert des Komplexitätsgrades der Kante ist null, so kann der Fremdschlüssel, der die Relation des anderen an der Beziehung beteiligten Entitytyps referenziert, den NULL-Wert annehmen.

Aus den genannten Regeln ergibt sich, daß bei **binären Beziehungen** Beziehungstypen dann in eigene Relationen überführt werden, wenn die Komplexitätsgrade beider Kanten einen (max)-Wert größer als 1 besitzen. In den anderen Fällen kann der Beziehungstyp in der Relation eines Entitytyps abgebildet werden. Im folgenden werden die verschiedenen Arten der Beziehungen, wie sie im Kapitel "Binäre Aggregation" (s. S. 46) differenziert wurden, als Relationen dargestellt (vgl. auch Abbildung 28 und 29). Wird ein Entity- oder Beziehungstyp nicht als Relation dargestellt, so wird angegeben, in welcher anderen Relation die beschreibenden Attribute zu finden sind. Bei der Definition wird sowohl auf den CREATE-Befehl als auch auf die Datentypangabe verzichtet. Nicht-abbildbare Konstrukte werden durch Ausrufezeichen gekennzeichnet:

```
(1)   TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE B( b, ...,
      PRIMARY KEY (b))

      TABLE AB( a, b UNIQUE, ...,
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b))

(2)   TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE B( b, a UNIQUE, ..., AB-Attribute...,
      PRIMARY KEY (b), FOREIGN KEY (a) REFERENCES A(a))

(3)   TABLE A( a, b UNIQUE, ..., B-Attribute..., AB-Attribute...,
      PRIMARY KEY (a))

(4)   TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE B( b, a, ..., AB-Attribute...,
      PRIMARY KEY (b), FOREIGN KEY (a) REFERENCES A(a))
```

```
(5)   TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE B( b, a NOT NULL, ..., AB-Attribute...,
      PRIMARY KEY (b), FOREIGN KEY (a) REFERENCES A(a))
```

```
(6)   TABLE A( a, ...,                                  Þ nicht abbildbar!
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES B(a)) !!!

      TABLE B( b, a, ..., AB-Attribute...,
      PRIMARY KEY (b), FOREIGN KEY (a) REFERENCES A(a))
```

```
(7)   TABLE A( a, ...,                                  Þ nicht abbildbar!
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES B(a)) !!!

      TABLE B( b, a NOT NULL, ..., AB-Attribute...,
      PRIMARY KEY (b), FOREIGN KEY (a) REFERENCES A(a))
```

```
(8)   TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE B( b, ...,
      PRIMARY KEY (b))

      TABLE AB( a, b, ...,
      PRIMARY KEY (a,b), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b))
```

```
(9)   TABLE A( a, ...,                                  Þ nicht abbildbar!
      PRIMARY KEY (a))

      TABLE B( b, ...,
      PRIMARY KEY (b), FOREIGN KEY (b) REFERENCES AB(b)) !!!

      TABLE AB( a, b, ...,
      PRIMARY KEY (a,b), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b))
```

```
(10)  TABLE A( a, ...,                                  Þ nicht abbildbar!
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES AB(a)) !!!

      TABLE B( b, ...,
      PRIMARY KEY (b), FOREIGN KEY (b) REFERENCES AB(b)) !!!

      TABLE AB( a, b, ...,
      PRIMARY KEY (a,b), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b))
```

```
(11)  TABLE A( a, ...,
      PRIMARY KEY (a1))

      TABLE AB( a1, a2 UNIQUE, ...,
      PRIMARY KEY (a1), FOREIGN KEY (a1) REFERENCES A(a),
      FOREIGN KEY (a2) REFERENCES A(a))
```

```
(12)  TABLE A( a1, a2 UNIQUE, ..., AB-Attribute...,
      PRIMARY KEY (a1), FOREIGN KEY (a2) REFERENCES A(a1))
```

```
(13)  TABLE A( a1, a2, ..., AB-Attribute...,
      PRIMARY KEY (a1), FOREIGN KEY (a2) REFERENCES A(a1))

(14)  TABLE A( a, ...
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES AB(a1))

      TABLE AB( a1, a2, ...,
      PRIMARY KEY (a1), FOREIGN KEY (a1) REFERENCES A(a),
      FOREIGN KEY (a2) REFERENCES A(a))

(15)  TABLE A( a, ...,
      PRIMARY KEY (a1))

      TABLE AB( a1, a2, ...,
      PRIMARY KEY (a1,a2), FOREIGN KEY (a1) REFERENCES A(a),
      FOREIGN KEY (a2) REFERENCES A(a))

(16)  TABLE A( a, ...,                              Þ nicht abbildbar!
      PRIMARY KEY (a1), FOREIGN KEY (a) REFERENCES AB(a1)) !!!

      TABLE AB( a1, a2, ...,
      PRIMARY KEY (a1,a2), FOREIGN KEY (a1) REFERENCES A(a),
      FOREIGN KEY (a2) REFERENCES A(a))

(17)  TABLE A( a, ...,                              Þ nicht abbildbar!
      PRIMARY KEY (a1), FOREIGN KEY (a) REFERENCES AB(a1), !!!
      FOREIGN KEY (a) REFERENCES AB(a2)) !!!

      TABLE AB( a1, a2, ...,
      PRIMARY KEY (a1,a2), FOREIGN KEY (a1) REFERENCES A(a),
      FOREIGN KEY (a2) REFERENCES A(a))
```

Bei rekursiven, binären Beziehungen muß zum Teil die Fremdschlüsselbedingung auf die eigene Relation verweisen, da sowohl der Beziehungstyp als auch der Entitytyp in einer Relation abgebildet werden (Typen 12 und 13).

Die Typen (6) und (7) sind nicht abbildbar, da die (min)-Werte von eins der Kanten Fremdschlüsselbeziehungen von den Relationen der Entitytypen in die Relationen der zusammengefaßten Entity- und Beziehungstypen notwendig machen. Diese Fremdschlüsselbeziehungen sind nicht möglich, da das jeweilige Attribut a in den Relationen B mehrfach vorkommen kann. Die Typen (9), (10), (16) und (17) sind ebenfalls nicht abbildbar, da die Attribute b bzw. a1 in den Relationen AB nicht einzeln, sondern nur in Kombination mit dem jeweiligen Attribut a bzw. a2 als zusammengesetzter Primärschlüssel eindeutig sind.

Bei der binären Beziehung des Typs (3) können beide Entitytypen und der Beziehungstyp in einer Relation abgebildet werden. Dies läßt sich verdeutlichen, wenn alle Typen einzeln als Relationen dargestellt werden:

```
-     TABLE A( a, ...,
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES AB(a))

-     TABLE B( b, ...,
      PRIMARY KEY (b), FOREIGN KEY (b) REFERENCES AB(b))

-     TABLE AB( a, b, ...,
      PRIMARY KEY (a), UNIQUE (b),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b))
```

Relation A ist über die FOREIGN KEY-Angabe existentiell an das Vorhandensein eines Tupel in der Relation AB gebunden. Analoges gilt für Relation B. Relation AB besitzt als Primärschlüssel das Attribut a, durch die UNIQUE-Klausel des Attributs b wäre dieses allerdings ebenso geeignet. Beide Attribute haben, da sie die funktionalen Abhängigkeiten des Beziehungstyps repräsentieren, Fremdschlüsseleigenschaft. Die NOT NULL-Bedingung wird durch die Primärschlüssel- und UNIQUE-Eigenschaft impliziert. Aufgrund der Abhängigkeit kann aber in jeder Relationen nur dann ein Tupel existieren, wenn jeweils genau ein entsprechendes Tupel in den beiden anderen Relationen vorhanden ist. Dies führt zu genau einer Relation, wie sie in (3) dargestellt ist.

Die sich aufgrund der obengenannten Regeln ergebende Überführung von **Mehrfachbeziehungen** soll anhand der unterschiedlichen Typen aus Abbildung 33 vorgenommen werden. Dabei werden die Entitytypen A, B und C jeweils in eigenen Relationen abgebildet:

```
-     TABLE A( a, ...,
      PRIMARY KEY (a)

-     TABLE B( b, ...,
      PRIMARY KEY (b)

-     TABLE C( c, ...,
      PRIMARY KEY (c)
```

Der Beziehungstyp ABC wird in Abhängigkeit vom jeweiligen Typ in folgenden Relationen abgebildet:

```
(1)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a,b,c), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

(2)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a,b), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))
```

```
(3)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a,b), UNIQUE (a,c)
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

(4)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a,b), UNIQUE (a,c), UNIQUE (b,c),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

(5)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

(6)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a), UNIQUE (b), FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

(7)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a), UNIQUE (b), UNIQUE (c),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

(8)   TABLE ABC( a, b, c, ...,
      PRIMARY KEY (a), UNIQUE (b,c),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))
```

Dreifachbeziehungen, bei denen der Komplexitätsgrad von Kanten eine Untergrenze von eins besitzt, können nur dann in das Relationenmodell überführt werden, wenn auch der Wert der Obergrenze gleich eins ist. Hier greift die gleiche Restriktion wie bei den binären Beziehungen der Typen (9), (10), (16) und (17), bei denen eine Fremdschlüsselbeziehung aus der Relation des Entitytyps in die Relation des Beziehungstyps möglich sein muß. Dies ist nur dann abbildbar, wenn der Primärschlüssel der Relation des Entitytyps auch Determinante des Beziehungstyps ist. Bei gleichen Determinanten können aber der Entitytyp und der Beziehungstyp in eine Relation überführt werden. Bsw. ergeben sich für den abgewandelten Typ (5) der Dreierbeziehungen, bei dem die Kante zwischen A und ABC einen Komplexitätsgrad von (1,1) aufweist, folgende Relationen:

```
-     TABLE A( a, b, c, ..., ABC-Attribute...,
      PRIMARY KEY (a), FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

-     TABLE B( b, ...,
```

```
      PRIMARY KEY (b))

-     TABLE C( c, ...,
      PRIMARY KEY (c))
```

Überführungen von Entitytypen, die aus einer **Uminterpretation eines Beziehungstyps** entstanden sind, können zu eigenen Relationen führen. Abbildung 107 zeigt eine entsprechende Konstruktion.

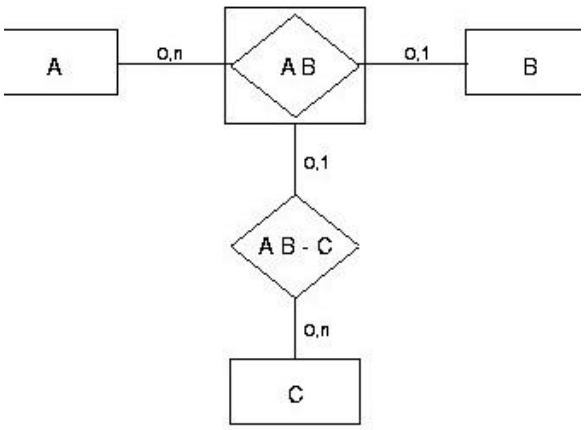


Abb. 107: Beispiel eines zum Entitytyp uminterpretierten Beziehungstyps

Bei einer Überführung anhand der oben aufgestellten Regeln für Entity- und Beziehungstypen ergeben sich folgende Relationen:

```
-     TABLE A( a, ...,
      PRIMARY KEY (a))

-     TABLE B( b, a, AB-Attribute..., c,  AB-C-Attribute...,
      PRIMARY KEY (b),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (c) REFERENCES C(c)) !!!      Þ nicht abbildbar!

-     TABLE C( c, ...,
      PRIMARY KEY (c))
```

Die Relation B stellt zum einen den Entitytyp B sowie den Beziehungstyp AB dar. Zum anderen wird aufgrund des Komplexitätsgrades von (0,1) der Kante zwischen den Typen AB und AB-C der Beziehungstyp AB-C in die Relation B übernommen. Daraus ergeben sich die Fremdschlüsselbeziehungen zu den Relationen A und C. Eine Beziehung AB-C kann aber nur existieren, wenn eine Beziehung AB existiert. Übertragen auf die Relationen bedeutet dies, daß das Attribut c der Relation B nur dann einen gültigen Wert ungleich NULL

annehmen kann, wenn der Wert des Attributs a ebenfalls ungleich dem NULL-Wert ist. Da diese Abhängigkeit im Relationenmodell nicht abgebildet werden kann, muß der Beziehungstyp AB explizit als Relation ausformuliert werden:

```
-     TABLE A( a, ...,
      PRIMARY KEY (a))

-     TABLE B( b, ...,
      PRIMARY KEY (b))

-     TABLE AB( b, a, c, ..., AB-C-Attribute...,
      PRIMARY KEY (b),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (b) REFERENCES B(b),
      FOREIGN KEY (c) REFERENCES C(c))

-     TABLE C( c, ...,
      PRIMARY KEY (c))
```

Komplexitätsgrade, die eine andere Untergrenze als null oder eins bzw. eine andere Obergrenze als eins oder "n" besitzen (d. h. **ungleich der 1,c,m-Notation**), können nur bedingt abgebildet werden. Abbildung 108 zeigt einen binären Beziehungstyp, dessen Kante einen Komplexitätsgrad von (2,4) aufweist.

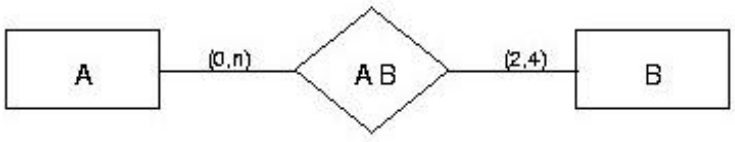


Abb. 108: Binäre Beziehung mit einer Komplexität von (2,4)

Eine möglich Überführung in Relationen könnte wie folgt aussehen:

```
-     TABLE A( a, ...,
      PRIMARY KEY (a))

-     TABLE B( b, a1 NOT NULL, a2 NOT NULL, a3, a4,  AB-Attribute1,
      AB-Attribute2, AB-Attribute3, AB-Attribute4...,
      PRIMARY KEY (b), FOREIGN KEY (a1) REFERENCES A(a),
      FOREIGN KEY (a2) REFERENCES A(a),
      FOREIGN KEY (a3) REFERENCES A(a),
      FOREIGN KEY (a4) REFERENCES A(a),
      CHECK (a1 <> a2  AND  a1 <> a3  AND  a1 <> a4),
      CHECK (a2 <> a3  AND  a2 <> a4),
      CHECK (a3 <> a4))
```

Die Attribute a1 bis a4 der Relation B sind Fremdschlüssel mit einer Referenz auf den Primärschlüssel der Relation A. Die ersten beiden Attribute (a1 und a2) müssen einen gültigen Wert besitzen (Untergrenze = 2). Insgesamt können vier Referenzen gesetzt werden (Obergrenze = 4). Gleichzeitig muß sichergestellt werden, daß ein referenziertes Tupel der Relation A nur einmal referenziert wird. Dies wird über die CHECK-Bedingungen erzwungen. Dieser Lösungsvorschlag ist nur bedingt einsetzbar, da das Fremdschlüsselattribut in Abhängigkeit von dem (max)-Wert des Komplexitätsgrades wiederholt werden muß.

Allgemein läßt sich sagen, daß die **Darstellungsmächtigkeit der 1,c,m-Notation eine Obermenge zu der des Relationenmodells** bezüglich der Abbildung von Beziehungen darstellt. Wenn allerdings Referenzen auf nichteindeutige Attribute oder Attributkombinationen zugelassen wären, könnten alle durch die 1,c,m-Notation formulierbaren Typen von Beziehungen im Relationenmodell dargestellte werden.

Gruppierung

Gruppierungen können, wie bereits in Kapitel "Gruppierung" (s. S. 60) gezeigt, als Aggregation mit bestimmten Komplexitätsgraden gebildet werden. Abbildung 109 zeigt die äquivalente Darstellung einer Gruppierung und einer binären Aggregation vom Typ (5) entsprechend der Definition in dem Kapitel "Binäre Aggregation" (s. S. 46). Daraus ergibt sich die Überführungsmöglichkeit in die Relationen:

```
-     TABLE Kostenstelle( KoStNr, ...,
      PRIMARY KEY (KoStNr))

-     TABLE Maschine( MaschNr, KoStNr NOT NULL, ...,
      PRIMARY KEY (MaschNr),
      FOREIGN KEY (KoStNr) REFERENCES Kostenstelle(KoStNr))
```

Häufig sind aber gerade bei Gruppierungen in der Praxis die Schlüssel hierarchisch aufgebaut. Abbildung 110 zeigt ein entsprechendes Beispiel, bei dem der Schlüssel der Maschine aus der Kostenstellennummer und einem zweistelligen Zähler besteht. Die Übertragung einer Gruppierung mit diesem Schlüsselaufbau in das Relationenmodell ist ebenso möglich:

```
-     TABLE Maschine( KoStNr, LaufendeMasch, ...,
      PRIMARY KEY (KoStNr, LaufendeMasch),
      FOREIGN KEY (KoStNr) REFERENCES Kostenstelle(KoStNr))
```

In diesem Fall wird ein Entitytyp mit einer Schlüsselattributkombination dargestellt, was sonst nur bei Beziehungstypen üblich ist.

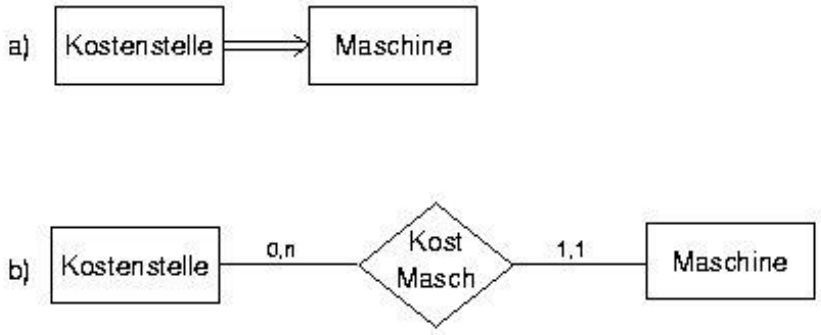


Abb. 109: Beispiel einer Gruppierung

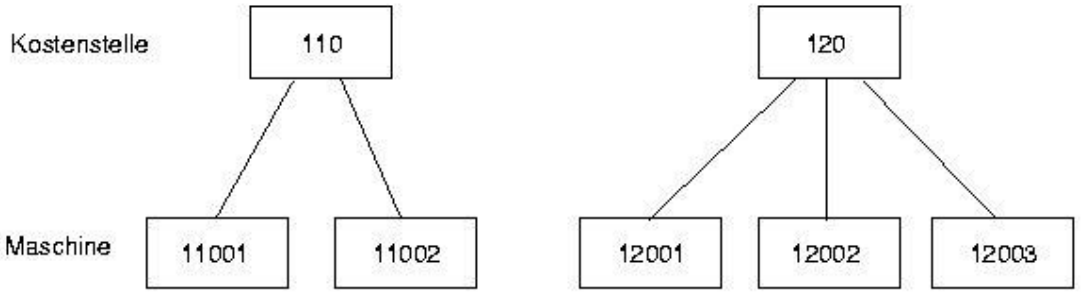


Abb. 110: Hierarchischer Schlüsselaufbau einer Gruppierung

Generalisierung

Bei der Generalisierung können sowohl der generische Entitytyp als auch die Entitytypen der Subtypen in Relationen überführt werden. Der eigentliche Generalisierungsoperator, der in den Diagrammen des Expanded ERM als Dreieck dargestellt wird, wird selbst nicht durch eine Relation ausgedrückt. Abbildung 111 zeigt das Beispiel einer Generalisierung der Entitytypen *NC-Maschine* und *Engpaßaggregat* zu dem Entitytyp *Maschine*.

Die daraus resultierenden Relationen sind:

```
-     TABLE Maschine( MNr, ...,
      PRIMARY KEY (MNr))

-     TABLE NC-Maschine( MNr, ...,
      PRIMARY KEY (MNr),
      FOREIGN KEY (MNr) REFERENCES Maschine(MNr))

-     TABLE Engpaßaggregat( MNr, ...,
      PRIMARY KEY (MNr),
      FOREIGN KEY (MNr) REFERENCES Maschine(MNr))
```

Alle Relationen besitzen den gleichen Primärschlüssel *MNr*. Zusätzlich wird für die Primärschlüssel der Relationen der Subtypen eine Referenzbedingung zu dem Primärschlüssel der Relationen des generischen Typs aufgebaut. Dadurch wird sichergestellt, daß jedes Tupel der Subtypen auch ein entsprechendes Tupel in der generischen Struktur besitzt. Diese Beziehung wird auch als semireferentielle Integrität bezeichnet (s. auch Steinbauer/Wedekind 85).

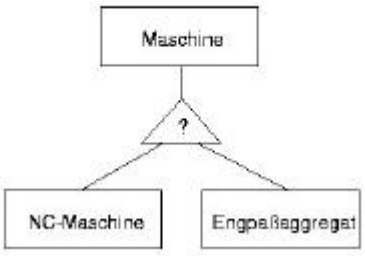


Abb. 11: Beispiele einer Generalisierung

Durch die genannten Referenzbedingungen kann weder sichergestellt werden, daß für jedes Tupel des generischen Typs auch ein Tupel in einem Subtyp besteht, noch daß ein Tupel des generischen Typs nicht in mehreren Subtypen enthalten ist. Daraus folgt, daß die gezeigte Überführung eine nicht-vollständige, nicht-disjunkte Generalisierung (Typ (0,n)) repräsentiert.

Die vollständige Generalisierung würde eine Referenzbeziehung von dem generischen Typ in die Subtypen erfordern, wobei die Referenzen zu den einzelnen Subtypen allerdings mit Exklusiv-Oder untereinander verbunden wären. Eine solche Formulierung ist im Relationenmodell nicht möglich.

Die Disjunktionen der Generalisierung würden eine "negative" Referenz von jedem Subtyp zu allen anderen Subtypen erfordern, die ausschließt, daß ein Tupel dieses Subtyps in einem anderen Subtyp existiert. Auch diese Abbildung ist im Relationenmodell nicht möglich.

Für die Generalisierung ergeben sich damit folgende Abbildungsmöglichkeiten im Relationenmodell:

- `Generalisierungstyp (1,1)` þ **`nicht abbildbar!`**
- `Generalisierungstyp (0,1)` þ **`nicht abbildbar!`**
- `Generalisierungstyp (1,n)` þ **`nicht abbildbar!`**
- `Generalisierungstyp (0,n)` þ **`abbildbar!`**

Die nicht direkt abbildbaren Generalisierungstypen können allerdings durch Hilfskonstrukte im Relationenmodell unterstützt werden:

- Bei disjunkten Generalisierungen empfiehlt es sich, in die Relationen des generischen Typs ein Attribut *Subtyp* aufzunehmen, das den Namen des Subtyps des Tupels enthält:

  ```
  TABLE Maschine( MNr, Subtyp, ...,
  PRIMARY KEY (MNr),
  CHECK Subtyp IN ('NC-Maschine','Engpaßaggregat'))
  ```

  Dadurch kann, wenn aufgrund des fehlenden Integritätszwangs unbeabsichtigt mehrere Tupel in Subtypen vorhanden sind, der richtige Subtyp bestimmt werden. Über eine CHECK-Bedingung wird sichergestellt, daß das Attribut *Subtyp* als Aufzählungstyp nur die Werte der Subtypnamen enthalten kann.

- Bei disjunkten, vollständigen Generalisierungen kann zusätzlich das Attribut *Subtyp* als NOT NULL definiert werden :

  ```
  TABLE Maschine( MNr, Subtyp NOT NULL, ...,
  PRIMARY KEY (MNr),
  CHECK Subtyp IN ('NC-Maschine','Engpaßaggregat'))
  ```

- Besitzt der generische Typ keine eigenen Attribute außer dem Schlüsselattribut und wird das Schlüsselattribut von keinen anderen Relationen als den Subtypen für Fremdschlüsselbeziehungen benötigt (d. h. wenn der generische Typ keine Kanten zu

Beziehungstypen besitzt), so muß der generische Typ nicht als eigene Relation abgebildet werden. Die komplette Menge des generischen Typs, d. h. die Ausprägungsmenge des Schlüsselattributs des Subtyps, kann als Outer Join ermittelt werden (vgl. hierzu Codd 79), der als VIEW mit dem Namen des generischen Typs hinterlegt werden kann.

- Die Abbildung einer nicht-disjunkten, vollständigen Generalisierung kann durch die Aufnahme von Attributen unterstützt werden, die anzeigen, ob das Tupel des generischen Typs auch jeweils ein Tupel in dem Subtyp besitzt. Die Attribute erhalten die Namen der Subtypen und werden jeweils über eine CHECK-Bedingung als Bool'sche Variablen mit dem numerischen Wert null für "nicht vorhanden" und eins für "vorhanden" definiert. Eine zusätzliche Checkbedingung stellt sicher, daß die Summe aller so definierten Attribute mindestens eins beträgt, womit die Vollständigkeit ausgedrückt wird:

```
TABLE Maschine( MNr, NC-Maschine NOT NULL,
                  Engpaßaggregat NOT NULL, ...,
PRIMARY KEY (MNr),
CHECK NC-Maschine  IN (0,1),
CHECK Engpaßaggregat IN (0,1),
CHECK (NC-Maschine + Engpaßaggregat >= 1 ))
```

Beziehungstypen mit alternativen Entitytypen

Die Überführung von Beziehungstypen mit alternativen Entitytypen in das Relationenmodell mit Hilfe zusätzlicher Generalisierungen (vgl. hierzu Abbildung 40 b) ist nur unzureichend möglich, da die Generalisierungen disjunkt sein müssen und deren Abbildung nicht direkt möglich ist. Deshalb soll versucht werden, entsprechend der Darstellung des Expanded ERM nur die beteiligten Entitytypen und den Beziehungstyp als Relationen darzustellen (vgl. Abbildung 63). Dabei müssen die Bedingungen für die Überführung von Beziehungstypen (vgl. Kapitel "Entitytypen und Beziehungstypen", S. 174) beachtet werden. Bei der Überführung stößt man auf folgende Schwierigkeiten:

- Werden Fremdschlüsselreferenzen in die Relationen der alternativen Entitytypen benötigt, so ist dies nicht abbildbar, da FOREIGN KEY-Bedingungen eindeutig formuliert werden müssen. Dies ist immer dann der Fall, wenn der Beziehungstyp in eine eigene Relation überführt wird oder mit den einzelnen Entitytyp zusammengefaßt wird.

- Wird der Beziehungstyp dagegen mit den Relationen der alternativen Entitytypen zusammengefaßt, so müssen auch die die Beziehung beschreibenden Attribute in die Relationen übernommen werden. Dies ist aber nicht sinnvoll, weil die Motivation für die Konstruktion der Beziehung gerade die Zusammenfassung der gleichen beschreibenden Attribute aus unterschiedlichen Beziehungstypen ist.

Dies soll beispielhaft an Beziehungen der Typen (4a), (4b) und (8) gezeigt werden:

```
(4a)  TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE C( c, ..., B-Attribute..., !!!
      PRIMARY KEY (c),
      FOREIGN KEY (a) REFERENCES A(a))

      TABLE D( d, ..., B-Attribute..., !!!      Þ nicht sinnvoll!
      PRIMARY KEY (d),
      FOREIGN KEY (a) REFERENCES A(a))

(4b)  TABLE A( a, c, d, ..., B-Attribute...,          Þ nicht abbildbar!
      PRIMARY KEY (a),
      FOREIGN KEY (c) REFERENCES C(c) !!!,
      FOREIGN KEY (d) REFERENCES D(d)) !!!

      TABLE C( c, ...,
      PRIMARY KEY (c))

      TABLE D( d, ...,
      PRIMARY KEY (d))

(8)   TABLE A( a, ...,
      PRIMARY KEY (a))

      TABLE B(a, c oder d !!!, ...,                   Þ nicht abbildbar!
      PRIMARY KEY (a, c oder d),
      FOREIGN KEY (a) REFERENCES A(a),
      FOREIGN KEY (c) REFERENCES C(c),
      FOREIGN KEY (d) REFERENCES D(d))

      TABLE C( c, ...,
      PRIMARY KEY (c))

      TABLE D( d, ...,
      PRIMARY KEY (d))
```

Zusammenfassend läßt sich sagen, daß sich Beziehungstypen mit alternativen Entitytypen nicht im Relationenmodell darstellen lassen.

Versionsbehaftete Objekte

Versionsverwaltungen werden durch das Relationenmodell nicht unterstützt. Aufbauend auf der unvollständigen Modellierung eines versionsbehafteten Entitytyps in Abbildung 46 soll eine Überführung in das Relationenmodell vorgenommen werden:

Die Entitytypen *ANr* und *Zeit* dienen der Konstruktionshilfe und enthalten nur die Primärschlüssel *a* bzw. ein Attribut des Datentyps DATE mit der Bedeutung *gültig_ab*. Sie werden daher nicht in eigene Relationen übernommen. Die zum Entitytyp uminterpretierte Beziehung *A* stellt einen generischen Typ innerhalb der Generalisierung zu den Subtypen *A-lebend* und *A-Historie* dar. Da die drei Entitytypen die gleichen beschreibenden Attribute besitzen und nur zu Verdeutlichung konstruiert wurden, soll die Generalisierung in eine Relation überführt werden. Nachfolgend sind zwei Möglichkeiten der Überführung dargestellt (vgl. auch Kinzinger 83; Müller/Steinbauer 83):

- In der Relation werden für alle Entities, auch für nicht mehr lebende, die kompletten Versionsreihen gespeichert. Der Primärschlüssel muß daher aus einer Kombination aus der Entitynummer und dem Gültigkeitsbeginn gebildet werden. Das Bool'sche Attribut *aktuell* besitzt nur bei dem letzten Tupel einer noch lebenden Versionsreihe den Wert "wahr". Für die versionfreie Sicht kann eine VIEW definiert werden, die nur aus der Entitynummer und den beschreibenden Attributen der lebende Entities besteht:

```
TABLE A( a, gültig_ab, aktuell, ...,
PRIMARY KEY (a, gültig_ab))

VIEW A_Versionsfrei AS
      (SELECT a, ...
      FROM A
      WHERE aktuell = 'ja')
```

- Bei der zweiten Möglichkeit werden ebenfalls alle Entities der kompletten Versionsreihen in einer Relation gespeichert. Das Bool'sche Attribut *gültig* besitzt allerdings bei allen Tupeln einer noch lebenden Versionsreihe den Wert "wahr". Dadurch können neue Versionen eines Entities einfacher eingefügt werden. Die View zur Formulierung der versionfreien Sicht muß aus allen Tupel diejenigen herausfiltern, die in dem Attribut *gültig* den Wert "wahr" besitzen und zusätzlich pro Versionsreihe den größten Wert des Attributs *gültig_ab*:

```
TABLE A( a, gültig_ab, gültig, ...,
PRIMARY KEY (a, gültig_ab))


VIEW A_Versionsfrei AS
            (SELECT a, ...
            FROM A x1
            WHERE gültig = 'ja' AND gültig_ab =
                  (SELECT max(gültig_ab)
                  FROM A x2
                  WHERE x1.a = x2.a))
```

Bei den gewählten Darstellungen werden allerdings nicht die Integritätsbedingungen, die an eine Versionkontrolle gestellt werden, impliziert (vgl. Kapitel "Versionsbehaftete Objekte", S. 66). Diese müssen bei der Datenmanipulation explizit eingehalten werden.

Bei Relationen, die aus Beziehungstypen oder aus einer Zusammenfassung von Entity- und Beziehungstypen entstanden sind, ergibt sich die Schwierigkeit, daß sich durch die Fremdschüsselbedingung Referenzen auf Tupel in anderen Relationen ergeben können, die nicht mehr existent sind. Die FOREIGN KEY-Bedingung müßte auf das letzte Tupel einer lebenden Versionsreihe beschränkt sein.

Clusterbildung und komplexe Objekte

Bei der Clusterbildung und komplexen Objekten sollen komplizierte Datenstrukturen durch eine interne Repräsentation einschließlich der notwendigen Integritäten dargestellt und durch einfache Zugriffsmöglichkeiten zugänglich gemacht werden. Dagegen arbeitet das Relationenmodell mit flachen Tabellen, so daß komplexe Strukturen mit Hilfe des Normalisierungsprozesses zerlegt werden müssen. Man erhält dadurch für die Abbildung eines Objektes mehrere Relationen, d. h. die Information zu einem Objekt ist über mehrere Relationen verteilt. Es gibt daher **keine** direkten Abbildungsmöglichkeiten für Cluster oder komplexe Objekte im Relationenmodell.

Mit Hilfe der Views kann allerdings der Zugriff auf komplexe Strukturen wesentlich vereinfacht werden. Beispielsweise kann eine komplette Generalisierung mit allen Entitytypen für einen bestimmten Anwendungsfall als Cluster zu betrachten sein. Das Beispiel aus Abbildung 111, bei dem der Entitytyp *Maschine* aus den Typen *NC-Maschine* und *Engpaßaggregat* generalisiert und in drei Relationen überführt wurde, kann als eine View definiert werden. Die View *Maschine_alles* wird als Vereinigung (Befehl UNION innerhalb der View-Definition) von vier selektierten Mengen gebildet. Die erste Menge selektiert mit Hilfe eines Natural Join (vgl. hierzu Date 81) alle Maschinen, die sowohl NC-

Maschinen als auch Engpaßaggregate sind. Die zweite und dritte SELECT-Anweisung liefert jeweils die Maschinen, die nur in einem Subtyp spezialisiert sind. In der letzten Menge sind alle Maschinen enthalten, die nicht spezialisiert sind. Für Maschinen, die keine NC-Maschinen oder keine Engpaßaggregate sind, erhalten die entsprechenden Attribute den NULL-Wert.

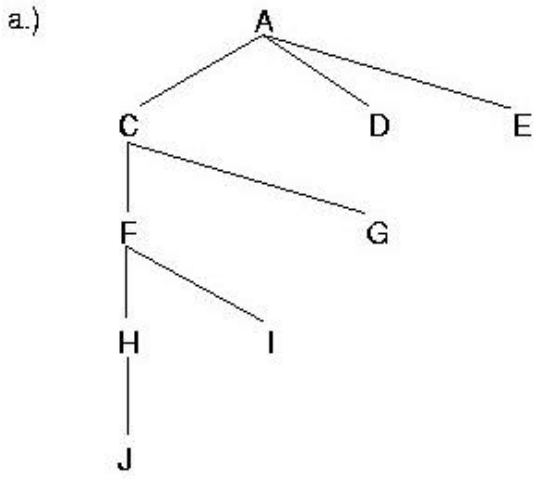
```
VIEW Maschine_alles AS (
      (SELECT Maschine.MNr, Maschine-Attribute,
       NC-Maschine-Attribute, Engpaßaggregat-Attribute
       FROM Maschine, NC-Maschine, Engpaßaggregat
       WHERE Maschine.MNr = NC-Maschine.MNr AND
             Maschine.MNr = Engpaßaggregat.MNr)
   UNION
      (SELECT Maschine.MNr, Maschine-Attribute,
       NC-Maschine-Attribute, NULL
       FROM Maschine, NC-Maschine
       WHERE Maschine.MNr = NC-Maschine.MNr AND
       Maschine.MNr NOT IN
            (SELECT MNr FROM Engpaßaggregat))
   UNION
      (SELECT Maschine.MNr, Maschine-Attribute,
       NULL, Engpaßaggregat-Attribute
       FROM Maschine, Engpaßaggregat
       WHERE Maschine.MNr = Engpaßaggregat.MNr AND
       Maschine.MNr NOT IN
            (SELECT MNr FROM NC-Maschine))
   UNION
      (SELECT Maschine.MNr, Maschine-Attribute,
       NULL, NULL
       FROM Maschine
       WHERE Maschine.MNr NOT IN
            (SELECT MNr FROM NC-Maschine) AND
            Maschine.MNr NOT IN
            (SELECT MNr FROM Engpaßaggregat)))
```

Bei dem zweiten Beispiel soll eine Teilestückliste als ein Objekt betrachtet werden. Stücklisten werden als binäre, rekursive Beziehungstypen (vgl. Kapitel "Binäre Aggregation", S. 46, Typ (15)) abgebildet und dementsprechend in zwei Relationen *Teil* und *Struktur* überführt. Eine Stücklistenauflösung stellt einen rekursiven Prozeß dar, der aus der mengenorientierten Sicht des Relationenmodells nicht nachgebildet werden kann. Deshalb muß bereits bei der Definition der Stücklistenauflösung die Rekursionstiefe angegeben werden und jede einzelne Strukturstufe gesondert formuliert werden. Das Beispiel gibt eine dreistufige Auflösung wieder, d. h. zu jedem Teil werden, soweit vorhanden, alle untergeordneten Teile, deren untergeordnete sowie von diesen noch einmal die untergeordneten Teile aufgezeigt. Die View *Stueckliste3* ist das Abbild von der Vereinigung

dreier Stukturmengen. Der erste SELECT-Befehl liefert zu jedem Teil das unmittelbar untergeordnete Teil, ferner eine Kennzeichung, daß es sich um die erste Stücklistenstufe handelt sowie ein Sortierkriterium. Der zweite und dritte SELECT-Befehl liefert die analogen Daten für die folgenden Stufen. Dabei muß für eine Stufe "n" ein rekursiver Join mit "n" Tabellen der Relation *Struktur* durchgeführt werden, und durch eine Selektion (WHERE-Bedingung) müssen die richtigen Tupel herausgefiltert werden. Anschließend werden die drei Mengen vereinigt. Das Attribut *Sortierung* wird aus Teilenummern, ausgehend von dem übergeordneten Teil bis zu dem untergeordneten Teil der Stufe "n", konkateniert. Die Ausgabe einer Stückliste erfolgt über den angegebenen SELECT-Befehl. Über die WHERE-Bedingung werden nur die zu der Stückliste eines Teils gehörenden Komponenten selektiert. Die ORDER BY-Bedingung garantiert, daß nach jedem Teil entweder ein untergeordnetes Teil, falls nicht vorhanden, ein gleichstufiges Teil oder das nächst höhere Teil ausgegeben wird (Deep-First-Suche). Durch die Stufenkennung ist ersichtlich, auf welchem Rang sich die Komponente relativ zum gewünschten Oberteil befindet:

```
-     TABLE Teil( TNr, ...,
      PRIMARY KEY (TNr))

-     TABLE Struktur( OTNr, UTNr, ...,
      PRIMARY KEY (OTNr,UTNr),
      FOREIGN KEY (OTNr) REFERENCES Teil(TNr),
      FOREIGN KEY (UTNr) REFERENCES Teil(TNr))

-     VIEW Stueckliste3( Stufe, UTNr, OTNr, Sortierung, ...) AS
            (SELECT '1..', UTNr, OTNr, UTNr, ...
             FROM Struktur
          UNION
             SELECT '.2.', x2.UTNr, x1.OTNr,
                    x2.OTNr || x2.UTNr, ...
             FROM Struktur x1, Struktur x2
             WHERE x1.UTNr = x2.OTNr
          UNION
             SELECT '..3', x3.UTNr, x1.OTNr,
                    x2.OTNr || x3.OTNr || x3.UTNr, ...
             FROM Struktur x1, Struktur x2, Struktur x3
             WHERE x1.UTNr = x2.OTNr AND x2.UTNr = x3.OTNr)

-     SELECT Stufe, UTNr, ...
      FROM Stueckliste3
      WHERE OTNr = '...'
      ORDER BY Sortierung
```

Abbildung 112 enthält das Beispiel einer Stückliste als Gozintograph (a) und das Ergebnis des SELECT-Befehls (b) für das Teil A als Liste.



b.)

Stufe	UTNr	...
1..	C	
.2.	F	
..3	H	
..3	I	
1..	D	
.2.	G	
1..	E	

Abb. 112: Stücklistenbeispiel als Gozintograph (a) und Liste (b)

Semantische Integritätsbedingungen

Für die Abbildung von semantischen Integritätsbedingungen, die über die Möglichkeiten der Mengenalgebra hinausgehen, stellt die Sprache SQL die Konstrukte CHECK und FOREIGN KEY bereit. Die Bedingung CHECK wirkt auf Attribute innerhalb eines Tupels einer Relation und gehört damit zu den intrarelationalen Integritätsbedingungen, während die FOREIGN KEY-Anweisung als interrelationale Bedingung die Beziehung der Relationen untereinander beeinflußt.

Aufbauend auf der in Kapitel "Semantische Integritätsbedingungen" (s. S. 73) aufgestellten Klassifizierung werden die **objekttypinternen Bedingungen für Einzelobjekte** mit Hilfe der CHECK-Anweisung dargestellt:

(1.1) Die Domänenbeschränkung kann sowohl für die Untergrenze als auch für die Obergrenze definiert werden:

```
TABLE A( a, Wert, ...,
PRIMARY KEY (a),
CHECK (Wert BETWEEN 1000 AND 8999))
```

(1.2) Über eine CHECK-Bedingung kann sichergestellt werden, daß der Wert eines Attributs mit der Ableitung aus anderen Attributen übereinstimmt:

```
TABLE A( a, Sollmenge, Istmenge, Abweichung, ...,
PRIMARY KEY (a),
CHECK (Abweichung = Istmenge - Sollmenge))
```

Darüberhinaus besteht auch die Möglichkeit, über eine View das Attribut nur virtuell zu definieren, so daß es bei Bedarf abgeleitet wird:

```
TABLE A( a, Sollmenge, Istmenge, ...,
PRIMARY KEY (a))

VIEW A2( a, Sollmenge, Istmenge, Abweichung, ...) AS
(SELECT a, Sollmenge, Istmenge, Istmenge - Sollmenge
 FROM A)
```

(1.3) Für die Formulierung bedingter Attribute müßte ein bedingter Vergleich oder eine bedingte Zuweisung möglich sein. Da ein IF-Operator nicht zur Verfügung steht, kann eine solche Bedingung **nicht** abgebildet werden.

(1.4) Wechselseitig abhängige Attribute können bsw. durch folgende CHECK-Bedingung formuliert werden:

```
TABLE A( a, Starttermin, Endtermin, ...,
PRIMARY KEY (a),
CHECK (Starttermin < Endtermin))
```

Für alle anderen Integritätsbedingungen, die auf mehr als ein Tupel wirken, kann zu deren Formulierung die CHECK-Bedingung nur dann genutzt werden, wenn durch den Überführungsprozeß in das Relationenmodell einzelne Entity- und Beziehungstypen in eine Relation überführt werden (s. dazu Kapitel "Entitytypen und Beziehungstypen", S. 89). Bsw. könnte eine **objektkombinationsabhängige Beziehung** bei einer binären Beziehung vom Typ (3) (s. Kapitel "Binäre Aggregation", S. 46) entsprechend Abbildung 113, bei der das Alter eines Meister immer größer als das Alter eines Auszubildenden ist, folgendermaßen abgebildet werden:

```
TABLE Personal( Meister, MeisterAlter,
                Azubi UNIQUE, AzubiAlter, ...,
PRIMARY KEY (Meister),
CHECK (MeisterAlter < AzubiAlter))
```

Ein solcher Anwendungsfall ist allerdings eher als unwahrscheinlich einzustufen. Daraus folgt, daß außer den oben gezeigten einzelobjektabhängigen Bedingungen und mit Einschränkungen der Komplexitätsgrade **keine** weiteren semantischen Integritätsbedingungen abgebildet werden können.

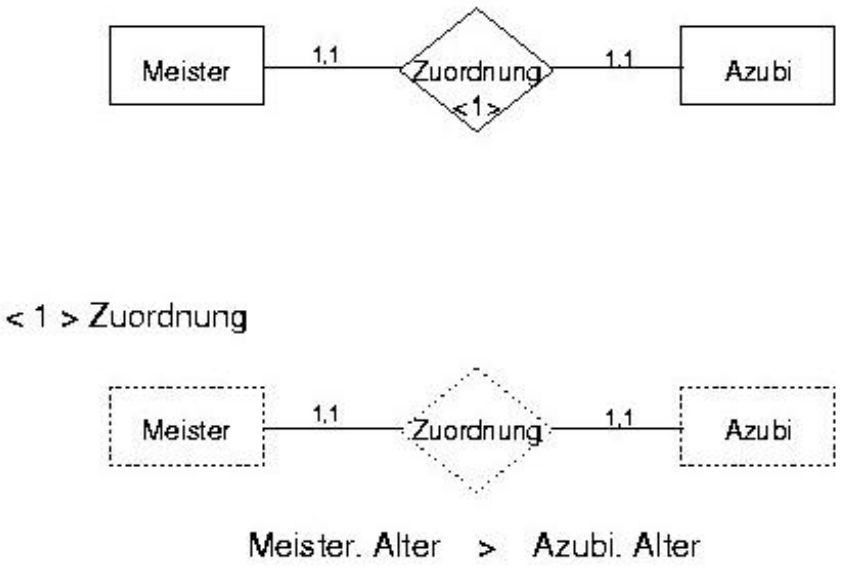


Abb. 113: PERM mit objektkombinationsabhängiger Beziehung

8.4 Erweiterungen des Relationenmodells

Neben den Datenbanksystemen, die sich an das Relationenmodell bzw. an die ISO-SQL-Definition anlehnen, gibt es bereits nichtstandardisierte Erweiterungen, die die semantische Ausdruckskraft und damit die Abbildungsmöglichkeiten vergrößern.

NF^2-Modelle

Während bei dem klassischen Relationenmodell nur Attribute mit skalaren Werten erlaubt sind, gestatten das NF^2- oder Non First Normal Form-Modell (Schek/Scholl 83, Dadam et al. 86), sowie das darauf aufbauende Extended NF^2-Modell (vgl. hierzu Elsholtz 89) und das dynamische NF^2- oder NF^{2D}-Modell (Benn 86) die Abbildung von nichtskalaren Werten und Relationen als Domäne eines Attributs. Der Verzicht auf die erste Normalform erlaubt somit die Nestung von Relationen in anderen Relationen. Dadurch können insbesondere hierarchische Strukturen gut modelliert werden. Abbildung 114 zeigt das Beispiel der Maschinen-Kostenstellengruppierung aus Abbildung 109 als NF^2-Relation:

Kostenstelle					
KoStNr	KoStName	...	Maschine		
			MaschNr.	MaschName	...
450	Galvanik	...	1	Chrombad	...
			2	Kupferbad	...
			3	Gebläse	...
510	Dreherei	...	101	Drehbank 1	...
			130	Zentrieren	...
			102	Drehbank 2	...

Abb. 114: Beispiele einer Maschinen-Kostenstellen-Gruppierung als NF^2-Relation

Datenbanksysteme, die das NF^2-Modell unterstützen, sind bereits im produktiven Einsatz (vgl. Saar 89).

Relationenmodell mit ereignisgesteuerten Trigger

Einige bereits in der Realisierung befindlichen Erweiterungen von Datenbanksystemen bieten die Möglichkeit, Prozeduren in einer höheren Programmiersprache zu schreiben und wie die Datenbeschreibung in die Datenbank aufzunehmen. Diese Prozeduren können dann automatisch beim Auftreten von definierten Ereignissen angestoßen (Triggern) werden (Stonebraker et al. 86). Datenbanksysteme mit dieser Möglichkeit werden auch als aktive Datenbanken (Dittrich 90) bezeichnet und im Rahmen der Normierungsbemühungen für einen zweiten Datenbanksystemstandard (SQL2) diskutiert (Olle 90). Zwar müssen auch hier die Bedingungen in formaler Sprache (Programmcode) spezifiziert werden, doch werden diese durch die Separierung von der Anwendungsapplikation und der Einlagerung in das Datenbanksystem Bestandteil der Datenstrukturen. Durch die Mächtigkeit der Programmiersprache können alle formulierbaren Integritätsbedingungen abgebildet werden. Im folgenden werden beispielhaft eine objektkombinationsabhängige Bedingung und eine Komplexitätsbedingung dargestellt. Die objektkombinationsabhängige Bedingung ist Abbildung 49 entnommen. Der Entitytyp *Mitarbeiter* und der Beziehungstyp *Hierarchie* sind als binäre, rekursive Beziehung von Typ (13) in eine Relation *Mitarbeiter* überführt worden. Die Prozedur "Mitarbeiter_Alter_Test" überprüft für ein gegebenes Tupel mit Hilfe eines

SELECT-Befehls, ob das Alter des referenzierten Vorgesetzten kleiner als das Alter des Mitarbeiters ist. Ist dies nicht der Fall oder ist kein Vorgesetzter definiert, so liefert die Prozedur eine OK-Meldung. Der Trigger "Mitarbeiter_Alter" gibt an, daß die Prozedur bei jeder INSERT- und UPDATE-Operation ausgeführt werden soll:

```
-     TABLE Mitarbeiter( PersNr, VorgPersNr, Alter, ...,
      PRIMARY KEY (PersNr),
      FOREIGN KEY (VorgPersNr) REFERENCES Mitarbeiter(PersNr))

-     PROCEDURE Mitarbeiter_Alter_Test( :TestVorg, :TestAlter )
      BEGIN
            IF ( :TestVorg IS NULL )
                  RETURN( Ok )
            ENDIF
            VorgAlter = SELECT Alter
                        FROM Mitarbeiter
                        WHERE PersNr = :TestVorg
            IF ( VorgAlter < TestAlter )
                  RETURN( Fehler )
            ENDIF
            RETURN( Ok )
      END

-     TRIGGER Mitarbeiter_Alter
      AFTER INSERT, UPDATE OF Mitarbeiter
      EXEC Mitarbeiter_Alter_Test( VorgPersNr, Alter )
```

Im zweiten Beispiel werden sowohl die Komplexität einer binären Beziehung mit Werten ungleich der 1,c,m-Notation sowie eine Beziehungspfadkomplexitätsbedingung dargestellt. Dies entspricht den Angaben aus Abbildung 53 (s. S. 81). Die Entitytypen sind in entsprechende Relationen überführt worden. Die Beziehungstypen können in die Relationen der Entitytypen integriert werden. Die Prozedur "WM_Komplex_Test" prüft für jedes Tupel der Relation *Maschine*, die angelegt oder geändert wurde, ob dem referenzierten Wartungsbeauftragen auch nicht mehr als 20 Maschinen zugeordnet sind.

Die Beziehungspfadkomplexität wird durch die FOREIGN KEY-Bedingungen in den Relationen *Maschine* und *MGruppe* beeinflußt. Deshalb werden zwei Trigger definiert, die bei neuen oder geänderten Tupeln der beiden Relationen deren Auswirkungen überprüfen. Die Prozedur "BeziehPfad_Komplex1_Test" testet über einen zweifach genesteten SELECT-Befehl die Anzahl der indirekten Referenzen auf die Relation *Wartbeauftragter* in der Relation *Kostenstelle* für einen bestimmten Wartungsbeauftragten, der durch das Attribut PersNr in einem Tupel der Relation *Maschine* determiniert wird. Die Prozedur "BeziehPfad_Komplex2_Test" testet die Auswirkungen auf alle betroffenen Beziehungspfade bei Änderungen einer Maschinengruppe. Dies sind alle Wartungsbeauftragten der zu der Gruppe gehörenden Maschinen. Dazu wird die maximale Anzahl der unterschiedlichen

Kostenstellen eines Wartungsbeauftragten ermittelt. Übersteigt die Anzahl den Wert 3, so ist die Integritätsbedingung verletzt.

```
-     TABLE Wartbeauftrag( PersNr, ...,
      PRIMARY KEY (PersNr))

-     TABLE Kostenstelle( KostNr, ...,
      PRIMARY KEY (KostNr))

-     TABLE MGruppe( MGrpNr, KoStNr NOT NULL, ...,
      PRIMARY KEY (MGrpNr),
      FOREIGN KEY (KoStNr) REFERENCES Kostenstelle(KoStNr))

-     TABLE Maschine( MaschNr, PersNr, MGrpNr NOT NULL, ...,
      PRIMARY KEY (MaschNr),
      FOREIGN KEY (PersNr) REFERENCES Wartbeauftrag(PersNr),
      FOREIGN KEY (MGrpNr) REFERENCES MGruppe(MGrpNr))

-     PROCEDURE WM_Komplex_Test( :WPersNrTest )
      BEGIN
            IF ( :WPersNrTest IS NULL )
                  RETURN( Ok )
            ENDIF
            Anzahl =    SELECT COUNT(*)
                        FROM Maschine
                        WHERE PersNr = :WPersNrTest
            IF ( Anzahl > 20)
                  RETURN( Fehler )
            ENDIF
            RETURN(Ok)
      END

-     TRIGGER WM_Komplex
      AFTER INSERT, UPDATE OF Maschine
      EXEC WM_Komplex_Test( PersNr )

-     PROCEDURE BeziehPfad_Komplex1_Test( :WPersNrTest)
      BEGIN
            IF ( :WPersNrTest IS NULL )
                  RETURN( Ok )
            ENDIF
            Anzahl =    SELECT COUNT(DISTINCT KoStNr)
                        FROM MGruppe
                        WHERE MGrpNr IN
                              (SELECT MGrpNr
                               FROM Maschine
                               WHERE PersNr = :WPersNrTest))
            IF ( Anzahl > 3 )
                  RETURN( Fehler )
            ENDIF
            RETURN( Ok )
      END
```

```
-     TRIGGER BeziehPfad_Komplex1
      AFTER INSERT, UPDATE OF Maschine
      EXEC BeziehPfad_Komplex1_Test( PersNr )

-     PROCEDURE BeziehPfad_Komplex2_Test( :TestMGrpNr )
      BEGIN
            Anzahl =    SELECT MAX(COUNT(DISTINCT KoStNr))
                        FROM Maschine, MGruppe
                        WHERE Maschine.MGrpNr = MGruppe.MGrpNr
                              AND Maschine.PersNr IN
                              (SELECT PersNr
                               FROM Maschine
                               WHERE MGrpNr = :TestMGrpNr)
                        GROUP BY PersNr
            IF ( Anzahl > 3 )
                  RETURN( Fehler )
            ENDIF
            RETURN( Ok )
      END

-     TRIGGER BeziehPfad_Komplex2
      AFTER INSERT, UPDATE OF MGruppe
      EXEC BeziehPfad_Komplex2_Test( MGrpNr )
```

Die Beispiele zeigen, daß mit einem Triggersystem prinzipiell alle Integritätsbedingungen abbildbar sind, die Formulierungen allerdings relativ aufwendig und unübersichtlich werden.

8.5 Überführung ausgewählter Strukturen des Referenzmodells

Durch fehlende semantische Ausdruckskraft des Relationenmodells kann nur ein Teil der Strukturinformationen des Referenzmodells direkt in SQL-Befehle umgesetzt werden. Insbesondere die vielfältig konstruierten Bedingungen der Relationship-Constraint-Diagramme können nicht dargestellt werden. Da die Überführung der Datenstrukturen anhand der genannten Regeln im Gegensatz zu der Modellierung ein rein formaler Akt ist, soll sie nur anhand von einigen Beispielen erfolgen. Auf eine komplette Transformation wird verzichtet.

Als Beispiele werden das Schichtmodell aus Abbildung 84 (s. S. 130), das zu einem Cluster zusammenfaßt wurde, die Arbeitsplandarstellung mit Hilfe von Zustandsgraphen aus Abbildung 95 (s. S. 142) sowie der Mehrfachbeziehungstyp Planbelegung für die Feinterminierung von Ressourcenbedarfen aus Abbildung 101 (s. S. 153) gewählt.

Aus Abbildung 84 ergeben sich die Relationen:

```
-     TABLE Schichtmodell( SMNr, ...,
      PRIMARY KEY (SMNr))

-     TABLE Schicht( SNr, Beginn, Dauer, ...,
      PRIMARY KEY (SNr))

-     TABLE Zyklustag( SMNr, ZTNr, SNr, ...,
      PRIMARY KEY (SMNr, ZTNr),
      FOREIGN KEY (SMNr) REFERENCES Schichtmodell(SMNr),
      FOREIGN KEY (SNr) REFERENCES Schicht(SNr))

-     TABLE Pause( PNr, PBeginn, PDauer, ...,
      PRIMARY KEY (PNr))

-     TABLE SchichtPause( SNr, PNr, ...,
      PRIMARY KEY (SNr,PNr)
      FOREIGN KEY (SNr) REFERENCES Schicht(SNr),
      FOREIGN KEY (PNr) REFERENCES Pause(PNr))
```

Abbildung 95 wird durch folgende Relationen repräsentiert:

```
-     TABLE Teil( TNr, ...,
      PRIMARY KEY (TNr))

-     TABLE APL( TNr, ...,
      PRIMARY KEY (TNr),
      FOREIGN KEY (TNr) REFERENCES Teil(TNr))

-     TABLE TZustand( TZNr, TNr, ...,
      PRIMARY KEY (TZNr),
      FOREIGN KEY (TNr) REFERENCES Teil(TNr))

-     TABLE AG( AGNr, TNr, ...,
      PRIMARY KEY (AGNr),
      FOREIGN KEY (TNr) REFERENCES APL(TNr))

-     TABLE TZKante( AGNr, VonTZNr, NachTZNr, ...,
      PRIMARY KEY (AGNr,VonTZNr,NachTZNr),
      FOREIGN KEY (AGNr) REFERENCES AG(AGNr),
      FOREIGN KEY (VonTZNr) REFERENCES TZustand(TZNr),
      FOREIGN KEY (NachTZNr) REFERENCES TZustand(TZNr))
```

Die Vierfachaggregation der Planungsbelegungen der Einzelressourcen aus Abbildung 101 ergibt die Relation:

```
-     TABLE Planbeleg( ERessNr, AAGNr, VonZeit NOT NULL,
                       BisZeit NOT NULL, ...,
      PRIMARY KEY (ERessNr,AAGNr),
      FOREIGN KEY (AAGNr) REFERENCES AAG(...),
      FOREIGN KEY (ERessNr) REFERENCES Einzelressource(ERessNr),
      CHECK (VonZeit < BisZeit))
```

9. Zusammenfassung und Ausblick

Die Methodik des in der Arbeit entwickelten Expanded Entity-Relationship-Modells enthält einen umfassenden Ansatz für die Abbildung der in einem industriellen Fertigungsprozeß auftretenden Datenstrukturen, bei der alle zu den Daten gehörenden Integritätsbedingungen abgedeckt sind. Dabei wurde, soweit möglich, auf Sprachkonstrukte bestehender Modelle zurückgegriffen. Für bisher nicht abgedeckte Datenstruktursemantik wurden neue Sprachkonstrukte erarbeitet. Mit Hilfe der entwickelten Methodik wurde ein Referenzmodell für die Fertigungsbereiche Fertigungssteuerung, Betriebsdatenerfassung und -verarbeitung, Lager- und Transportsteuerung, Prüfausführung und Instandhaltung entworfen. Des weiteren wurde die Möglichkeiten aufgezeigt, inwieweit diese Datenstrukturen in dem Relationenmodell abgebildet werden können, und bei welchen Konstruktionsoperatoren aufgrund semantischer Lücken keine direkte Umsetzung vorgenommen werden kann.

Die aufgezeigten Abbildungsoperatoren der Beschreibungssprache lassen sich auch in anderen Anwendungsgebieten nutzen. Dies gilt beispielsweise für die Spezifizierung der Aggregationsarten sowie für die beziehungstypabhängigen Integritätsbedingungen, die durch die Relationship-Constraint-Diagramme dargestellt werden. Insoweit kann das Expanded Entity-Relationship-Modell auch als allgemeine Datenmodellbeschreibungssprache verstanden werden, die die komplette Semantik statischer Datenstrukturen abdeckt.

Seitens der Anwendungsentwicklung ist ein zunehmendes Bewußtsein für die Bedeutung der Modellierungsebene bei der Gestaltung von Informationssystemen zu verzeichnen, was sich nicht zuletzt durch eine am Markt zu beobachtende Entwicklung bezüglich des Angebots an CASE-Tools (Computer Aided Software Engineering) widerspiegelt. Bei den bestehenden Datenbanksystemen ist kann festgestellt werden, daß diese zu mächtigen Entwicklungswerkzeugen weiterentwickelt und mit bisher isolierten CASE-Tools verbunden werden. Dabei fehlt allerdings noch ein umfassendes Methodenkonzept, das sowohl alle Teilmodelle eines Informationsmodells abdeckt als auch alle Architekturebenen befriedigend unterstützt. Als zukünftiges Aufgabenfeld sind u. a. noch folgende Problemstellungen zu lösen:

- Vollständige Abdeckung aller Teilmodelle eines Informationsmodells mit geeigneten, aufeinander abgestimmten Methoden, wie dies für die Datenmodellierung in dem vorliegenden Ansatz gezeigt wurde.

- Unterstützung der Anwendungsentwicklung mit CASE-Tools, die nach einer exakten Definition des Informationsmodells die Umsetzungsphase in ein Informationssystem weitgehendst automatisieren.

- Weiterentwicklungen der Datenbanksysteme, so daß diese die gesamte Datensemantik aufnehmen können, wie dies am Beispiel des mit einem Triggerkonzept erweiterten Relationenmodells (aktive Datenbank) erläutert wurde.

Literaturverzeichnis

Abiteboul/Hull 87
Abiteboul, S.; Hull, R.: IFO: A Formal Semantic Database Model, in: ACM Transactions on Database Systems, 12 (1987), S. 525 - 565.

Adam 87
Adam, D.: Ansätze zu einem integrierten Konzept der Fertigungssteuerung bei Werkstattfertigung, in: Adam, D. (Hrsg.), Neuere Entwicklungen in der Produktions- und Investitionspolitik, Wiesbaden 1987.

Adam 88
Adam, D.: Fertigungssteuerung I, Grundlagen der Produktionsplanung und -steuerung, Schriften zur Unternehmensführung Band 38, Wiesbaden 1988.

Aldinger 85
Aldinger, L.: Leitstandunterstützte kurzfristige Fertigungssteuerung bei Einzel- und Kleinserienfertigung, Berlin-Heidelberg-New York-Tokio 1985.

Allen 83
Allen, J. F.: Maintaining knowledge about temporal intervals, in: Communications of the ACM, 26 (1983), S. 832 ff.

Appelrath 85
Appelrath, H.-J.: Von Datenbanken zu Expertensystemen, Berlin-Heidelberg-New York-Tokio 1985.

Atkinson et al. 89
Atkinson, M.; Bancilhon, F.; De Witt, D.; Dittrich, K. R.; Maier, D.; Zdonik, S.: The Object-Oriented Database System Manifesto, Rapport Technique, Altaïr 1989.

Aue et al. 90
Aue, D., Baresch, M., Keller, G.: URMEL: Ein Unternehmensmodellierungsansatz, in: Scheer, A.-W. (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 71, Saarbrücken 1990.

Batini et al. 86
Batini, C.; Lenzerini, M.; Navathe, S. B.: A Comparative Analysis of Methodologies for Database Schema Integration, in: ACM Computing Surveys, 18 (1986), S. 323 - 364.

Batra et al. 90
Batra, D.; Hoffer, J. A.; Bostrom, R. P.: Comparing Representations with Relational and EER Models, in: Communications of the ACM, 33 (1990), S. 126 - 139.

Becker 87
Becker, J.: Architektur eines EDV-Systems zur Materialflußsteuerung, Berlin-Heidelberg-New York-Tokio 1987.

Becker 90
Becker, J.: Die Integration in CIM-Systemen - Notwendigkeit und Realisierungsmöglichkeiten der EDV-gestützten Verbindung betrieblicher Bereiche, Habilitationsschrift Universität Saarbrücken 1990.

Becker/Keller 89
Becker, J.; Keller, G.: Datentechnische und organisatorische Aspekte einer CIM-Realisierung, in: Information Management, 4 (1989), Heft 3, S. 20 - 27.

Benn 86
Benn, W.: Dynamische nicht-normalisierte Relationen und symbolische Bildbeschreibung, Berlin-Heidelberg-New York-Tokio 1986.

Blumental/Landwehr 85
Blumental, R.; Landwehr, K.: Einsatz des offenen Echtzeit-Datenbanksystems BAPAS-DB in der industriellen Anwendung mit hohen Datenraten, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 96 - 100.

Böhnlein et al. 90
Böhnlein, P. G.; Nittel, S.; Dittrich, K. R.: Semantische Datenmodelle, in: HMD - Theorie und Praxis der Wirtschaftsinformatik, 27 (1990), Heft 152, S. 116 - 127.

Bolour et al. 82
Bolour, A.; Anderson, T. L.; Dekeyser, L. J.; Wong, H. K. T.: The role of time in information processing: a survey, in: ACM SIGMOD Record, 12 (1982), No. 3.

Brachman 79
Brachman, R. J.: On the Epistemological Status of Semantic Networks, in: Findler, N. J. (Hrsg.), Associative Networks, New York-San Francisco 1979, S. 3 - 50.

Briand et al. 88a
Briand, H.; Ducateau, C.; Hebrail, Y.; Herin-Aime, D.; Kouloumdjian, J.: From Minimal Cover to Entity-Relationship Diagram, in: March, S. T. (Hrsg.), Entity-Relationship Approach (Proceedings of the Sixth International Conference on Entity-Relationship Approach, New York 1987), Amsterdam-New York-Oxford-Tokyo 1988, S. 287 - 304.

Brodie 83
Brodie, M. L.: Association: A Database Abstraction for Semantic Modeling, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 577 - 602.

Brodie 84
Brodie, M. L.: On the Development of Data Models, in: Brodie et al. (Hrsg.), On Conceptual Modelling, Berlin-Heidelberg-New York-Tokio 1984, S. 19 - 47.

Brombacher 88
Brombacher, R.: Entscheidungsunterstützungssysteme für das Marketing-Management, Berlin-Heidelberg-New York-Tokio 1988.

Bullinger/Lampkemeyer 89
Bullinger, H.-J.; Lampkemeyer, U.: Relationale Datenbanken - Kern der rechnerintegrierten Produktion, in: CIM Management, 5 (1989), Heft 6, S. 28 - 33.

Caldiera/Quitadamo 83
Caldiera, G.; Quitadamo, P.: Conceptual Representation of Data and Logical IMS Design, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 299 - 317.

Cammarata et al. 89
Cammarata, S.; Ramachandra, P.; Shane, D.: Extending a Relational Database with Deferred Referential Integrity Checking and Intelligent Joins, in: ACM SIGMOD Record, 18 (1989) S. 88 - 97.

Chen 76
Chen, P. P.: The Entity-Relationship Model - Toward a Unified View of Data, in: ACM Transactions on Database Systems, 1 (1976), S. 9 - 36.

Chen 83
Chen, P. P.: A Preliminary Framework for Entity-Relationship Model, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 19 - 28. er 2.ER 1981

Codd 70
Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, in: Communications of the ACM, 13 (1970), S. 377 - 387.

Codd 79
Codd, E. F.: Extending a Database Relational Model to Capture More Meaning, in: ACM Transaction on Database Systems, 4 (1979), S. 397 - 434.

Codd 86
Codd, E. F.: Missing Information (Applicable and Inapplicable) in Relational Databases, in: ACM SIGMOD Record, 15 (1986), S. 53 - 78.

Codd 87
Codd, E. F.: More Commentary on Missing Information in Relational Databases, in: ACM SIGMOD Record, 16 (1987), S. 42 - 50.

Dadam et al. 86
Dadam, P.; Küspert, K.; Andersen, F.; Blanken, H.; Erbe, R.; Günauer, J.; Lum, V.; Pistor, P.; Walch, G.: A DBMS Prototype to Support Extended NF^2 Relations: An Integrated View on Flat Tables and Hierarchies, in: ACM SIGMOD Record, 15 (1986), S. 356 - 367.

Date 81
Date, C. J.: An Introduction to Database Systems Reading, Vol. I, Massachusetts, 1981.

Date 83
Date, C. J.: An Introduction to Database Systems Reading, Vol. II, Massachusetts, 1983.

Davis/Bonell 89
Davis, J. P.; Bonnell, R. D.: Modeling Semantic Constraints with Logic in the EARL Data Model, in: Proceedings of the Fifth International Conference on Data Engineering, Los Angeles, 1989, S. 226 - 233.

DeMarco 78
de Marco, T.: Structured Analysis and System Specification, New York 1978.

Deppisch et al. 85a
Deppisch, U.; Günauer, J.; Walch, G.: Speicherungsstrukturen und Adressierungstechniken für Komplexe Objekte des NF^2-Relationenmodells, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 441 - 459.

Deppisch et al. 85b
Deppisch, U.; Obermeit, V.; Paul, H.-B.; Schek, H.-J.; Scholl, M.; Weikum, G.: Ein Subsystem zur stabilen Speicherung versionsbehafteter, hierarchisch strukturierter Tupel, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 421 - 429.

Destunis 80
Destunis, S.: Die Datenbank - Ihre Voraussetzung und Nutzung in der Prozeßdatenverarbeitung als Prozeßdatenbank, Dissertation Technische Hochschule Aachen 1980.

DeTroyer 89
de Troyer, O.: RIDL: A Tool for the Computer-Assisted Engineering of Large Databases in the Precent of Integrity Constraints, in: ACM SIGMOD ,Record, 18 (1989), S 418 - 429.

DiBattista/Lenzerini 88
Di Battista, G.; Lenzerini, M.: Object Modeling Based on Logic, in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 77 - 95.

Dittrich 89
Dittrich, K. R.: Objektorientierte Datenbanksysteme, in: Informatik Spektrum, 12 (1989), S. 215 - 218.

Dittrich 90
Dittrich, K. R.: Objektorientierte Datenmodelle als Basis komplexer Anwendungssysteme - Stand der Entwicklung und Einsatzperspektiven, in: Wirtschaftsinformatik, 32 (1990), S. 228 - 237.

Dittrich et al. 85
Dittrich, K. R.; Kotz, A. M.; Mülle, J. A.: Basismechanismen für komplexe Konsistenzprobleme in Entwurfsdatenbanken, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 70 - 72.

Dogac/Chen 83
Dogac, A.; Chen, P. P.: Entity-Relationship Model in ANSI/SPARC Framework, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 357 - 374.

Döttling 81
Döttling, W.: Flexible Fertigungssysteme: Steuerung und Überwachung des Fertigungsablaufs, Berlin-Heidelberg-New York-Tokio 1981.

Eberlein 84
Eberlein, W.: CAD-Datenbanksysteme - Architektur Technischer Datenbanken für Integrierte Ingenieursysteme, Berlin-Heidelberg-New York-Tokio 1984.

Eick 84
Eick, C. F.: Methoden und rechnergestützte Werkzeuge für den logischen Datenbankentwurf, Dissertation Universität Karlsruhe 1984.

Elmasri et al. 85
Elmasri, R.; Weeldreyer, J.; Hevner, A.: The Category Concept: An Extension to the Entity-Relationship Approach, in: Data & Knowledge Engineering, 1 (1985), S. 75 - 116.

Elmasri et al. 90
Elmasri, R.; El-Assal, I.; Kouramajian, V.: Semantics of temporal data in an extended ER model, in: Kangassalo, H. (Hrsg.), Proceedings of the Ninth International Conference on Entity-Relationship Approach, Lausanne 1990, S. 249 - 264.

Elsholtz 89
Elsholtz, A.: Konzepte und Werkzeuge zur Integration von Standard- und Non-Standard-Datenbanksystemen im CIM-Bereich, Berlin 1989.

Eversheim/Wienand 87
Eversheim,W.; Wienand, L.: Aufbau einer überbetrieblichen Werkzeugdatenbank, in: VDI-Z, 129 (1987), S. 95 - 100.

Feldmann/Miller 86
Feldmann, P.; Miller, D.: Entitiy Model Clustering: Structuring a Data Model by Abstraction, in: The Computer Journal, 19 (1986), S. 348 - 361.

Ferg 85
Ferg S.: Modelling the Time Dimension in an Entity-Relationship Diagram, in: Chen P. P. (Hrsg.), Entity-Relationship Approach, The Use of ER Concept in Knowledge Representation (Proceedings of the Fourth International Conference on Entity-Relationship Approach, Chicago 1985), Amsterdam-New York-Oxford 1985, S. 280 - 286.

Fischer 89
Fischer, W.: Technische Datenbanken- eine Lücke in der Praxis, in: CIM Management, 5 (1989), Heft 6, S. 16 - 22.

Gebhardt 87
Gebhardt, F.: Semantisches Wissen in Datenbanken - Ein Literaturbericht, in: Informatik Spektrum, 10 (1987), S. 79 - 98.

Grill et al. 87
Grill, E.; Flittner, J.; Rauch, W.: Integration von CAD/CAE/CAM über Relationale Datenbanken, in: Information Management, 2 (1987), Heft 1, S. 54 - 64.

Hagelstein/Rifaut 88
Hagelstein, J.; Rifaut, A.: A Semantic Analysis of the Collection Concept, in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 21 - 36.

Hainaut 90
Hainaut, J.-L.: Entity-Relationship models: formal specification and comparison, in: Kangassalo, H. (Hrsg.), Proceedings of the Ninth International Conference on Entity-Relationship Approach, Lausanne 1990, S. 53 - 66.

Hammer/McLeod 81
Hammer, M.; Mc Leod, D.: Database Description with SDM: A Semantic Database Model, in: ACM Transaction on Database Systems, 6 (1981), S. 351 - 386.

Härder 84
Härder, T.: Überlegungen zur Modellierung und Integration der Zeit, SFB 124, Report 19/84, Kaiserslautern 1984.

Härder 85
Härder, T.; Reuter, A.: Architektur von Datenbanksystemen für Non-Standard-Anwendungen, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 253 - 286.

Härder 89a
Härder, T.: Grenzen und Erweiterungsmöglichkeiten relationaler Datenbanksysteme für Nicht-Standard-Anwendungen, in: Scheer, A.-W. (Hrsg.), Praxis Relationaler Datenbanken, Saarbrücken 1989, S. 1 - 25.

Härder 89b
Härder, T.: Die Rolle von Datenbanken in CIM, in: CIM Management, 5 (1989), Heft 6, S. 4 - 10.

Härder et al. 85
Härder, T.; Keller, W.; Mitschang, B.; Siepmann, E.; Zimmermann, G.: Datenstrukturen und Datenmodelle für den VLSI-Entwurf, SFB 124, Report 26/85, Kaiserslautern 1985.

Helberg 87
Helberg, P.: PPS als CIM-Baustein, Berlin 1987.

Heß 89
Heß, H.: Objektktorientierter Systementwurf, in: Information Management, 4 (1989), Heft 3, S. 76 - 77.

Heuer 88
Heuer, A.: Exakte Charakterisierung eines semantischen Datenmodells und seiner Operationen durch relationale Konzepte, Dissertation Technische Universität Clausthal 1988.

Hohenstein et al. 87
Hohenstein, U.; Neugebauer, L.; Saake, G.; Ehrich, H.-D.: Three-Level-Specification of Databases Using an Extended Entity-Relationship Model, in: Wagner, R. R. et al. (Hrsg.), Informationsbedarfsermittlung und -analyse für den Entwurf von Informationssystemen, Berlin-Heidelberg-New York-Tokio 1987, S. 58 - 88.

Hohenstein/Gogolla 88
Hohenstein, U.; Gogolla, M.: A Calculus for an Extended Entity-Relationship Model Incorporating Arbitrary Data Operations and Aggregate Functions, in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 1 - 20.

Hopcroft/Ullman 69
Hopcroft, J. E.; Ullman, J. D.: Formal Languages and their Relation to Automata, Readings-Menlo Park-London-Don Mills, 1969.

ISO 89
ISO/IEC 9075, JTC 1/SC 21/WG 3, 1989-04-01, Information processing system, Database language SQL with integrity enhancement, 1989.

ISO 90
ISO/IEC, JTC 1/SC 21/WG 3, N1132, Workingdraft, Reference Model of Data Management (CD2 10032), 1990.

Jablonski 90
Jablonski, S.: Datenverwaltung in verteilten Systemen, Berlin-Heidelberg-New York-Tokio 1990.

Jablonski et al. 88
Jablonski, S.; Ruf, T.; Wedekind, H.: Implementierung eines verteilten Datenverwaltungssystems für technische Anwendungen, in: Valk, R. (Hrsg.), Vernetzte und komplexe Informatik-Systeme II, Berlin-Heidelberg-New York-Tokio 1988, S. 639 - 657.

Kang 87
Kang, M.: Entwicklung eines Werkstattsteuerungssystems mit simultaner Termin- und Kapazitätsplanung, in: Spur, G. (Hrsg.), Produktionstechnik Berlin Forschungsberichte für die Praxis 55, München-Wien 1987.

Kappe/Suer 88
Kappe, T.; Suer, K.: CADD- Computer aided database design, in: Angewandte Informatik, 30 (1988), S. 496 - 477.

Katz et al. 86
Katz, R. H.; Chang, E.; Bhateja, R.: Version Modeling Concepts for Computer-Aided Design Databases, in: ACM SIGMOD Record, 15 (1986), S. 379 - 386.

Kemke 88
Kemke, C.: Der Neuere Konnektionismus - Ein Überblick, in: Informatik Spektrum, 11 (1988), S. 143 - 162.

Kent 83
Kent, W.: A Simple Guide to Five Normal Forms in Relational Database Theory, in: Communications of the ACM, 26 (1983), S. 121 - 125.

Kim 90
Kim, W.: Object-Oriented Databases: Definition and Research Directions, in: IEEE Transaction on Knowledge and Data Engineering, 2 (1990), S. 327 - 341.

Kinzer 71
Kinzer, D.: Ein Verfahren zum mittelfristigen Kapazitätsabgleich bei Werkstattfertigung, Dissertation Technische Hochschule Aachen 1971.

Kinzinger 83
Kinzinger, H.: Erweiterung einer Datenbank-Anfragesprache zur Unterstützung des Versionskonzeptes, in: Schmidt, J. W. (Hrsg.), Sprachen für Datenbanken, Berlin-Heidelberg-New York-Tokio 1983, S. 96 - 112.

Klein 90
Klein, J.: Vom Informationsmodell zum integrierten Informationssystem, in: Information Management, 5 (1990), Heft 2, S. 6 - 16.

Klopprogge 83a
Klopprogge, M. R.: TERM: An Approach to Include the Time Dimension in the Entity-Relationship Model, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 473 - 508.

Klopprogge 83b
Klopprogge, M. R.: Gegenstands- und Beziehungsgeschichten: ein Konzept zur Beschreibung und Verwaltung zeitveränderlicher Informationen in Datenbanken, Dissertation Universität Karlsruhe 1983.

Köhl 89
Köhl, E.: Datenverteilung - eine wesentliche Voraussetzung für robuste CIM-Systeme, in: CIM Management, 5 (1989), Heft 6, S. 46 - 49.

Köhl/Malsbender 88
Köhl, E.; Malsbender G.: Anforderung an Systemstrukturen zur Integration von flexiblen Fertigungssystemen in die Werkstattsteuerung, in: CIM Management, 4 (1988), Heft 3, S. 40 - 45.

Kratzer/Schreier 85
Kratzer, K.; Schreier, U.: Behandlung von Ausnahmesituationen mit einer Metadatenbank, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 177 - 198.

Kreutzer 90
Kreutzer, W.: Grundkonzepte und Werkzeugsysteme objektorientierter Systementwicklung - Stand der Forschung und Anwendung, in: Wirtschaftsinformatik, 32 (1990), S. 211 - 227.

Kung 90
Kung, C.: Object Subclass Hierarchy in SQL: A simple Approach, in: Communications of the ACM, 33 (1990), S. 117 - 125.

Küspert 89
Küspert, K.: Non-Standard-Datenbanksysteme, in: Informatik Spektrum 9 (1986), S. 184 - 185.

Lambrecht 80
Lambrecht, B.: Konzept für eine interaktive Datenbank-Entwurfs-Methodik, Dissertation Technische Universität Berlin 1980.

Lamersdorf 85
Lamersdorf, W.: Semantische Repräsentation komplexer Objektstrukturen: Modelle für nichtkonventionelle Datenbankanwendung, Berlin-Heidelberg-New York-Tokio 1985.

Lamersdorf/Schmidt 83
Lamersdorf, W.; Schmidt, J. W.: Rekursive Datenmodelle, in: Schmidt, J. W. (Hrsg.), Sprachen für Datenbanken, Berlin-Heidelberg-New York-Tokio 1983, S. 148 - 168.

Lampkemeyer et al. 89
Lampkemeyer, U.; Dittmer, H.; Petersen, W.: Werkzeugwesen als Teil des Ressourcenmanagements, in: ZwF 84 (1989), S. 388 - 391.

Lausen/Marx 90
Lausen, G.; Marx, B.: Das Relationenmodell und die Normalisierung, in: HMD - Theorie und Praxis der Wirtschaftsinformatik, 27 (1990), Heft 152, S. 30 - 42.

Lenz 90
Lenz, H.-J.: Metadaten, in: Mertens, P. et al. (Hrsg.), Lexikon der Wirtschaftsinformatik, 2. Auflage, Berlin-Heidelberg-New York-Tokio 1990, S. 279.

Lenzerini/Santucci 83
Lenzerini, M.; Santucci, G.: Cardinality Constraints in the Entity-Relationship Model, in: Davis C. G. et al. (Hrsg.), Entity-Relationship Approach to Software Engineering (Proceedings of the Third International Conference on Entity-Relationship Approach, Anaheim 1983), Amsterdam-New York-Oxford 1983, S. 529 - 549.

Ley 84
Ley, W.: Entwicklung von Entscheidunghilfen zur Integration der Fertigungshilfsmitteldisposition in EDV-gestützten Produktionsplanungs- und -steuerungssystemen, Forschungsberichte der VDI, Reihe 2, Düsseldorf 1984.

Lin et al. 89
Lin, C.-C.; Mark, L.; Sellis, T.; Faloutsos, C.: Performance issues in the binary relationship model, in: Data & Knowlegde Engineering, 4 (1989), S. 195 - 221.

Ling 85
Ling, T.-W.: An analysis of multivalued and join dependencies based on the entity-relationship approach, in: Data & Knowlegde Engineering, 1 (1985), S. 253 - 271.

Lipeck 82
Lipeck, U. W.: Ein algebraischer Kalkül für einen strukturierten Entwurf von Datenabstraktionen, Dissertation Universität Dortmund 1982.

Lipeck 89
Lipeck, U. W.: Dynamische Integrität von Datenbanken - Grundlagen der Spezifikation und Überwachung, Berlin-Heidelberg-New York-Tokio 1989.

Lockemann et al. 85
Lockemann, P. C.; Adams, M.; Bever, M.; Dittrich, K. R.; Ferkinghoff, B.; Gotthard, W.; Kotz, A. M.; Liedtke, R.-P.; Lüke, B.; Mülle, J. A.: Anforderungen technischer Anwendungen an Datenbanksysteme, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 1 - 26.

Lockemann/Rademacher 90
Lockemann, P. C.; Rademacher; K.: Konzepte, Methoden und Modelle zur Datenmodellierung, in: HMD - Theorie und Praxis der Wirtschaftsinformatik, 27 (1990), Heft 152, S. 3 - 16.

Loos 87
Loos, P.: Feinplanung und -steuerung in Fertigungsinseln - Konzeption des Intelligenten Leitstandes FI-2, in: Scheer, A.-W. (Hrsg.), GI-Fachausschuß Informatik in Produktion und Materialwirtschaft, Leitstandskonzepte im Rahmen von PPS/CIM, Frankfurt 1987, S. 1 - 23.

Loos 89
Loos, P.: Die relationale Datenbank im Mittelpunkt der Standardsoftware FI-2 zur Steuerung von Fertigungsleitständen, in: Scheer, A.-W. (Hrsg.), Praxis Relationaler Datenbanken, Saarbrücken 1989, S. 207 - 221.

Loos 90a
Loos, P.: Datenbankeinsatz in der Fertigung - Probleme und Lösungsansätze, in: Scheer, A.-W. (Hrsg.), Datenbanken 1990 - Praxis relationaler Datenbanken, Saarbrücken 1990, S. 389 - 419.

Loos 90b
Loos, P.: Leitstand, in: Mertens, P. et al. (Hrsg.), Lexikon der Wirtschaftsinformatik, 2. Auflage, Berlin-Heidelberg-New York-Tokio 1990, S. 259.

Loos et al. 89
Loos, P.; Juen, G.; Scheer, A.-W.: Produktionsplanung und -steuerung - PPS, München-Wien 1989.

Loos/Ruffing 86
Loos, P.; Ruffing, T.: Verteilte Produktionsplanung und -steuerung unter Einsatz von Mikrocomputern, in: Scheer, A.-W. (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 52, Saarbrücken 1986.

Luft 90
Luft, A. L.: Datenmodelle, in: Mertens, P. et al. (Hrsg.), Lexikon der Wirtschaftsinformatik, 2. Auflage, Berlin-Heidelberg-New York-Tokio 1990, S. 132 - 133.

Markowitz/Shoshani 89
Markowitz, V. M.; Shoshani, A.: On the Correctness of Representing Extented Entity-Relationship Structures in the Relational Model, in: ACM SIGMOD Record, 18 (1989), S. 430 - 439.

Marti 83
Marti, R. W.: Integrating Database and Program Descriptions Using an ER-Data Dictionary, in: Davis C. G. et al. (Hrsg.), Entity-Relationship Approach to Software Engineering (Proceedings of the Third International Conference on Entity-Relationship Approach, Anaheim 1983), Amsterdam-New York-Oxford 1983, S. 377 - 392.

Martin 87
Martin J.: Einführung in die Datenbanktechnik, München-Wien 1987.

Meier 87
Meier, A.: Erweiterung relationaler Datenbanksysteme für technische Anwendungen, Berlin-Heidelberg-New York-Tokio 1987.

Mitschang 85
Mitschang, B.: Charakterisierung des Komplex-Objekt-Begriffs und Ansätze zu dessen Realisierung, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 382 - 400.

Mitschang 88
Mitschang, B.: Ein Molekül-Atom-Datenmodell für Non-Standard-Anwendungen, Berlin-Heidelberg-New York-Tokio 1988.

Müller-Ettrich 89
Müller-Ettrich, G. (Hrsg.): Effektives Datendesign - Praxis-Erfahrungen, Köln 1989.

Müller/Steinbauer 83
Müller, Th.; Steinbauer, D.: Eine Sprachschnittstelle zur Versionskontrolle in CAM-Datenbanken, in: Schmidt, J. W. (Hrsg.), Sprachen für Datenbanken, Berlin-Heidelberg-New York-Tokio 1983, S. 76 - 95.

Mylopoulos/Brodie 89
Mylopoulos, J.; Brodie, M. L. (Hrsg.): Readings in Artificial Intelligence and Databases, San Mateo 1989.

Navathe et al. 86
Navathe, S. B.; Elmasri, R.; Larson, J.: Integrating User Views in Database Design, in: Computer, 19 (1986), S. 50 - 62.

Navathe/Pillalamarri 88
Navathe, S. B.; Pillalamarri, M. K.: OOER: Toward Making the ER Approach Object Oriented; in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 55 - 76.

Nissing 82
Nissing, Th.: Beitrag zur Entwicklung eines dezentralen Produktionsplanungs- und -steuerungssystems auf der Basis verteilter Datenbestände, Dissertation Technische Hochschule Aachen 1982.

Nittel 89
Nittel, S.: Relationale und objektorientierte Datenbanksysteme für CIM-Applikationen - ein Vergleich, in: CIM Management, 5 (1989), Heft 6, S. 11 - 14.

Olle 90
Olle, T. W.: Expressing Relationships using Constraints in ISO SQL2, Arbeitspapier, Ninth International Conference on Entity-Relationship Approach, Lausanne 1990.

Ortner 83
Ortner, E.: Der Begriffskalkül - eine Konstruktionssprache für die Spezifikation von Datenbankanwendungen auf der Ebene der Benutzer, in: Schmidt, J. W. (Hrsg.), Sprachen für Datenbanken, Berlin-Heidelberg-New York-Tokio 1983, S. 169 - 182.

Ortner 85
Ortner, E.: Semantische Modellierung - Datenbankentwurf auf der Ebene der Benutzer, in: Informatik Spektrum, 8 (1985), S. 20 - 28.

Ortner/Söllner 88
Ortner, E.; Söllner, B.: Data Dictonary, in: Information Management, 3 (1988), Heft 3, S. 26 - 33.

Ortner/Söllner 89
Ortner, E.; Söllner, B.: Semantische Datenmodellierung nach der Objekttypenanalyse, in: Informatik Spektrum, 12 (1989), S. 31 - 42.

Österle/Brenner 86
Österle, H.; Brenner, W.: Integration durch Synonymerkennung, in: Information Management, 1 (1986), Heft 2, S. 54 - 62.

Parent/Spaccapietra 90
Parent, C.; Spaccapietra. S.: About entities, complex objects and object-oriented data models, École Polytechnique Fédérale de Lausanne, Département d'Informatique, Lausanne 1990.

Petersen/Trimm 87
Petersen, W.; Trimm, H. E.: Werkzeugverwaltung auf Basis einer relationalen Datenbank, AWF-Fachtagung Rüstzeitverkürzung, Eschborn 1987, S. 113 - 135.

Pistorius 85
Pistorius, E.: Informationsabbildungen für automatisierte Arbeitsplanung, c.G.Spur München Wien 1985. Carl Hanser Verlag arbeitsplanung cap CA 1 639 SB

Put 88
Put, F.: The ER Approach Extended with the Action Concept as a Conceptual Modelling Tool, in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 283 - 300.

Rahm 89
Rahm, E.: Der Database-Sharing-Ansatz zur Realisierung von Hochleistungs-Transaktionen, in: Informatik Spektrum, 12 (1989), S. 65 - 81.

Rebsam 83
Rebsam, J.: Datenbankentwurf im Dialog - Integrierte Beschreibung von Strukturen, Transaktionen und Konsistenz, Dissertation ETH Zürich 1983.

Reuter 81
Reuter, A.: Fehlerbehandlung in Datenbanksystemen - Datenbank-Recovery, München-Wien 1981.

Rochfeld et al. 90
Rochfeld, A.; Morejon, J.; Negros, P.: Inter-Relationship Links in E-R Model, in: Kangassalo, H. (Hrsg.), Proceedings of the Ninth International Conference on Entity-Relationship Approach, Lausanne 1990, S. 143- 156.

Röhrle/Kratzer 88
Röhrle, J.; Kratzer, K.: Verwaltung von Integritätsbedingungen in einem Wörterbuch, in: Angewandte Informatik, 30 (1988), S. 18 - 26.

Rowe/Stonebraker 87
Rowe, L. A.; Stonebraker; M. R.: The Postgres Data Model, in: Stonebraker; M. R.; Rowe, L. A. (Hrsg.), The Postgres Papers, Memorandum No. UCB/ERL M86/85, University of California, Berkeley 1987, S. 33 - 46.

Ruffing 90
Ruffing, T.: Fertigungssteuerung bei Fertigungsinseln - Eine funktionale und datentechnische Informationsarchitektur, Saarbrücken 1990.

Saar 89
Saar, A.: Rationalisierung durch Einsatz von Werkzeug-Datenbanken mit PISA, in: Werkstatt und Betrieb, 122 (1989), S. 246 - 248.

Sanden 89
Sanden, B.: An Entity-Life Modeling Approach to the Design of Concurrent Software, in: Communications of the ACM, 32 (1989), S. 330 - 343.

Schacher/Winkler 89
Schacher, D.; Winkler, K.: Ingenieurdatenbanken zur Integration von CAD/CAM, in: CIM Management, 5 (1989), Heft 6, S. 23 - 27.

Schäfer 86
Schäfer, G.: Entwurf logischer Datenstrukturen für konzeptionelle Schemata von Datenbanken, in: Planung, Information und Unternehmensführung Band 12, Bergisch Gladbach 1986.

Scheer 74
Scheer, A.-W.: Instandhaltungspolitik, Wiesbaden 1974.

Scheer 76
Scheer, A.-W.: Produktionsplanung auf der Grundlage einer Datenbank des Fertigungsbereichs, München Wien 1976.

Scheer 86
Scheer, A.-W.: Neue Architektur für EDV-Systeme zur Produktionsplanung und -steuerung, in: Scheer, A.-W. (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 53, Saarbrücken 1986.

Scheer 88a
Scheer, A.-W.: Entwurf eines Unternehmensdatenmodells, in: Information Management, 3 (1988), Heft 1, S. 14 - 23.

Scheer 88b
Scheer, A.-W.: Enterprise wide Data Model (EDM) as a Basis for Integrated Information Systems, in: Scheer, A.-W. (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 56, Saarbrücken 1988.

Scheer 89
Scheer, A.-W.: Unternehmens-Datenbanken - Der Weg zu bereichsübergreifenden Datenstrukturen, in: Scheer, A.-W. (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 63, Saarbrücken 1989.

Scheer 90a
Scheer, A.-W.: CIM (Computer Integrated Manufacturing) - Der computergesteuerte Industriebetrieb, 4. Auflage, Berlin-Heidelberg-New York-Tokio 1990.

Scheer 90b
Scheer, A.-W.: EDV-orientierte Betriebswirtschaftslehre - Grundlagen für ein effizientes Informationsmanagement, 4. Auflage, Berlin-Heidelberg-New York-Tokio 1990.

Scheer 90c
Scheer, A.-W.: Modellierung betriebswirtschaftlicher Informationssysteme (Teil 1: Logisches Informationsmodell), in: Scheer, A.-W. (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 67, Saarbrücken 1990.

Scheer 90d
Scheer, A.-W.: Datenmodelle: Wichtigster Baustein für die Gestaltung von Datenbanken, Anwendungssoftware und Integrationskonzepten, in: Scheer, A.-W. (Hrsg.), Datenbanken 1990 - Praxis Relationaler Datenbanken, Saarbrücken 1990, S. 137 - 154.

Scheer 90e
Scheer, A.-W.: Unternehmensdatenmodell, in: Information Management, 5 (1990), Heft 1, S. 92 - 94.

Scheer 90f
Scheer, A.-W.: Wirtschaftsinformatik - Informationssysteme im Industriebetrieb, 3. Auflage, Berlin-Heidelberg-New York-Tokio 1990.

Schek/Scholl 83
Schek, H.-J.; Scholl, M.: Die NF^2-Relationenalgebra zur einheitlichen Manipulation externer, konzeptueller und interner Datenstrukturen, in: Schmidt, J. W. (Hrsg.), Sprachen für Datenbanken, Berlin-Heidelberg-New York-Tokio 1983, S. 113 - 133, S. 113 - 133.

Schiefer/Rehm 89
Schiefer, B.; Rehm, S.: Eine Anfragesprache für ein strukturell-objektorientiertes Datenmodell, in: Härder, T. (Hrsg.), Datenbanksysteme in Büro,Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1989, S. 373 - 388.

Schlageter/Stucky 83
Schlageter, G.; Stucky, W.: Datenbanksysteme: Konzepte und Modelle, 2. Auflage, in: Leitfäden der angewandten Mathematik und Mechanik, Stuttgart 1983.

Schubert et al. 79
Schubert, L. K.; Goebel, R. G.; Cercone, N. J.: The Structure and Organisation of a Semantic Net for Comprehension and Inference, in: Findler, N. J. (Hrsg.), Associative Networks, New York-San Francisco 1979, S. 121 - 175.

Schulte 87
Schulte, U.: Praktikable Ansatzpunkte zur Realisierung von Datenmanagment-Konzepten, in: Information Management, 2 (1987), Heft 4, S. 26 - 31.

Schwarze 87
Schwarze, J.: Daten- und Datenbankorientierung in der Betriebswirtschaft, in: Angewandte Informatik, 29 (1987), S. 51 - 58.

Shoval/Even-Chaime 87
Shovel, P.; Even-Chaime, M.: ADDS: A system for automatic database schema design based on the binary-relationship model, in: Data & Knowlegde Engineering, 2 (1987), S. 123 - 144.

Sinz 87
Sinz, E. J.: Datenmodellierung betrieblicher Probleme und ihre Unterstützung durch ein wissensbasiertes Entwicklungssystem, Habilitationsschrift Universität Regensburg 1987.

Sinz 88
Sinz, E. J.: Das Strukturierte Entity-Relationship-Modell (SERM), in: Angewandte Informatik, 30 (1988), S. 191 - 202.

Sinz 90
Sinz, E. J.: Das Entity-Relationship-Modell und seine Erweiterungen, in: HMD - Theorie und Praxis der Wirtschaftsinformatik, 27 (1990), Heft 152, S. 17 - 29.

Smith/Smith 77a
Smith, J. M.; Smith, D. C.: Database Abstractions: Aggregation, in: Communications of the ACM, 2 (1977), S. 405 - 413.

Smith/Smith 77b
Smith, J. M.; Smith, D. C.: Database Abstractions: Aggregation and Generalization, in: ACM Transaction on Database Systems, 2 (1977), S. 105 - 433.

Snodgrass/Ahn 85
Snodgrass, R.; Ahn, I.: A Taxonomy of time in databases, in: ACM SIGMOD Record, 14 (1985), S. 236 - 246.

Spaccapietra et al. 90
Spaccapietra. S.; Parent, C.; Yetongnon, K.; Abaidi, M. S.: Generalizations: a formal and flexible approach, École Polytechnique Fédérale de Lausanne, Département d'Informatique, Lausanne 1990.

Spaccapietra/Parent 90
Spaccapietra. S.; Parent, C.: View Integration: A Step Forward in Solving Structural Conflicts, École Polytechnique Fédérale de Lausanne, Département d'Informatique, Lausanne 1990.

Springstell/Chuang 88
Springstell, F. N.; Chuang, P. J.: ERDDS: The intelligent E-R-based Database Design System, yielding Normal Forms under Extended Regularity, in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 211 - 230.

Steinbauer 83
Steinbauer, D.: Transaktionen als Grundlage zur Strukturierung und Integritätssicherung in Datenbank-Anwendungssystemen, Dissertation Universität Erlangen 1983.

Steinbauer/Wedekind 85
Steinbauer, D.; Wedekind, H.: Integritätsaspekte in Datenbanksystemen, in: Informatik Spektrum, 8 (1985), S. 60 - 68.

Stonebraker et al. 86
Stonebraker, M.; Hanson, E.; Hong, C.-H.: The Design of the Postgres Rule System, in: Stonebraker, M. (Hrsg.), Readings in Database Systems, San Mateo 1988, S. 556 - 565.

Stonebraker et al. 90
Stonebraker, M.; Rowe, L. A.; Hirohama, M.: The Implementation of Postgres, in: IEEE Transaction on Knowledge and Data Engineering, 2 (1990), No. 1, S. 125 - 142.

Studer 88
Studer, R.: A Conceptual Model for Physical and Logical Time, in: March, S. T. (Hrsg.), Entity-Relationship Approach (Proceedings of the Sixth International Conference on Entity-Relationship Approach, New York 1987), Amsterdam-New York-Oxford-Tokyo 1988, S. 223 - 236.

Su 85
Su, S. Y. W.: Modeling Integrated Manufacturing Data Using SAM*, in: Blaser, A.; Pistor, P. (Hrsg.), Datenbank-Systeme für Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1985, S. 27 - 49.

Su/Lo 80
Su, S. Y. W.; Lo, D. H.: A Semantic Association Model for Conceptual Database Design, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to System Analysis and Design (Proceedings of the International Conference On Entity-Relationship Approach to System Analysis and Design, Los Angeles 1979), Amsterdam-New York-Oxford 1980, S. 169 - 192.

Tabourier 83
Tabourier, Y.: Further Development of the Occurrence Stucture Concept: The EROS Approach, in: Davis C. G. et al. (Hrsg.), Entity-Relationship Approach to Software Engineering (Proceedings of the Third International Conference on Entity-Relationship Approach, Anaheim 1983), Amsterdam-New York-Oxford 1983, S. 565 - 583.

Tabourier/Nanci 83
Tabourier, Y.; Nanci, D.: The Occurrence Structure Concept: An Approach to Structural Integrity Constraints in the Entity-Relationship Model, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 73 - 108. er 2.ER 1981

Tasker 88
Tasker, D.: An Entity relationship View of Time, in: March, S. T. (Hrsg.), Entity-Relationship Approach (Proceedings of the Sixth International Conference on Entity-Relationship Approach, New York 1987), Amsterdam-New York-Oxford-Tokyo 1988, S. 237 - 247.

Teorey et al. 86
Teorey, T. J.; Yang, D.; Fry, J. P.: A Logical Design Methodology for Relational Databases Using The Extended Entity-Relationship Model, in: ACM Computing Surveys, 18 (1986), S. 197 - 222.

Teorey et al. 89
Teorey, T. J.; Wei, G.; Bolton, D. B.; Koenig, J. A.: ER Model Clustering as an Aid for User Communication and Documentation in Database Design, in: Communications of the ACM, 32 (1989), S. 975 - 987.

Thurnherr 80
Thurnherr, B.: Konzepte und Sprachen für den Entwurf konsistenter Datenbanken, Dissertation ETH Zürich 1980.

Twine 89
Twine, S.: Mapping between a NIAM conceptual schema and KEE frames, in: Data & Knowledge Engineering, 4 (1989), S. 125 - 155.

Verheijen/VanBekkum 82
Verheijen, G. M. A.; van Bekkum, J.: NIAM: An Information Analysis Method, in: Olle, T. W. et al. (Hrsg.), Information Systems Design Methodologies - A Comparative Review, Amsterdam-New York-Oxford 1982, S. 537 - 589.

Vetter 89
Vetter, M.: Aufbau betrieblicher Informationssysteme mittels konzeptioneller Datenmodellierung, 6. Auflage, Stuttgart 1989.

Vinek et al. 82
Vinek, G.; Rennert, P. F.; Tjoa, A. M.: Datenmodellierung - Theorie und Praxis des Datenbankentwurfs, Würzburg-Wien 1982.

Wagner 88
Wagner, C. F.: Implementing Abstraction Hierarchies, in: Batini, C. (Hrsg.), Proceedings of the Seventh International Conference on Entity-Relationship Approach, Rom 1988, S. 267 - 282.

Webre 83
Webre, N.: An Extended Entity-Relationship Model and Its Use on a Defense Project, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on Entity-Relationship Approach, Washington D. C. 1981), Amsterdam-New York-Oxford 1983, S. 173 - 193.

Wedekind 85
Wedekind, H.: Die Komposition beim Datenbank-Schemaentwurf als Kennzeichung, in: Angewandte Informatik, 27 (1985), S. 420 - 423.

Wedekind 86
Wedekind, H.: Integrierte Fertigungsdatenbanken, in: Hommel, G.; Schindler, S. (Hrsg.), Informatik-Anwendungen - Trends und Perspektiven, Berlin-Heidelberg-New York-Tokio 1986, S. 90 - 108.

Wedekind 87
Wedekind, H.: Fertigungsdatenbanken, in: Informatik Spektrum, 10 (1987), S. 40 - 41.

Wedekind 88
Wedekind, H.: Nullwerte in Datenbanksystemen, in: Informatik Spektrum, 11 (1988), S. 97 - 98.

Wedekind/Zörntlein 87
Wedekind, H.; Zörntlein, G.: Eine konzeptionelle Basis für den Einsatz von Datenbanken in Flexiblen Fertigungssystemen, in: Informatik Spektrum, 10 (1987), S 83 - 96.

Weikum et al. 87
Weikum, G.; Neumann, B.; Paul, H.-B.: Konzeption und Realisierung einer mengenorientierten Seitenschnittstelle zum effizienten Zugriff auf Komplexe Objekte, in: Schek, H.-J.; Schlageter, G. (Hrsg.), Datenbank-Systeme in Büro, Technik und Wissenschaft, Berlin-Heidelberg-New York-Tokio 1987, S. 212 - 230.

Wiendahl 87
Wiendahl, H.-P.: Belastungsorientierte Fertigungssteuerung, München 1987.

Wigand 88
Wigand, R. T.: Fünf Grundsätze für die erfolgreiche Einführung des Informations-Management, in: Information Management, 3 (1988), Heft 2, S. 24 - 30.

Wildemann 84
Wildemann, H. (Hrsg.): Dialogorientierte Produktionsplanung und -steuerung mit On-Line Betriebsdatenerfassung, fmt-Report 6, München 1984.

Wilkes 89
Wilkes, W.: Versionsunterstützung in Datenbanken, in: Informatik Spektrum, 12 (1989), S. 166 - 168.

Wong/Katz 80
Wong, E.; Katz, R. H.: Logical Design and Schema Conversion for Relational and DBTG Databases, in: Chen, P. P. (Hrsg.), Entity-Relationship Approach to System Analysis and Design (Proceedings of the International Conference On Entity-Relationship Approach to System Analysis and Design, Los Angeles 1979), Amsterdam-New York-Oxford 1980, S. 311 - 321.

Yang 88
Yang, Y. K.: An Enhanced Data Model for CAD/CAM Database Systems, in: 25th ACM/IEEE Design Automation Conference Proceedings (June 1988), Anaheim 1988, S. 263 - 268.

Zehnder 87
Zehnder, C. A.: Informationssysteme und Datenbanken, Stuttgart 1987.

Zörntlein 88
Zörntlein, G.: Flexible Fertigungssysteme: Belegung, Steuerung und Datenorganisation, München-Wien 1988.

Register