Prof. Dr.-Ing. Peter Steininger

Datenbankmanagement

Pub.-Nr. 00012026DBM001

Materialsammlung zur Vorlesung

März 2026

Nachdruck auch auszugsweise verboten
Alle Rechte vorbehalten

© Dieses Skriptum unterliegt dem Copyright von Prof. Dr.-Ing. Peter Steininger, anderen Autoren und Verlagen (s. Literaturverzeichnis, Hinweise im Text usw.). Das Skriptum darf nur von Studierenden der FOM Hochschule für Oekonomie und Management gGmbH für das Modul "Datenbankmanagement" (DBM) für Studienzwecke verwendet werden.

Die Weitergabe an Dritte ist nicht zulässig. Insbesondere das Veröffentlichen im Internet ist verboten. Wenn Dritte zu diesem Skriptum Zugang haben wollen, bedarf dies der vorherigen schriftlichen Genehmigung durch Prof. Dr.-Ing. Peter Steininger (peter.steininger@fom.de).

Gliederung

	Kapitel	Veranstaltung	Seite
0	Literaturverzeichnis	DBM-00	III
	0.1 Primärliteratur	DBM-00	III
	0.2 Sekundärliteratur	DBM-00	III
1	Grundbegriff	DBM-02	1
	1.1 Beispieldatenbank	DBM-02	1
	1.2 Das Datenbanksystem-Modell	DBM-02	1
2	Das Relationale Datenmodell	DBM-02	2
	2.1 Entwicklung des Modells	DBM-02	2
	2.2 Mathematische Grundlage	DBM-02	2
	2.2.1 Beispiel 1	DBM-02	4
	2.2.2 Beispiel 2	DBM-02	4
	2.3 Definitionen und Folgerungen	DBM-02	5
	2.4 Relationenschema	DBM-02	6
3	SQL-Anweisung	DBM-02	6
	3.1 Entwicklungsgeschichte von SQL	DBM-02	6
	3.2 Datentypen	DBM-02	8
	3.3 Operatoren	DBM-02	10
	3.4 SELECT-Anweisung	DBM-02	12
	3.4.1 DISTINCT-Klausel	DBM-02	13
	3.4.2 Spaltennamen	DBM-02	14
	3.4.3 Sortierung	DBM-02	14
	3.5 Restriktion	DBM-02	15
	3.6 Berechnete Spalten	DBM-02	17
	3.7 Verarbeitungsreihenfolge von SQL-Anweisungen	DBM-02	17
	3.8 Unterabfragen	DBM-02	18
	3.9 SQL integrierte Standardfunktionen	DBM-02	21
	3.10 Gruppierung	DBM-02	23
	3.11 Verbünde (Joins)	DBM-02	24
	3.12 Inner-/Outer-Join	DBM-02	29
	3.13 Mengenoperationen	DBM-02	32
4	Data Manipulation Language	DBM-02	33
	4.1 INSERT-Anweisung	DBM-02	33
	4.2 DELETE-Anweisung	DBM-02	35
	4.3 Tabelle löschen	DBM-02	36
	4.4 UPDATE-Anweisung	DBM-02	37
5	Sichten	DBM-02	38

6	Data Definition Language	DBM-02	42
6.1	Tabelle erstellen	DBM-02	42
6.1.1	Integritätsbedingungen	DBM-02	43
6.1.1.1	Primärschlüssel	DBM-02	44
6.1.1.2	Eindutige Spalten	DBM-02	45
6.1.1.3	Check-Bedingung	DBM-02	45
6.1.1.4	Default-Bedingung	DBM-02	46
6.1.2	Referentielle Integrität	DBM-02	46
7	Einführung	DBM-90	49
7.1	Grundbegriffe	DBM-90	49
7.2	Datenspeicherung	DBM-90	55
8	Datenmodellierung	DBM-90	56
8.1	Grundlagen	DBM-90	56
8.2	Phasen	DBM-90	58
8.3	Basismethoden	DBM-90	63
8.4	Spezialmethoden	DBM-90	66
9	Entity-Relationship-Modell	DBM-90	70
9.1	Grundlagen	DBM-90	70
9.2	Entitäten	DBM-90	73
9.3	Attribute	DBM-90	75
9.4	Domänen	DBM-90	77
9.5	Beziehungen	DBM-90	77
9.6	Kardinalitäten	DBM-90	79
9.7	Spezielles	DBM-90	81
10	Relationen Modell	DBM-90	82
10.1	Grundlagen	DBM-90	82
10.2	Normalisierung	DBM-90	88
11	Datenbank-Systeme (DBS)	DBM-91	97
11.1	Einführung	DBM-91	97
11.1.1	Grundbegriffe	DBM-91	97
11.1.2	Verteilung und Speicherung	DBM-91	102
11.1.3	Historie	DBM-91	105
11.1.4	Datenbankmodelle	DBM-91	106
11.1.5	Rechte und Sicherheit	DBM-91	109
11.2	Relationenmodell	DBM-91	111
11.2.1	Locking und Locks	DBM-91	111
11.2.2	Datenorganisation	DBM-91	113
11.3	Online Transaction Processing (OLTP), Data Warehouse (DWH) und Data Mining	DBM-91	122
11.3.1	Management Information System (MIS)	DBM-91	122
11.3.2	Data Warehouse (DWH)	DBM-91	125
11.3.3	Online Analytical Processing (OLAP) und Online Transaction Processing (OLTP)	DBM-91	129

0 Literaturverzeichnis

0.1 Primärliteratur

ELMASRI, Ramez; NAVATHE, Shamkant B.:
Grundlagen von Datenbanksystemen: Bachelorausgabe.
München: Pearson Studium, 3. Auflage 2009.

VOSSEN, Gottfried:
Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme
München: Oldenbourg Verlag, 5. Auflage 2008.

0.2 Sekundärliteratur

BEAULIEU, Alan; HEYMANN-REDER, Dorothea; SCHULTEN, Lars:
Einführung in SQL
Köln: O'Reilly, 2. Auflage 2009.

HEINRICH, Lutz; LEHNER, Franz:
Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur
München: Oldenbourg Wissenschaftsverlag, 8. Auflage 2005.

JAROSCH, Helmut:
Grundkurs Datenbankentwurf: Eine beispielorientierte Einführung für Studenten und Praktiker
Wiesbaden: Vieweg+Teubner Verlag, 3. Auflage 2009.

KLUG, Uwe:
SQL: Der Einstieg in die deklarative Programmierung
Bochum: W3l, 2006.

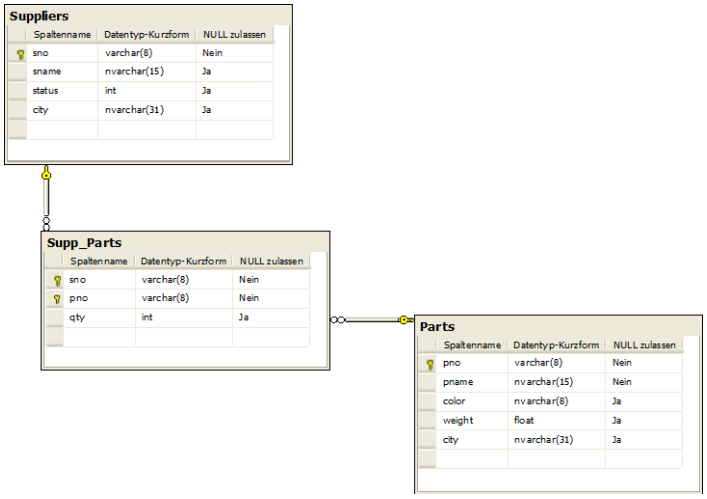
SCHICKER, Edwin:
Datenbanken und SQL: Eine praxisorientierte Einführung mit Hinweisen zu Oracle und MS-Access.
Stuttgart: B. G. Teubner Verlag, 3. Auflage 2000.

1 Grundbegriff

1.1 Beispieldatenbank

Datenbankmanagement

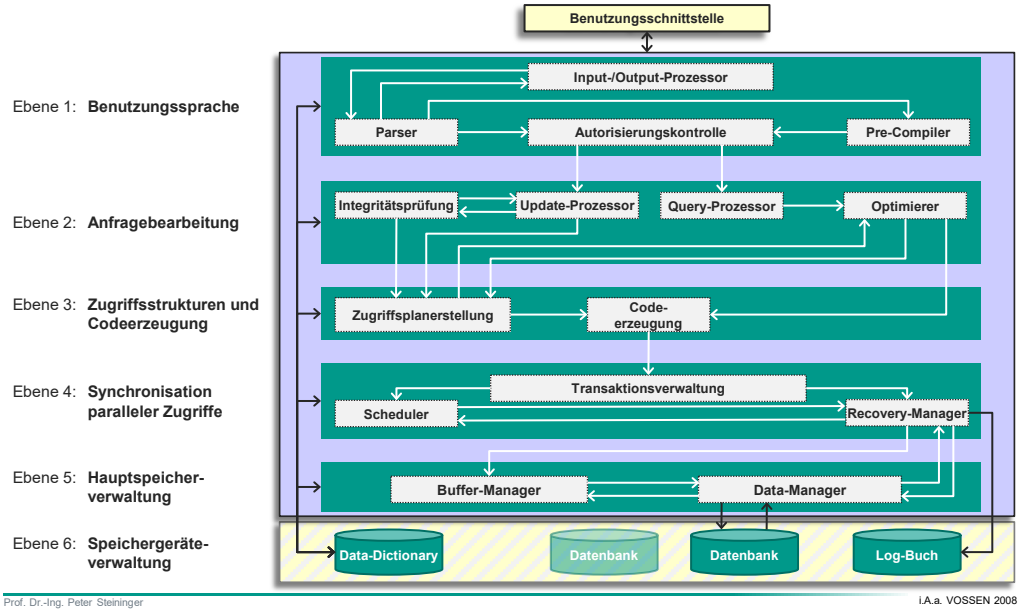
Beispieldatenbank – ER-Modell



Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-01933

1.2 Das Datenbanksystem-Modell

Datenbanksystem (DBS) – Blockdiagramm



Prof. Dr.-Ing. Peter Steininger
2026-SS
i.A.a. VOSSEN 2008
DBM-0001-2014-01035

2 Das Relationale Datenmodell

2.1 Entwicklung des Modells

Datenbankmanagement

Das relationale Datenmodell I/IV

CODD, Edgar F. (Ted) 1970:
"A Relational model of Data for Large Shared Data Banks"

Information Retrieval — F. RAMSHAW, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not the solution. Activity of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inferential, full relational models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed briefly. In Section 3, problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchy of data, networks of data, relations, normal form, relational, view, prompting, purpose, retrieval, composition, join, extract, restrict, projection, inferential, data integrity

CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. INTRODUCTION

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not the solution. Activity of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inferential, full relational models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed briefly. In Section 3, problems of redundancy and consistency in the user's model.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, no superimposed ordering nor additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which yields maximum independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed briefly in Section 3. The network model, on the other hand, has spawned a number of mechanisms, not the least of which is mistaking the representation for the data itself, that are ad hoc in character (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limits of present and proposed data systems, and also this relation notion (from a logical standpoint) of comparing representations of data within a single system. Examples of this kind of comparison are given in various parts of this paper. Some implications of the relational view for future hardware are noted.

1.2. FORM, INFORMATION IN PRESENT FORMS

The provision of data description tables in recently delivered data management systems is a step in the direction toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in the machine. However, many changes in data representation characteristics which can be changed without logically impacting some application programs is still quite limited. Further, the model of data with which some interest is still shackled with representational properties which ought to be in the hardware/software/firmware layers of data (as opposed to individual lines). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.3. ORDERING, INDEXING, ACCESS PATHS

data bank may be stored in a variety of ways, some involving no ordering for ordering, some permitting each element to participate in one ordering, others permitting each element to participate in several orderings. Let us consider those ordering systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses in storage. For instance, the ordering of parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the ordering of elements is preserved. Suppose such a file is identified by for it a subscribing of the

Volume 13 / Number 6 / June, 1970 — Communications of the ACM 377

Communications of the ACM
Volume 13, Issue 6 (June 1970)
Pages: 377 - 387
Year of Publication: 1970
ISSN:0001-0782
http://portal.acm.org/citation.cfm?doid=362384.362685

- Datensicht als zweidimensionale Tabelle
- Basierend auf der mathematischen Theorie der Relationen und Prädikatenlogik 1. Ordnung
- Anwendung von systematischen Methoden

Aufbauende Verfahren

- Normalisierungstheorie
- Relationale Abfragesprache

2.2 Mathematische Grundlage

Das relationale Datenmodell II/IV

Definition:

- Gegeben sind nicht notwendigerweise disjunkte Mengen (Domänen) $D_1, D_2, \ldots, D_n$
- R wird als Relation auf diesen Mengen bezeichnet, wenn R eine Menge von geordneten n-Tupeln ist, sodass $d_1 \in D_1$, $d_2 \in D_2$ und $d_n \in D_n$ gehört
- D_i wird als Wertebereich von R bezeichnet
- n wird als Grad von R bezeichnet
- Darstellung der Relation erfolgt in Form von Spalten und Zeilen (Tabelle)
  - Spalten werden als Attribute bezeichnet
  - Zeilen bilden die n-Tupel

Das relationale Datenmodell III/IV

Definition:

- Relationenschema $R(A_1, A_2, ..., A_n)$:
  - Relationenname R und
  - Attributliste
- $R.A_i$:
  - Rollenname (Attribut) A eines bestimmten Wertebereichs D im relationalen Schema R
- Funktion 'dom' ordnet Attribut A_i Wertebereich D_i zu (D_i=dom(A_i))
- Grad:
  - Anzahl der Attribute (n)
- relationales Schema beschreibt Relation
  - Beispiel:
    - Book(ISBN, BuchTitel, Edition)

Das relationale Datenmodell IV/IV

Definition:

- Relation r(R) eines Relationenschemas $R(A_1, A_2, ..., A_n)$:
- Menge von n-Tupeln:
  - $r = \{t_1, t_2, ..., t_m\}$
    - Kardinalität: Anzahl der Tupel (m)
- n-Tupel:
  - $t = (v_1, v_2, ..., v_n)$: geordnete Liste von n Werten
- Forderung:
  $v_i \in D_i$ = dom(A_i) oder spezieller Null-Wert
- $t[A_i]$ oder $t.A_i$: i-ter Wert im Tupel t
- t[X]: Wertekombination bzgl. Attributmenge X des Tupels t
- $r(R) \subseteq dom(A_1) \times dom(A_2) \times ... \times dom(A_n)$ (mathematische Relation)

2.2.1 Beispiel 1

Das relationale Datenmodell – Beispiel 1

A_1 = ISBN	D_1 = ISBN	= {1234, 2234, ...}	$D_1 = dom(A_1)$
A_2 = BuchTitel	D_2 = Titel	= {"DBMS", "ORACLE", ...}	$D_2 = dom(A_2)$
A_3 = Edition	D_3 = Edition_Number	= {1,2,3,4, ...}	$D_3 = dom(A_3)$

Relationenschema:

Book(ISBN:ISBN, BuchTitel:Titel, Edition:Edition_Number)
Grad = 3
Book.ISBN, Book.BTitel, Book.Edition sind Attribute von Book

Relation:

$r(Book) = \{t_1, t_2, ..., t_m\}$

t_1 = (1234, "DBMS", 7)
$v_1 = 1234 \in$ ISBN; v_2 = "DBMS" $\in$ Titel; $v_3 = 7 \in$ Edition_Number

t_2 = (2345, "ORACLE",2)
t_2[ISBN] = 2345; t_2[BuchTitel] = "ORACLE"; t_2[Edition] = 2

2.2.2 Beispiel 2

Das relationale Datenmodell – Beispiel 2

$t_i \in dom(A_1) \times dom(A_2) \times dom(A_3)$

$t_1 \in D_1 \times D_2 \times D_3$

$t_1 \in$ ISBN × Titel × Edition_Number

$r(Book)=\{t_1, ...,t_m\} \subseteq$ ISBN × Titel × Edition_Number

2.3 Definitionen und Folgerungen

Das relationale Datenmodell III/III

Definition:

- Jedes Attribut besitzt einen eindeutigen Namen.
- Jedes Attribut wird durch seinen Namen referenziert.
- Attribute unterliegen keiner Ordnung.
- Attribute in einer Tabelle müssen atomar sein.
- Jede Zeile in einer Tabelle ist eindeutig.
- Die Kardinalität ist die Anzahl der Zeilen zu einem Zeitpunkt.

Modellierung von Beziehungen

Folgerungen:

- Wegen der Eindeutigkeit einer Zeile existiert immer ein Schlüssel
- Primärschlüssel dienen zur Modellierung von Beziehungen zwischen Relationen
  - Primärschlüssel
  - Fremdschlüssel
  - Zusammengesetzter Primärschlüssel
  - Zusammengesetzter Fremdschlüssel
  - Schlüsselkandidat

2.4 Relationenschema

Relationenschema

Relationenschema: FLUGGAST { Nation, PassNr, VName, NName, Titel }

- Relation: FLUGGAST
- Attribute:
  - Nation
  - PassNr
  - Vname
  - Nname
  - Titel

- Primärschlüssel:
  - Nation
  - PassNr

FLUGGAST

Nation	PassNr	VName	NName	Titel
Schweiz	1234	Peter	Mair	Dipl.-Ing.
Austria	3198	Arnold	Egger	
Deutschland	4545	Hugo	Faber	
Deutschland	8713	Emil	Gartner	Dr.

3 SQL-Anweisung

3.1 Entwicklungsgeschichte von SQL

Structured Query Language (SQL)

- **IBM SEQUEL (Structured English Query Language) 1971**
  - Prototyp System R

- **Relational Software Inc.**
  - 1979, kommerzielles System: Oracle V2

- **IBM SQL/DS 1981**

- **ISO-SQL1 (1986, ANSI), (1987, ISO)**
  - Addendum (1989)

- **ISO-SQL2 (1992)**
- **ISO-SQL3 (1999)**

- **ISO-SQL:2003 (2003)**
- **ISO-SQL:2006 (2006)**

SQL-Implementierungsmethoden

- **Direktes SQL**
  - SQL-Anweisungen interaktiv
- **Modulsprache**
  - SQL-Anweisungen in Programmen in getrennter Form
- **Call Level Interface (CLI)**
  - SQL-Anweisungen eingebettet durch SQL-Implementierung mittels einer Bibliothek für Programmiersprachen
- **Embedded SQL**
  - SQL-Anweisungen innerhalb der Programmiersprache

Structured Query Language 2 - SQL92

- **Sprachbestandteile**
  - Befehle zur Definition des Datenbankschemas
    - Data Definition Language (DDL)
  - Befehle zur Datenmanipulation = Projektion und Abfrage sowie Ändern, Einfügen, Löschen
    - Data Manipulation Language (DML)
  - Befehle für die Rechteverwaltung
    - Data Control Language (DCL)

Structured Query Language 2 - SQL92

- **SQL92**
  - Erweiterte Datentypen
  - Auf-/Abbau von Client-/Server-Verbindung
  - Dynamisches SQL
  - Cursor (scrollable)
  - Outer-Joins
  - Intersection-, Minus-Operator

3.2 Datentypen

SQL92 – Datentypen – numerische Datentypen I/II

- integer (integer4)
  - 4 Byte Zahl
- integer (integer2)
  - 2 Byte Zahl
- numeric (p, q)
  - Dezimalzahl mit genau p Stellen, davon q hinter dem Dezimalzeichen
- decimal (p, q)
  - Dezimalzahl mit mindestens p Stellen, davon q hinter dem Dezimalzeichen
- real
  - Einfach genaue Gleitpunktzahl (4 Byte)

SQL92 – Datentypen – numerische Datentypen II/II

- float (p)
  - Gleitpunktzahl, mindestens p Stellen genau
- double
  - Doppelt genaue Gleitpunktzahl (8 Byte)

SQL92 – Datentypen – Zeichen

- char{acter} (n)
  - Zeichenkette mit genau n Zeichen
  - Speicherbedarf: n Byte
- char{acter} varying (n)/varchar (n)
  - Zeichenkette mit höchstens n Zeichen
  - Speicherbedarf: Länge der Zeichenkette

SQL92 – Datentypen – Erweiterte Datentypen: Zeit, Datum usw.

- bit (n)
- bit varying (n)
- date
- time
- timestamp
- datetime
- blob
- clob
- stream

3.3 Operatoren

SQL92 – Operatoren

- Zahlen
  - +, -, *, /, %
- Zeichenketten
- Datum-/Zeitangaben
  - +, -
- Boolesche Verknüpfung
  - and, or, not
- Vergleiche
  - =, >, <, >=, <=, <>
- Typkonversion

SQL92 – Datentypenkonvertierung

Possible Implicit PL/SQL Data Type Conversions:

From:	To:									
	BLOB	**CHAR**	**CLOB**	**DATE**	**LONG**	**NUMBER**	**PLS_INTEGER**	**RAW**	**UROWID**	**VARCHAR2**
BLOB								Yes		
CHAR			Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CLOB		Yes								Yes
DATE		Yes			Yes					Yes
LONG		Yes						Yes		Yes
NUMBER		Yes			Yes		Yes			Yes
PLS_INTEGER		Yes			Yes	Yes				Yes
RAW	Yes	Yes			Yes					Yes
UROWID		Yes								Yes
VARCHAR2		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/datatypes.htm

SQL92 – Datentypenkonvertierung

From: \ To:	binary	varbinary	char	varchar	nchar	nvarchar	datetime	smalldatetime	decimal	numeric	float	real	bigint	int(INT4)	smallint(INT2)	tinyint(INT1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	ntext	text	sql_variant
binary	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	○	○	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	○	○	◉
varbinary	◉		◉	◉	◉	◉	◉	◉	◉	◉	○	○	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	○	○	◉
char	●	●		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	●	●	◉	●	◉	◉	◉	◉	◉
varchar	●	●	◉		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	●	●	◉	●	◉	◉	◉	◉	◉
nchar	●	●	◉	◉		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	●	●	◉	●	◉	○	◉	◉	◉
nvarchar	●	●	◉	◉	◉		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	●	●	◉	●	◉	○	◉	◉	◉
datetime	●	●	◉	◉	◉	◉		◉	●	●	●	●	●	●	●	●	●	●	●	●	◉	○	○	○	◉
smalldatetime	●	●	◉	◉	◉	◉	◉		●	●	●	●	●	●	●	●	●	●	●	●	◉	○	○	○	◉
decimal	◉	◉	◉	◉	◉	◉	◉	◉	*	*	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	○	○	○	◉
numeric	◉	◉	◉	◉	◉	◉	◉	◉	*	*	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	○	○	○	◉
float	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	◉	◉	◉	◉	◉	○	○	○	○	◉
real	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	◉	◉	◉	◉	○	○	○	○	◉
bigint	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	◉	◉	◉	◉	○	○	○	◉
int(INT4)	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	◉	◉	◉	○	○	○	◉
smallint(INT2)	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	◉	◉	○	○	○	◉
tinyint(INT1)	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	◉	○	○	○	◉
money	◉	◉	●	●	●	●	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	◉	○	○	○	◉
smallmoney	◉	◉	●	●	●	●	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	◉	○	○	○	◉
bit	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉		◉	◉	○	○	○	◉
timestamp	◉	◉	◉	◉	○	○	◉	◉	◉	◉	○	○	◉	◉	◉	◉	◉	◉	◉	◉	○	◉	○	○	○
uniqueidentifier	◉	◉	◉	◉	◉	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○	○	○	◉
image	◉	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○		○	○	○
ntext	○	○	●	●	◉	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			◉	○
text	○	○	◉	◉	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉			○
sql_variant	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○	●	○	○	○	

● Explicit conversion
◉ Implicit conversion
○ Conversion not allowed
* Requires explicit CAST to prevent the loss of precision or scale that might occur in an implicit conversion

http://msdn.microsoft.com/en-us/library/aa226054(SQL.80).aspx

3.4 SELECT-Anweisung

SQL92 – Abfragen

SELECT-Anweisung

```
SELECT          < expression >

FROM            < table-name >

     [WHERE     < search-condition >]

                [GROUP BY …
                           [HAVING …] ]

[ORDER BY …]
```


SQL92 – Selektion

Welche Teile mit welchen Merkmalen gibt es?

```
SELECT *
FROM   parts;
```

Resultat:

```
pno  pname color  weight  city
---- ----- ------ ------- -------
P1   Nut   red    12.0    London
P2   Bolt  green  17.0    Paris
P3   Screw blue   17.0    Rome
P4   Screw red    14.0    London
P5   Cam   blue   12.0    Paris
P6   Cog   red    19.0    London
(6 row(s) affected)
```

Bessere SQL-Alternative:

```
SELECT pno, pname, color, weight, city
FROM parts;
```

SQL92 – Projektion

Welche Farbe haben die Teile?

```
SELECT color
FROM   parts;
```

Resultat:

```
color
--------
red
green
blue
red
blue
red
(6 row(s) affected)
```


3.4.1 DISTINCT-Klausel

SQL92 – Projektion: DISTINCT

Welche Farbe haben die Teile überhaupt (keine Mehrfachnennung)?

```
SELECT DISTINCT color
FROM   parts;
```

Resultat:

```
color
--------
blue
green
red
(3 row(s) affected)
```

3.4.2 Spaltennamen

SQL92 – Spaltennamen

Veränderung der Attributnamen

```
SELECT          pno AS part_no,
                pname AS part_name,
                color AS part_color
FROM   parts;
```

Resultat:

```
part_no  part_name part_color
-------- --------- ----------
P1       Nut       red
P2       Bolt      green
P3       Screw     blue
P4       Screw     red
P5       Cam       blue
P6       Cog       red
(6 row(s) affected)
```


3.4.3 Sortierung

SQL92 – Sortierung

Alle Teile, absteigend nach Farbe, falls gleich, nach Name aufsteigend sortiert:

```
SELECT pno, color
FROM   parts
ORDER BY color DESC, pno ASC;
```

Resultat:

```
pno  color
---- --------
P1   red
P4   red
P6   red
P2   green
P3   blue
P5   blue
(6 row(s) affected)
```

3.5 Restriktion

SQL92 – Restriktion

Welche Teile sind in Paris gelagert?

```
SELECT pname, color
FROM   parts
WHERE city = 'Paris';
```

Resultat:

```
pname  color
------ --------
Bolt   green
Cam    blue
(2 row(s) affected)
```


SQL92 – Intervallangaben

Welche roten Teile wiegen zwischen 13 und 20 kg?

```
SELECT pname, weight
FROM   parts
WHERE color = 'red' AND weight BETWEEN 13 AND 20;
```

Resultat:

```
pname weight
----- -------
Screw 14.0
Cog   19.0
(2 row(s) affected)
```

SQL92 – Intervallangaben

Welche roten Teile wiegen zwischen 13 und 20 kg?

```
SELECT pname, weight
FROM parts
WHERE color = 'red' AND weight >= 13 AND weight <= 20;
```

Resultat:

```
pname weight
----- -------
Screw 14.0
Cog   19.0
(2 row(s) affected)
```


SQL92 – Patternmatching

Welche Teilenamen haben wenigstens 2 Buchstaben und fangen mit 'C' an?

```
SELECT pno, pname
FROM parts
WHERE pname LIKE 'C_%';
```

Resultat:

```
pno      pname
-------- ------
P5       Cam
P6       Cog
(2 row(s) affected)
```

3.6 Berechnete Spalten

SQL92 – berechnete Spalten I/II

Welche Teile wiegen mehr als 30 Pfund (1 lb = 0,4536 kb)?

```
SELECT pno,
       weight / 0.4536 AS wgt,
       'lbs' AS unit
FROM parts
WHERE weight / 0.4536 > 30
ORDER BY weight;
```

Resultat:

```
pno      wgt   unit
-------- ----- ----
P4       30.86 lbs
P2       37.47 lbs
P3       37.47 lbs
P6       41.88 lbs
(4 row(s) affected)
```


3.7 Verarbeitungsreihenfolge von SQL-Anweisungen

SQL92 – berechnete Spalten II/II

Warum zwei Berechnungen in der Abfrage?

Verarbeitungsregel für SQL:

- Kreuzprodukt über alle Tabellen, die FROM benennt
- Restriktion:
  Zeilen bestimmen, welche die WHERE-Bedingung erfüllen
- Gruppenbildung
- Gruppen-Restriktion
- Mengenbildung
- Sortieren
- Projektion:
  Spalten gemäß SELECT-Klausel auswählen

3.8 Unterabfragen

SQL92 – Unterabfragen I/II

Welche Teile liegen in der selben Stadt, in der S1 sitzt?

```
SELECT pno, pname
FROM parts
WHERE city = (
                          SELECT city
                          FROM suppliers
                          WHERE sno = 'S1'
                );
```

Resultat:

```
pno      pname
-------- ---------------
P1       Nut
P4       Screw
P6       Cog
(3 row(s) affected)
```

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-01971

SQL92 – Unterabfragen II/II

- **Unterabfragen liefern:**
  - genau eine Ergebniszeile
    - Ergebnis der Unterabfrage wird mit Attributen der Hauptabfrage verglichen
  - mehrere Ergebniszeilen
    - WHERE-Bedingung arbeitet mit Mengenoperationen
      - IN: prüft ob ein Wert in der Unterabfrage enthalten ist
      - EXISTS: prüft ob wenigstens eine Zeile eine Bedingung erfüllt
      - < Θ > ANY, < Θ > SOME: prüfen, ob irgendeine Zeile den Operator erfüllt
      - < Θ > ALL: prüft, ob alle Zeilen den Operator erfüllen
- **Θ-Operator: =, <, <=, >, >=, <>**

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-01972

SQL92 – Unterabfragen mit IN

Welche Teile sind in Städten gelagert, in denen auch Händler sitzen?

```
SELECT pno, pname, city
FROM parts
WHERE city IN (
                          SELECT city
                          FROM suppliers
               );
```

Resultat:

```
pno  pname  city
---- ------ -------
P1   Nut    London
P2   Bolt   Paris
P4   Screw  London
P5   Cam    Paris
P6   Cog    London
(5 row(s) affected)
```


SQL92 – Unterabfragen mit ALL

Welche Teile sind schwerer als das schwerste blaue Teil?

```
SELECT pno, pname, color
FROM parts
WHERE weight > ALL (
                          SELECT weight
                          FROM parts
                          WHERE color = 'blue'
                          );
```

Resultat:

```
pno  pname  color
---- ------ --------
P6   Cog    red
```

SQL92 – Unterabfragen mit ANY

Wer hat wie viel Schrauben (screw) verkauft?

```
SELECT sno, qty
FROM supp_parts
WHERE pno = ANY (
                        SELECT pno
                        FROM parts
                        WHERE pname = 'Screw'
                );
```

Resultat:

```
sno      qty
-------- -----------
S1       400
S1       200
S4       300
(3 row(s) affected)
```


SQL92 – Unterabfragen mit EXISTS I/II

Welche Händler sitzen in einer Stadt, die auch Teile hat?

```
SELECT          sname, city
FROM            suppliers
     WHERE EXISTS (
                        SELECT *
                        FROM parts
                        WHERE city = suppliers.city
                        );
```

Resultat:

```
sname  city
------ -------
Smith  London
Jones  Paris
Blake  Paris
Clark  London
(4 row(s) affected)
```

SQL92 – Unterabfragen mit EXISTS II/II

- **Unterabfrage referenziert Hauptabfrage mittels eines Attributes:**
  - city
- **Referenziertes Attribut ist sowohl in der Tabelle der Unterabfrage als auch in der der Hauptabfrage vorhanden und ist deshalb zu qualifizieren:**
  - suppliers.city

3.9 SQL integrierte Standardfunktionen

SQL92 – Funktionen

Skalare Funktion:

- Mathematische Funktionen
  - ABS, ACOS, ASIN, ATAN, ATN2, CEILING, COS, COT, EXP, FLOOR, LOG, LOG10, PI, POWER, RADIANS, ROUND, SIGN, SIN, SQRT, SQUARE, TAN
- Zeichenketten Funktionen
  - CHARINDEX/INSTR, LEFT, LEN, LOWER, LTRIM, RIGHT, RTRIM, SUBSTRING, UPPER
- Zeit- und Datumsfunktionen
  - CURRENT DATE, CURRENT TIME, CURRENT_TIMESTAMP, DAY, MONTH, SYSDATE, TRUNC, TO_DATE, YEAR

Aggregatfunktion

- AVG, MAX, MIN, SUM, STDEV, COUNT

SQL92 – Funktionen

Größte, kleinste und durchschnittliche Zahl von Teileverkäufen?

```
SELECT          MIN(qty) AS mini,
                MAX(qty) AS maxi,
                AVG(qty) AS schnitt
FROM  supp_parts;
```

Resultat:

```
mini  maxi  schnitt
----- ----- -------
100   400   258
(1 row(s) affected)
```


SQL92 – NULL-Werte und SELECT (I/II)

```
CREATE TABLE nullBsp (n INT);

INSERT INTO nullBsp (n) VALUES(NULL);
INSERT INTO nullBsp (n) VALUES(1);
INSERT INTO nullBsp (n) VALUES(2);

SELECT COUNT(*) AS tutti FROM nullBsp;

SELECT COUNT(n) AS really FROM nullBsp;

SELECT AVG(n) schnitt from nullBsp;

-- not allowed:
-- select avg(*) as stutti from nullBsp
-- use cast to obtain real-average:
SELECT avg (CAST(n AS real)) AS dschnitt FROM nullBsp;

DROP TABLE nullBsp
```

3.10 Gruppierung

SQL92 – Gruppierung

GROUP BY verarbeitet Aggregatfunktionen auf Teilmengen von Zeilen
Wie viele Teile wurden insgesamt durch einen Händler verkauft?

```
SELECT          sno, SUM(qty) as total
FROM  supp_parts
      GROUP BY    sno;
```

Resultat:

```
sno      total
-------- -----------
S1       1300
S2       700
S3       200
S4       900
(4 row(s) affected)
```

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-01981

SQL92 – Gruppierung – HAVING

HAVING schränkt die Verarbeitung der Aggregatfunktionen ein
Wie viele Teile wurden insgesamt durch einen Händler verkauft?
Aber keinen Händler ausgeben, der nur einmal verkauft hat.

```
SELECT          sno, SUM(qty) as total
FROM supp_parts
      GROUP BY sno
                HAVING count(qty) > 1;
```

Resultat:

```
sno      total
-------- -----------
S1       1300
S2       700
S4       900
(3 row(s) affected)
```

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-01982

SQL92 – Gruppierung – HAVING

Wie viele Teile wurden insgesamt durch einen Händler verkauft?
Aber keinen Händler ausgeben, der nur einmal verkauft hat und keine Verkäufe kleiner als 200.

```
SELECT          sno, SUM(qty) as total
FROM  supp_parts
      WHERE qty >= 200
                GROUP BY sno
                          HAVING count(qty) > 1;
```

Resultat:

```
sno      total
-------- -----------
S1       1100
S2       700
S4       900
(3 row(s) affected)
```


3.11 Verbünde (Joins)

SQL92 – Verbünde (Joins)

JOIN verbindet zwei Tabellen unter Angabe der Schlüsselbeziehung (PK-FK)

```
SELECT          s.sname, sp.pno
FROM  supp_parts sp
                JOIN       suppliers s
                ON         s.sno = sp.sno
WHERE sp.pno = 'P1';
```

Resultat:

```
sname  pno
------ --------
Smith  P1
Jones  P1
(2 row(s) affected)
```

SQL92 – Verbünde (Joins) - Syntax

SQL:

```
SELECT          s.sname, sp.pno
FROM            supp_parts sp, suppliers AS s
     WHERE      s.sno = sp.sno AND sp.pno = 'P1';
```

SQL92:

```
SELECT          s.sname, sp.pno
FROM            supp_parts sp
                JOIN       suppliers s
                ON         s.sno = sp.sno
WHERE sp.pno = 'P1';
```


SQL92 – Join-Berechnung I/IV

Tabelle: Suppliers AS s

sno	sname	status	city	rowID
S1	Smith	20	London	1
S2	Jones	10	Paris	2
S3	Blake	30	Paris	3
S4	Clark	20	London	4

Tabelle: Supp_Parts AS sp

sno	pno	qty	rowID
S1	P1	300	1
S1	P2	200	2
S1	P3	400	3
S1	P4	200	4
S1	P5	100	5
S1	P6	100	6
S2	P1	300	7
S2	P2	400	8
S3	P2	200	9
S4	P2	200	10
S4	P2	200	11
S4	P5	400	12

SQL92 – Join-Berechnung II/IV

SQL:

```
SELECT          s.sno AS [s.sno], sp.sno AS[sp.sno]
FROM  supp_parts AS sp, suppliers AS s;
```

s.rowID	sp.rowID	s.sno	sp.sno
1	1	S1	S1
…	…	…	…
1	9	S1	S3
…	…	…	…
2	3	S2	S1
…	…	…	…
4	11	S4	S4
4	12	S4	S4

SQL92 – Join-Berechnung III/IV

SQL:

```
SELECT          s.sname, sp.pno
FROM  supp_parts sp
                JOIN       suppliers s
                ON         s.sno = sp.sno
WHERE sp.pno = 'P1'
```

s.rowID	sp.rowID	s.sno	sp.sno
1	1	S1	S1
1	2	S1	S1
1	3	S1	S1
1	4	S1	S1
…	…	…	…
4	10	S4	S4
4	11	S4	S4
4	12	S4	S4

SQL92 – Join-Berechnung IV/IV

SQL:

```
SELECT          s.sname, sp.pno
```

s.rowID	sp.rowID	s.sname	s.sno	sp.sno
1	1	Smith	S1	S1
1	2	Smith	S1	S1
1	3	Smith	S1	S1
1	4	Smith	S1	S1
…	…		…	…
4	10	Clark	S4	S4
4	11	Clark	S4	S4
4	12	Clark	S4	S4

SQL92 – Join-Zusammenfassung

1. Schritt
   Kreuzprodukt aller Tabellen

2. Schritt
   Zusammengehörige Zeilen gemäß ON-Bedingung bestimmen

3. Schritt
   Restriktion gemäß WHERE

4. Schritt
   Projektion, um die gewünschten Spalten zu bestimmen

SQL92 – Join-Regeln

- **Spaltentypen müssen kompatibel sein**
  - Empfehlung:
    Geschwindigkeitsvorteil bei Identität der Spaltentypen
- **Spaltennamen müssen nicht identisch sein**
- **JOIN-Operator: =, >, <, >=, <=**
- **Restriktionsbedingungen zulässig**
- **Tabellenanzahl für Join vom DBMS abhängig**
- **JOIN-Operator = wird als Equi-Join bezeichnet**

SQL92 – Natural Join

JOIN der gleiche Spaltennamen verbindet

```
SELECT          s.sname, sp.pno
FROM            supp_parts sp
                NATURAL JOIN         suppliers s
     WHERE      sp.pno = 'P1';
```

Resultat:

```
sname  pno
------ --------
Smith  P1
Jones  P1
(2 row(s) affected)
```

SQL92 – kaskadierte Joins

JOIN der mehr als zwei Tabellen verbindet

```
SELECT          s.sname, sp.pno
FROM            supp_parts sp
                INNER JOIN suppliers s
                ON         sp.sno = s.sno
                INNER JOIN parts p
                ON         sp.pno = p.pno
      WHERE     qty > 300;
```

Resultat:

```
sname  pname  qty
------ ------ ----
Smith  Screw  400
Jones  Bolt   400
Clark  Cam    400
(3 row(s) affected)
```


3.12 Inner-/Outer-Join

SQL92 – Inner-/Outer-Join

- **JOIN-Bedingung ON kann eine Eingabe von NULL haben**
- **INNER JOIN verwirft Zeilen mit Join-Attributen, die NULL sind**
- **OUTER JOIN erlaubt dies**
  - LEFT OUTER JOIN: linke Tabelle NULL
  - RIGHT OUTER JOIN: rechte Tabelle NULL
  - FULL OUTER JOIN: linke oder rechte Tabelle NULL

SQL92 – LEFT OUTER JOIN

Welche Teile liegen in der selben Stadt wie welche Händler?
Händler ohne entsprechendes Teil sollen gelistet werden.

```
SELECT          sno, pno
FROM            suppliers s
                LEFT OUTER JOIN       parts p
                ON         s.city = p.city
```

Resultat:

```
sno      pno
-------- --------
S1       P1
S1       P4
S1       P6
S2       P2
…
S3       P5
S4       P1
…
S5       NULL
(11 row(s) affected)
```


SQL92 – RIGHT OUTER JOIN

Welche Teile liegen in der selben Stadt wie welche Händler?
Teile ohne entsprechende Händler sollen auch gelistet werden.

```
SELECT          sno, pno
FROM            suppliers s
                RIGHT OUTER JOIN      parts p
                ON         s.city = p.city
```

Resultat:

```
sno      pno
-------- --------
S1       P1
…
S3       P2
NULL     P3
S1       P4
S4       P4
…
S4       P6
(11 row(s) affected)
```

SQL92 – FULL OUTER JOIN

Welche Teile liegen in der selben Stadt wie welche Händler?
Teile ohne entsprechende Händler und Händler ohne entsprechende Teile sollen auch gelistet werden.

```
SELECT          sno, pno
FROM            suppliers s
                FULL OUTER JOIN       parts p
                ON         s.city = p.city
```

Resultat:

```
sno      pno
-------- --------
S1       P1
…
S3       P2
NULL     P3
S1       P4
S4       P4
…
S4       P6
S5       NULL
(12 row(s) affected)
```


SQL92 – NOT IN

Alle Händler aus Städten, wo es keine Teile gibt.

```
SELECT          sname, city
FROM            suppliers s
     WHERE      city NOT IN (
                                    SELECT     city
                                    FROM       parts
                           );
```

Resultat:

```
sname  city
------ -------
Adams  Athens
```

3.13 Mengenoperationen

SQL92 – Mengenoperatoren

- **UNION:**
  Vereinigung der Ergebniszeilen
- **UNION ALL:**
  Vereinigung der Ergebniszeilen inkl. Dupletten
- **INTERSECT:**
  Schnittmenge der Ergebniszeilen
- **MINUS:**
  Restmenge der Ergebniszeilen; erstes SELECT \ zweiter SELECT

SQL92 – UNION

Alle Städte, in denen Händler oder Teile sind.

```
SELECT          city
FROM            suppliers

     UNION

SELECT          city
FROM            parts;
```

Resultat:

```
city
----
Athens
London
Paris
Rome
(4 row(s) affected)
```

SQL92 – UNION-Regeln

- Beliebig viele Zeilenmengen können vereinigt werden
- Spaltennamen der Zeilenmengen können verschieden sein
- Spaltentypen müssen kompatibel sein
- Spaltentyp der Ergebnismenge wird von erster Zeilenmenge bestimmt

4 Data Manipulation Language

4.1 INSERT-Anweisung

SQL DML – Zeilen einfügen

SQL-Syntax:

```
INSERT INTO     <table_name>
     [(col1, col2, …, colN)]
                VALUES (val1, val2, …, valN)
```

Anzahl an col und der angegebenen Werte val muss übereinstimmen
Reihenfolge bestimmt die Zuordnung
nicht angegebene Attribute (erste Klammer) werden
mit DEFAULT-Wert belegt,
mit NULL belegt.
in der ersten Klammer fehlende Attribute müssen mit DEFAULT-Wert oder NULL-Wert definiert sein

SQL DML – INSERT-Beispiel

Es gibt einen neuen Händler S6:

```
INSERT INTO     suppliers
     (sno, sname, status, city)
               VALUES ('S6', 'Miller', 10, 'Orlando')
```


SQL DML – INSERT mittels Abfrage

kopieren eines Abfrageergebnisses in eine Tabelle:

```
BEGIN TRANSACTION
     INSERT INTO parts(pno, pname, color, weight, city)

     SELECT 'TRN_' + pno, pname, color, weight, city FROM parts;

     SELECT     *          FROM parts;
ROLLBACK
```

4.2 DELETE-Anweisung

SQL DML – Zeilen löschen

SQL-Syntax:

```
DELETE FROM      <table_name>
                 [ WHERE <search_condition> ]
```

WHERE schränkt den Umfang des Löschens ein
DELETE entfernt die Daten aus der Datenbank

SQL DML – DELETE-Beispiel

Es gibt keine Händler S1 … S3 mehr:

```
BEGIN TRANSACTION
      SELECT count(*) FROM supp_parts;
      SELECT count(*) FROM suppliers WHERE sno <= 'S3';

      DELETE FROM suppliers WHERE sno <= 'S3';

      SELECT * FROM suppliers;
      SELECT count(*) FROM supp_parts;
ROLLBACK
```

SQL DML – DELETE-Effekte

- DELETE löscht Daten unter Beachtung der Foreign-Key-Constraints
- Datenbankschema sollte bekannt sein, um "Effekte" durch das Löschen beurteilen zu können
- zum Löschen Transaktionen nutzen

4.3 Tabelle löschen

SQL DML – Tabelle löschen

SQL-Syntax:

```
DELETE TABLE     <table_name>
                 [ CASCADE ]
```

löschen der Tabelle inkl. aller Daten
CASCADE entfernt abhängige Datensätze in anderen Tabellen

4.4 UPDATE-Anweisung

SQL DML – Zeilen verändern

SQL-Syntax:

```
UPDATE          <table_name>
     SET        col = val
                [ WHERE <search_condition> ]
```

UPDATE verändert die Daten der Tabelle
UPDATE sollte in einer Transaktion ausgeführt werden

SQL DML – UPDATE

Händler Miller zieht nach Houston um und bekommt Status 15:

```
BEGIN TRANSACTION
     SELECT * FROM suppliers WHERE sno = 'S6';

     UPDATE suppliers
               SET status = 15, city = 'Houston'
                          WHERE sno = 'S6';

     SELECT * FROM suppliers WHERE sno = 'S6';

ROLLBACK
```

SQL DML – INSERT & UPDATE

SQL:

```
BEGIN TRANSACTION
      INSERT INTO          suppliers (sno, sname, status, city)
                 VALUES ('S7', 'Smith', 10, 'Orlando');
      INSERT INTO          supp_parts(sno, pno, qty)
                 VALUES ('S7', 'P1', 1200);

      UPDATE supp_parts SET qty = 1300
                 WHERE      sno = 'S7' AND pno = 'P1';
      UPDATE suppliers SET sno = 'S8'
                 WHERE      sno = 'S7';

      SELECT * FROM suppliers;
      SELECT * FROM supp_parts WHERE qty > 1000;
ROLLBACK
```


5 Sichten

Sichten - Views

- **Sichten – Views**
  virtuelle Tabellen, Attribute und deren Datentypen stammen von den Basistabellen
  Berechnung der "Tupel" zur Laufzeit (Abfrage)
  Anpassung an spezielle Benutzerbedürfnisse, Datenschutz usw.
  zum Verbergen von "komplexen" Strukturen (Bsp.: Primär-Fremdschlüssel-Beziehungen (Joins))

- **Vorteile**
  Adaption an verschiedene Benutzerklassen
  - nur relevante Daten werden angezeigt
  - Spalten können benutzungsfreundlich präsentiert werden

  Schemaänderungen sind möglich

- **Syntax:**

```
CREATE VIEW <view_name>
      [(<column_name> [, <column_name> …] ) ]
                 AS <query_expression>
                            [ WITH CHECK OPTION ]
```

View-Beispiel

Sicht, die alle roten Teile umfasst:

```
CREATE VIEW red_parts
     AS
                SELECT *
                          FROM parts
                                    WHERE color = 'red'

SELECT * from red_parts;
```

Resultat:

```
pno      pname           color weight city
-------- --------------- ----- ------ ------
P1       Nut             red   12.0   London
P4       Screw           red   14.0   London
P6       Cog             red   19.0   London
(3 row(s) affected)
```


INSERT in View

Mehrere Teile über den View red_parts einfügen:

```
INSERT INTO red_parts
     VALUES('P7', 'Washer', 'pink', 12, 'Rome');
INSERT INTO red_parts
     VALUES('P8', 'Cog', 'red', 0.5, 'Rome');
INSERT INTO red_parts
     VALUES('P9', 'Nut', 'red', 15, 'Rome');
```

Resultat:

INSERT in View

Sicht, die alle schweren, roten Teile umfasst:

```
CREATE VIEW highweight_red_parts
     AS
               SELECT *
                         FROM red_parts
                                   WHERE weight >= 10.0
```

Resultat:

```
pno      weight
-------- ------
P1       12.0
P4       14.0
P6       19.0
P9       15.0
(4 row(s) affected)
```


INSERT in View – CHECK OPTION

Sicht, die alle schweren, roten Teile umfasst:

```
INSERT INTO highweight_red_parts
     VALUES
               ('PA', 'Washer', 'red', 5, ' Stuttgart');
```

Resultat:

```
Msg 550, Level 16, State 1, Line 1
The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or
spans a view that specifies WITH CHECK OPTION and one or more rows resulting from the operation did
not qualify under the CHECK OPTION constraint.
The statement has been terminated.
```

Umfassendes VIEW-Beispiel

Sicht, die Anzahl aller Händler und Anzahl der Teile je Stadt angibt

…

Resultat:

```
city      TAnz        HAnz
--------- ----------- -------------------
Athens    NULL        1
London    3           2
Orlando   NULL        1
Paris     2           2
Rome      1           NULL
```


Änderungen auf Sichten

- **Voraussetzungen für das Ändern auf Sichten:**
  - nur eine Basisrelation
  - Schlüssel muss (komplett) sichtbar bleiben
  - keine Verwendung von
    - Aggregaten
    - Gruppierungen
    - Duplikateliminierung (DISTINCT)

6 Data Definition Language

SQL-Data Definition Language

- **Anweisungen, um Datenstrukturen zu definieren**
- **Definition der**
  - Datenbankstruktur (Tabellenerstellung usw.)
  - Speicherungsstrukturen (Heap usw.)
  - Zugriffshilfsstrukturen (Indizes usw.)

6.1 Tabelle erstellen

Tabelle anlegen

- **Beliebig viele Zeilenmengen können vereinigt werden**

```
CREATE TABLE     <table-name>
       (Spaltendefinition1, …, SpaltendefinitionN
       [, Integritätsregel1, …, IntegritätsregelM] )
```

- **wobei**

```
SpaltendefinitionI ::=      Spaltenname Typeangabe
       [DEFAULT Klausel]
       [Spaltenintegritätsregel]
```

- **Tabellenname muss innerhalb eines Schemas eindeutig sein**
- **Spaltenname muss innerhalb einer Tabelle eindeutig sein**
- **Spaltenintegritätsregel, z.B.: NOT NULL**

SQL92 – Create Table-Beispiel

2-spaltige Tabelle <example>, keine NULL-Werte

```
CREATE TABLE example
                (
                        id          INT         NOT NULL,
                        value       FLOAT       NOT NULL
                );
```


6.1.1 Integritätsbedingungen

Integritätsbedingungen

- **Operationale Integrität**
  - Gewährleisten der Funktion nach Hardwarefehlern, Vandalismus
  - Sicherung gegen Probleme, die durch Mehrbenutzerbetrieb entstehen
- **Semantische Integrität**
  - Konsistenz zur Laufzeit gemäß den definierten Bedingungen:
    - Entitäts-Integrität
      - Eine Zeile ist in der Tabelle eindeutig (Bsp.: CREATE UNIQUE INDEX, UNIQUE, PRIMARY KEY)
    - Wertebereichs-Integrität
      - Ein Wert wird auf eine gültige Domäne beschränkt (Bsp.: CHECK, DEFAULT, NOT NULL)
    - Referentielle-Integrität
      - Fremdschlüsselbeziehungen werden überwacht (Bsp.: FOREIGN KEY)
    - Benutzerdefinierte-Integrität
      - Modellierte Bedingungen (Bsp.: Trigger)

6.1.1.1 Primärschlüssel

Primärschlüssel

Attribut einer Tabelle, welches einen Satz eindeutig charakterisiert (Entitäts-Integrität)

```
CREATE TABLE p
                (
                            id          INT         NOT NULL,
                            name        VARCHAR(32),
                            PRIMARY KEY (id)
                );
INSERT INTO p VALUES (1, 'Ulf');
INSERT INTO p VALUES (2, 'Ute');
INSERT INTO p VALUES (1, 'Uwe');
```

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-02024

Zusammengesetzte Primärschlüssel

Ein Attribut ist nicht eindeutig
Primärschlüssel mittels mehrerer Attribute bildbar

```
ALTER TABLE Supp_Parts
     ADD CONSTRAINT PK_Supp_Parts
                PRIMARY KEY
                (
                            sno,
                            pno
                );
```

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2013-02025

6.1.1.2 Eindutige Spalten

Eindeutige Spalten

Nicht-Primärschlüssel-Attribute können mit UNIQUE auf Eindeutigkeit geprüft werden
UNIQUE-Attribute bezeichnen wir als Schlüsselkandidaten
Bsp.: Studierendentabelle mit id, name. (name soll eindeutig sein)

```
CREATE TABLE p
                (
                          id        INT       NOT NULL,
                          name      VARCHAR(32),
                          PRIMARY KEY (id),
                          UNIQUE (name)
                );
INSERT INTO p VALUES (1, 'Ulf');
INSERT INTO p VALUES (2, 'Ute');
INSERT INTO p VALUES (3, 'Ulf');
```


6.1.1.3 Check-Bedingung

Check-Constraint

- **Bedingung für erlaubte Attributwerte**

```
CHECK <Bedingung>
```

<Bedingung> wird formuliert analog zur WHERE-Klausel

- **Bsp.: Studierenden-Tabelle mit Name, Anrede; Anrede nur "Frau" oder "Herr"**

```
CREATE TABLE p
                (
                          id        INT       NOT NULL,
                          name      VARCHAR(32),
                          anr                 char(4),
                          PRIMARY KEY (id),
                          CHECK (anr IN ('Frau', 'Herr'))
                );
INSERT INTO p VALUES (1, 'Ulf', 'Herr');
INSERT INTO p VALUES (2, 'Ute', 'Frau');
INSERT INTO p VALUES (3, 'Don', 'Mr.');
```

6.1.1.4 Default-Bedingung

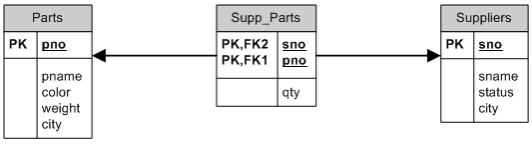
Default-Constraint

- **Standardwert für ein Attribut, wenn beim INSERT nicht spezifiziert**
- **Bsp.: Studierenden-Tabelle mit Name; wenn Name nicht spezifiziert, dann den angemeldeten Datenbankbenutzer**

```
CREATE TABLE p
                (
                        id          INT         NOT NULL,
                        name        VARCHAR(32)           DEFAULT current_user,
                        PRIMARY KEY (id),
                );
INSERT INTO p (id) VALUES (1234);
```


6.1.2 Referentielle Integrität

Referentielle Integrität I/III



Referentielle Integrität basiert auf sog. Fremdschlüsselbeziehungen
Bsp.:

```
CONSTRAINT <constraint_name> FOREIGN KEY
                (
                        <attribute_name>
                ) REFERENCES <table> ( <attribute_name> )
                { ON DELETE
                        [NO ACTION | CASCADE | SET NULL | SET DEFAULT ] }
                { ON UPDATE
                        [NO ACTION | CASCADE | SET NULL | SET DEFAULT ] }
```

Referentielle Integrität II/III

- **Bedeutung der Schlüsselwörter ON DELETE, ON UPDATE**

- **NO ACTION**
  - DELETE oder UPDATE der Herkunftstabelle wird verweigert, wenn es einen abhängigen Satz gibt

- **CASCADE**
  - DELETE oder UPDATE der Herkunftstabelle wird an die abhängige Tabelle durchgereicht

- **SET NULL**
  - Betroffene Zeilen der abhängigen Tabelle werden auf NULL gesetzt

- **SET DEFAULT**
  - Betroffene Zeilen der abhängigen Tabelle werden auf den DEFAULT-Wert der betroffenen Attribute gesetzt

Referentielle Integrität III/III

- **Beispiel:**
  Die Tabelle supp_parts ist von der Tabelle parts und der Tabelle suppliers abhängig. Jede Änderung soll an die Tabelle supp_parts weitergereicht werden.

```
CREATE TABLE supp_parts
      (
                sno                 VARCHAR(8) NOT NULL,
                pno                 VARCHAR(8) NOT NULL,
                qty                 int NULL,

                CONSTRAINT PK_supp_parts
                           PRIMARY KEY (sno, pno),

                CONSTRAINT FK_supp_parts_parts FOREIGN KEY (pno)
                REFERENCES parts (pno)
                           ON DELETE CASCADE ON UPDATE CASCADE,

                CONSTRAINT FK_supp_parts_suppliers FOREIGN KEY (sno)
                REFERENCES suppliers (sno)
                           ON DELETE CASCADE ON UPDATE CASCADE
      );
```

7 Einführung

7.1 Grundbegriffe

Einführung: Grundbegriffe – Datenmanagement

Konkrete Aufgabenbereiche

- Spezifikation und Realisierung des Datenmodells (Datenanalyse)
- Inproduktionsnahme des Datenmodells
- Organisation der Datenbeschaffung
- Wartung und Pflege des DBMS-System

} Ständig **durchzuführende Aufgaben**

Prof. Dr.-Ing. Peter Steininger
2026-SS

Quelle: HEINRICH 2005
DBM-0001-2012-00422

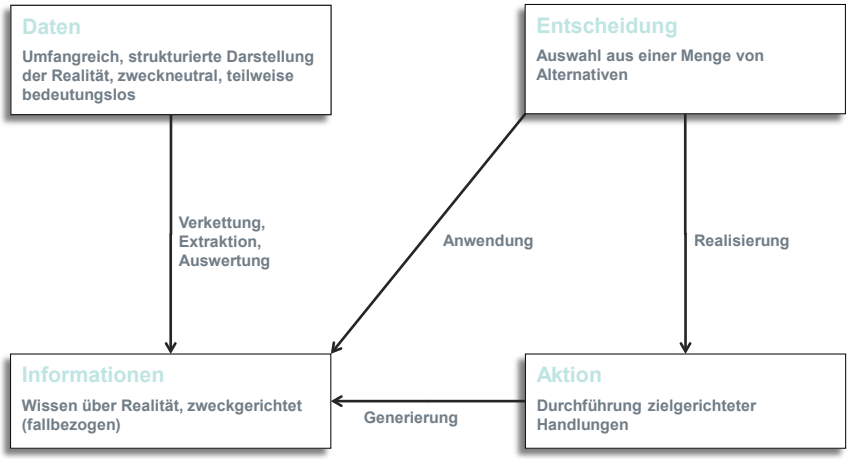
Einführung: Grundbegriffe – Informationssystem

Werkzeug zur Erfassung und Kommunikation von Informationen zum Zwecke der Erfüllung der Anforderungen seines Benutzers, der (Geschäfts-) Aktivitäten ihres Unternehmens und zur Erreichung der Unternehmensziele. Ein Informationssystems umfasst Daten, Datenbanksoftware, Rechner-Hardware, Personen, welche die Daten benutzen und verwalten, relevante Anwendungssoftware sowie Programmierer, die diese entwickeln.

Prof. Dr.-Ing. Peter Steininger
2026-SS

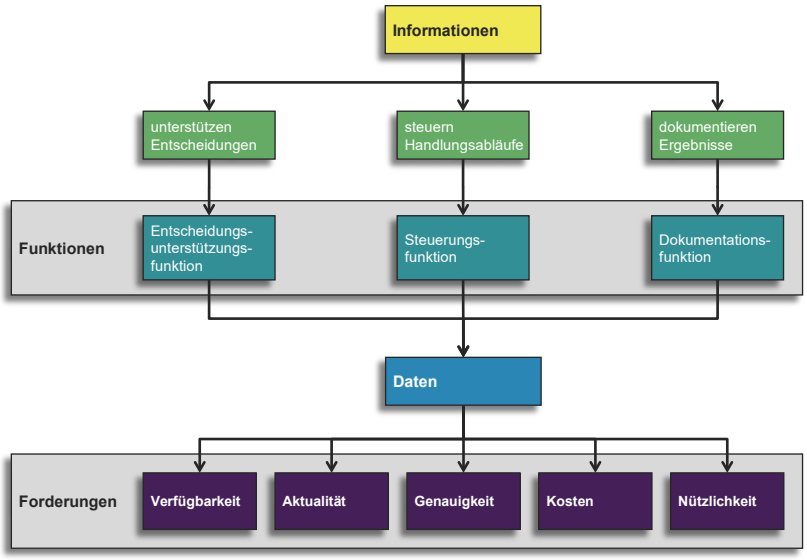
DBM-0001-2012-00423

Einführung: Grundbegriffe – Daten und Informationen



Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00424

Einführung: Grundbegriffe – Daten und Informationen



Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00425

Einführung: Grundbegriffe – Bedeutung von Informationen

Für die Wirtschaft

- Informationen sind die Basis für einen effektiven und effizienten Einsatz von Produktionsfaktoren.
  - Kontrolle
  - Durchführung
  - Planung
  - Entscheidung
  - Steuerung

  } Informationen

Einführung: Grundbegriffe – Bedeutung von Informationen

Information

- bestimmen das Überleben der Weltwirtschaft
- strategische Bedeutung
- zur richtigen Zeit, die richtige Informationen, am richtigen Ort
- Entscheidungsunterstützung
  - Datenbanken,
  - Data Warehouse,
  - Online Analytical Processing (OLAP),
  - Data Mining,
  - Management Information System (MIS),
  - Knowledge Management,
  - Customer Relationship Management (CRM)

Einführung: Grundbegriffe – Wissensmanagement

Inhalt

- Aufbau und Sicherung von Wissen
- Fakten (Wissen) zur Verfügung stellen
- Optimierung der Wissensverteilung

Bereiche

- Allgemeinwissen
- Fach Know-How (Was benötige ich und Wie wende ich es an)
- Soft-Skills (Wann wende ich es an)

Warum **"Informationsüberflutung"**

Prof. Dr.-Ing. Peter Steininger
2026-SS

Customer Relationship Management

Einführung: Grundbegriffe

Warum

- **Globalisierung** der Märkte
- Steigende **Transparenz** und Zunahme von Informationsquellen
- **Kundenanforderung** bzgl. der Produkte **nehmen zu**

Inhalt

- **Ausrichtung** der Geschäftsprozesse und Dienstleistungen auf die **Bedürfnisse** der Kunden
- **Identifikation** der profitablen **Kunden**
- segmentspezifische Marketingstrategien zur **Akquisition** von neuen Kunden

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00429

Einführung: Grundbegriffe – Prinzipien/Sichten

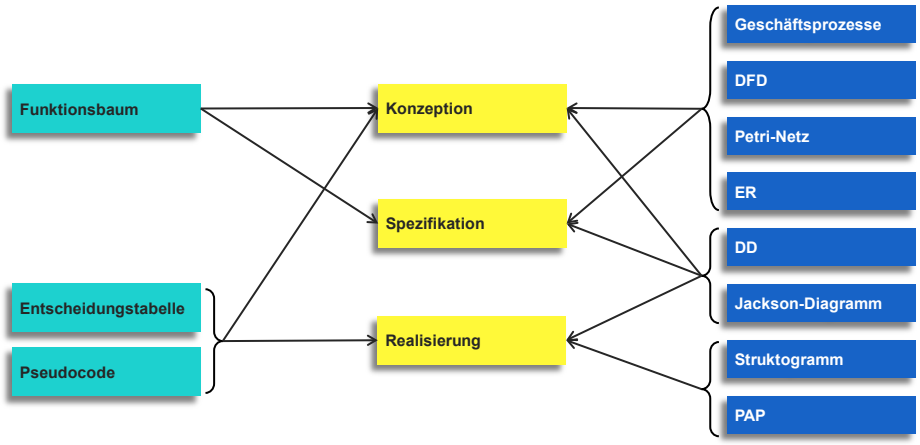
Zerlegung:

- Top-Down,
- Bottom-up

Sichten:

- **funktionsorientiert**
  Funktionen und Unterfunktionen / keine explizite Programm übergreifende Datenmodellierung
- **datenorientiert**
  ERM, Data Dictionary
- **objektorientiert**
  Klassen, Objekte, Attribute, Methoden, Kapselung, Vererbung/gleichzeitige Modellierung von Daten und Funktionen
- **prozessorientiert**
  Ereignis und Prozess

Einführung: Grundbegriffe – Methoden



Einführung: Grundbegriffe – Requirement Engineering

Anforderungstechnik. Das systematische, disziplinierte und quantitativ erfassbare Vorgehen beim Spezifizieren, d. h. Erfassen, Beschreiben und Prüfen von Anforderungen an Software. Verstehen und Beschreiben, was der Kunde wünscht oder braucht.

Einführung: Grundbegriffe – Ebenen der Modellierung

Systemanalyse = Systematische Vorgehensweise zur Anforderungsermittlung (Requirement Engineering). Festlegen der quantitativen und qualitativen Eigenschaften des zu entwickelnden Softwareproduktes (Sollkonzept)

Ebenen:

- Design
- Realisierung
- Inbetriebnahme/Wartung

7.2 Datenspeicherung

Einführung: Grundbegriffe – unstrukturiert

- Word-Dokument mit Personen bezogenen Daten

```
Business:

Willi Meier, Schillerstr. 3, 023123

Müller, Hans, 80000 München, Geburtstag: 23.4.1987, Schuhverkäufer

Jansen, Paul; Handy: 0172 65 78 78; 182 m; Schuhgröße 47
Aalen

Privat:

Thomas Berg, 45886 Gelsenkirchen, 100 Euro offen
```

Prof. Dr.-Ing. Peter Steininger
2014-WS
DBM-0001-2012-00434

Einführung: Grundbegriffe – strukturiert

- Dateien:

Name	Vorname	Straße	Nr.	PLZ	Ort	Datum
Meier	Willi	Schillerstr.	3	8000	München	23.04.02
Müller	Hans	Fischerstr.	3	45884	Gelsenkirchen	23.04.87

- sequenzielle Abarbeitung der Dateien (z.B. Magnetbänder, -kassetten)
- Sprachen zur Verwaltung großer Datenbestände: COBOL, FORTRAN, PASCAL
- Applikationen speichern "eigene Daten" isoliert in "eigenen Files"
- Datenspeicherung und Zugriff auf Daten nur durch spezielle Anwendungen

Prof. Dr.-Ing. Peter Steininger
2014-WS
DBM-0001-2012-00435

8 Datenmodellierung

8.1 Grundlagen

Datenmodellierung

Gegenstand der Datenmodellierung ist es, alle relevanten Informationen eines Systems, z. B. eines betrieblichen Funktionsbereiches (Vertrieb), zu beschreiben. Man benötigt dazu eine Syntax, die den Sachverhalt möglichst einfach, mit wenigen Syntaxelementen aber genügend formal, um exakt sein zu können, zu beschreiben.

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00436

Datenmodellierung: Einsatzgebiete

Entwicklung von Anwendungssystemen

- Individualsoftware im technische oder kommerziellen Bereich

Einführung von Informationssystemen

- Standardsoftware (SAP, ALCIB, ALCIM)

Organisationsprojekte

- z.B. Business Process Redesign

Prof. Dr.-Ing. Peter Steininger
2026-SS
-0001-2012-00437

Datenmodellierung: Datenorientiertes Vorgehen

"Daten leben ewig!"

- Hardware besteht einige Jahre
- Software übersteht die Hardware
- Daten überleben mehrere Softwarezyklen

Datenmodelle sind schnell umsetzbar, Vermeidung von Redundanz

Betriebswirtschaftliche Basis für Integration und Standardisierung

Die Datenorganisation bestimmt wesentlich

- die Systemleistung und
- die Datensicher und den Datenschutz.

Datenmodellierung: Sinn der Datenmodellierung

Unterschiedliche Zielgruppen:

- stellen unterschiedliche Ansprüche an das Informationssystem und
- verwenden unterschiedliche Fachsprachen.

Dient als Kommunikationsmittel zwischen

- IS-Entwickler, Endbenutzer und Management (Auftraggeber)

Durch Modellierung Abbildung der

- Aufbauorganisation,
- Daten,
- Funktionen und
- Ablauforganisation.

Datenmodellierung: Zweck der Modellierung

Informationssystem-Entwickler:

- Kommunikation, Qualitätssicherung und Dokumentation.

Endbenutzer:

- Kommunikation, Standardisierung, transparente Dokumentation der Aufgabe.

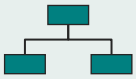
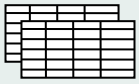
Management (Auftraggeber):

- Aufwandschätzung,
- Aufzeigen organisatorischer Gestaltungsspielräume,
- Kommunikation,
- Projektplanung,
- Investitionsentscheidungen,
- Dokumentation der Geschäftsprozesse

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00440

8.2 Phasen

Datenmodellierung: Phasen

Ebene	Objekt		Methode
Ebene 1	Reale Welt		• Verkauf • Produktion
Ebene 2	Betriebswirtschaft Modellierung		• Betrieblich Funktionen • Methode: Istanalyse
Ebene 3	Semantische Modellierung		• ERM
Ebene 4	Logische Modellierung		• Normalisierung • Relationales Modell
Ebene 5	Physische Modellierung		• DB Know-how • Tabellen

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00441

Datenmodellierung: Betriebswirtschaftliche Modellierung (I/II)

Abgrenzung Miniwelt (Umwelt)

Informationsbedarfsanalyse:

- Erhebung der Informationsobjekte
- Interviews, Analyse vorhandener Systeme usw.
- Methoden der Istanalyse
- Sammeln und Einordnen von Begriffen und Aussagen

Datenmodellierung: Betriebswirtschaftliche Modellierung (II/II)

Begriffe:

- **Homonyme Begriffe**:
  Begriffe mit unterschiedlicher Bedeutung (Bank/Bank)

- **Äquipollenzen** (lateinisch aequipollens = gleichviel geltend):
  Unterschiedliche Sichtweisen der selben Objekte
  - Mitarbeiter,
  - Aufträge (Auftrag aus rechtlicher, organisatorischer, buchhalterischer Sicht)

- **Vage Begriffe**:
  Dinge, die jeder erwähnt, aber nicht genau definiert sind (Bsp. Multimedia)

- **Bezeichner**:
  Problem der Identifikation von z. B. Artikelnummern (unterschiedlich in verschiedenen Bereichen)

Datenmodellierung: Semantische Modellierung (I/II)

Semantik = Lehre von der Bedeutung sprachlicher Zeichen

Modellierung der Miniwelt aus Unternehmenssicht durch Fachabteilung

Von der Implementierung und Datenbanksystem unabhängig

Semantisches Datenmodell (Entity Relationship Modell (ERM)):

- wirklichkeitsnah,
- ohne technische Gesichtspunkte der Implementierung,
- wenig Symbole → verständlich
- Kommunikation Informationssystem ← → Business

Datenmodellierung: Semantische Modellierung (II/II)

Vorteile:

- Verbessern der Kommunikation zwischen Anwender und Entwickler
- Verbessern der Analysefähigkeit und Wartbarkeit
- Verbessern der Dokumentation
- Erhöhen der Datenqualität durch bessere Interpretierbarkeit der Begriffe
- Dient als Entscheidungsgrundlage für die Auswahl von Standardsoftware
- Schaffen eines einheitlichen unternehmensweiten Sprachgebrauches

Datenmodellierung: Logische Modellierung (I/II)

Auswahl logisches Paradigma (Datenstruktur)

- Netzwerkmodell
- Hierarchisches Modell
- Relationales Modell
- Objektorientiertes Modell

Relationales Modell:

- Auflösung von n:m – Beziehungen
- Normalisierung
- Felddatentypen zuweisen

Datenmodellierung: Logische Modellierung (II/II)

Datenzeitpunkt:

- statisch
- dynamisch

Datenattribut:

- numerisch
- alphanumerisch
- grafisch
- funktional

Datenoperatoren:

- syntaktisch
- semantisch

Datenmodellierung: Physische Modellierung

Festlegung von Zugriffsrechten

- Insert
- Update
- Delete
- Select

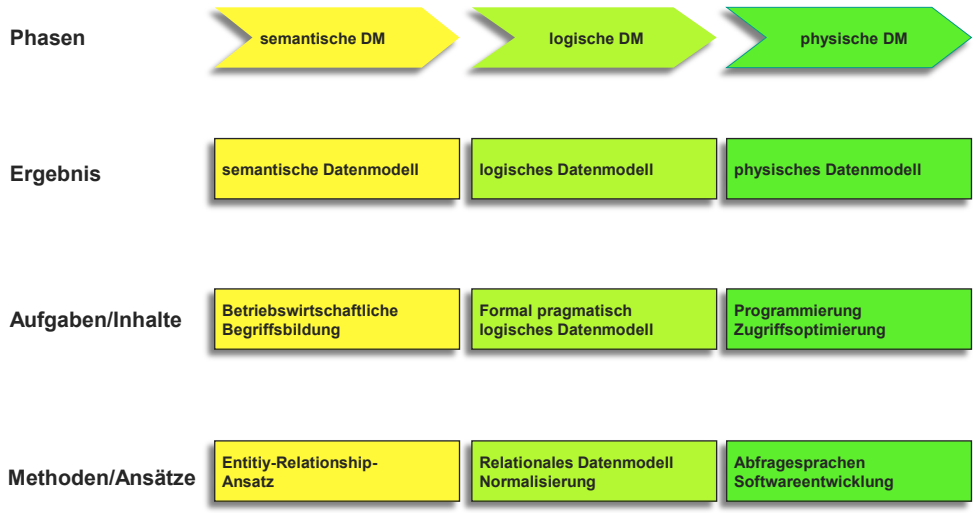
Optimierung des Datenbankzugriffs (Indizes)

Datenintegrität sicherstellen (Master/Detail)

Trigger, Verteilung, Backup usw.

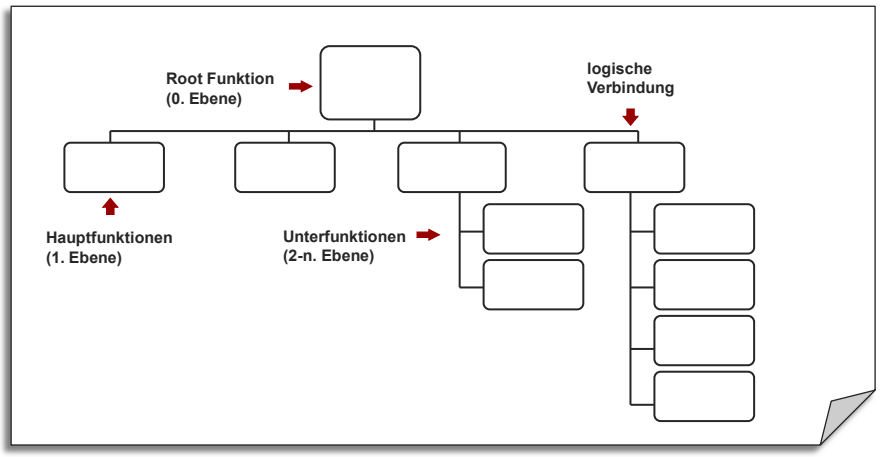
Arbeitsbereich eines Datenbank-Administrators

Datenmodellierung: Strukturierte Analyse



8.3 Basismethoden

Datenmodellierung: Strukturierte Analyse



Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00450

Datenmodellierung: Strukturierte Analyse

Beschreibung:

- zur Identifizierung der Geschäftsfunktionen, die innerhalb einer Organisation ausgeführt werden
- Eine Funktion ist eine fachliche Aufgabe bzw. Tätigkeit an einem Objekt zur Unterstützung eines oder mehrerer Unternehmensziele.
- Zerlegung bis zu Elementarfunktionen = Funktionen, die betriebswirtschaftlich nicht mehr sinnvoll zerlegbar sind.

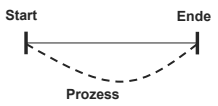
Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00451

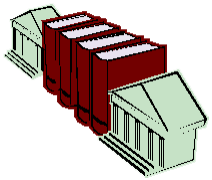
Datenmodellierung: Strukturierte Analyse

Prinzipien:

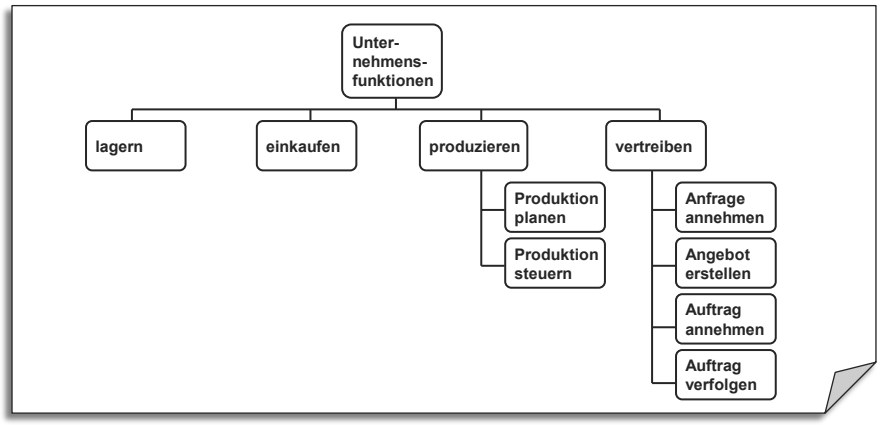
- Jede Funktion muss durch ihre Teilfunktionen vollständig repräsentiert werden
- keine Abläufe
- keine Reihenfolge
- keine Organisationsstrukturen

Funktion?

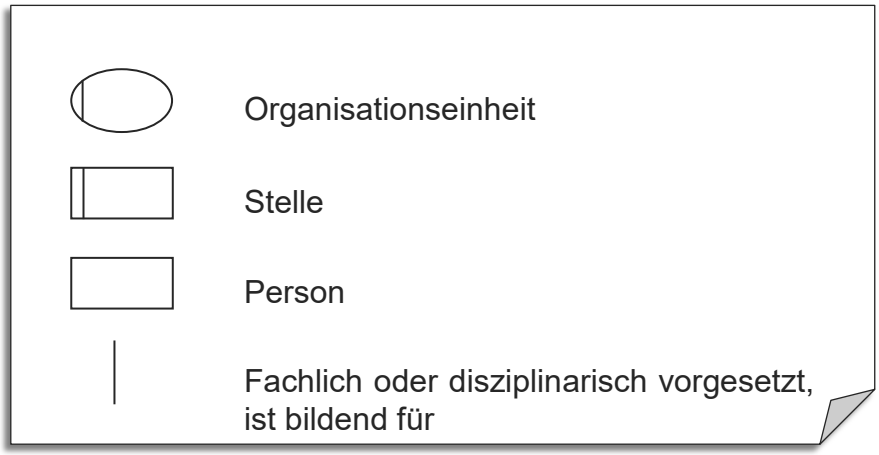




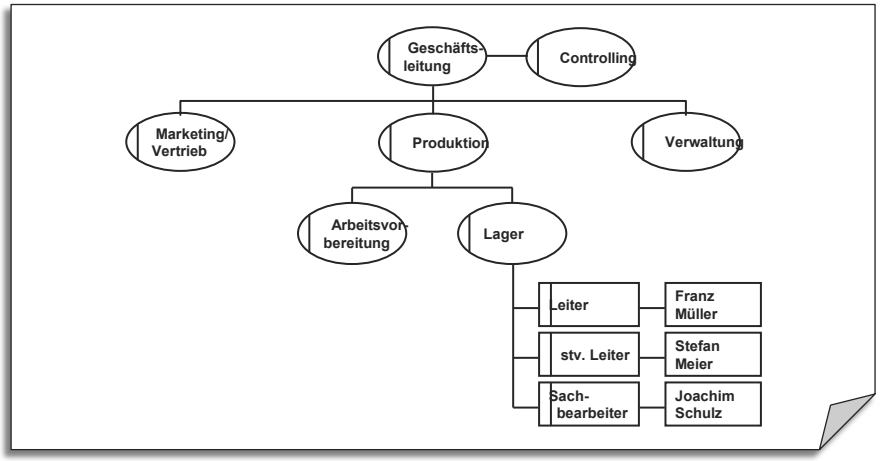
Datenmodellierung: Strukturierte Analyse



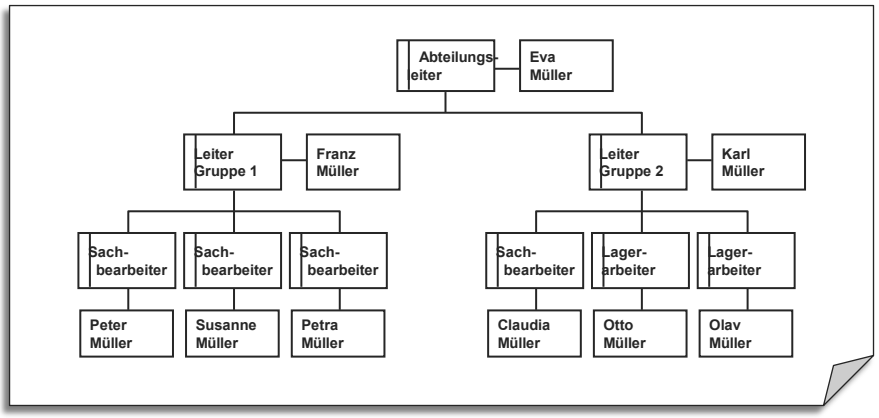
Datenmodellierung: Strukturierte Analyse



Datenmodellierung: Strukturierte Analyse



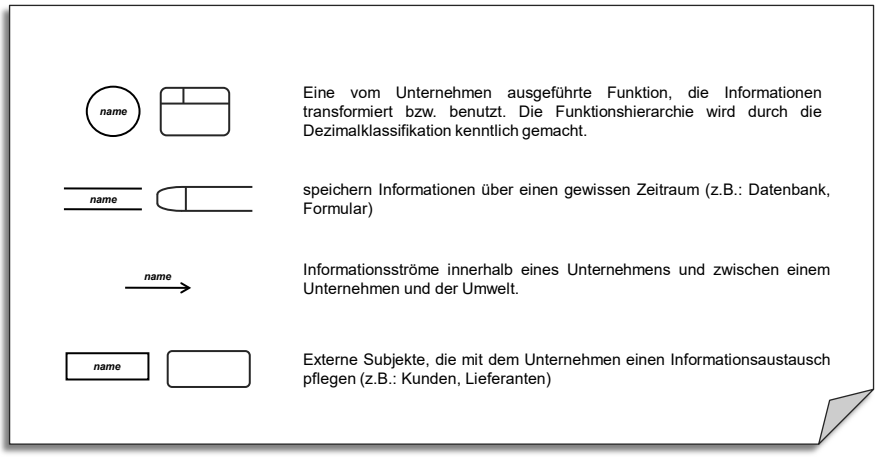
Datenmodellierung: Strukturierte Analyse



Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00456

8.4 Spezialmethoden

Datenmodellierung: Strukturierte Analyse



Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00457

Ausgangsbeispiel zur Erläuterung der Regeln:

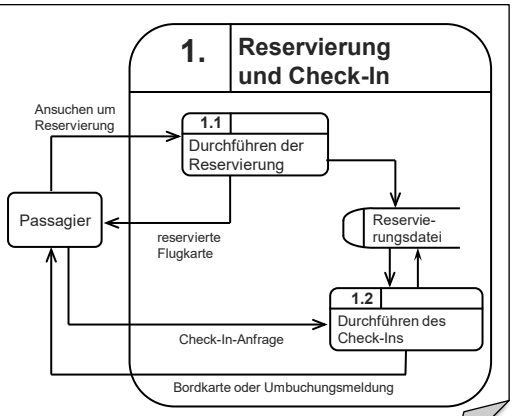
Datenmodellierung: Strukturierte Analyse

Durchführen der Reservierung:

Der Passagier sucht um eine Reservierung an. Bei Reservierungs-annahme wird die Reservierung gespeichert und der Passagier erhält ein Ticket.

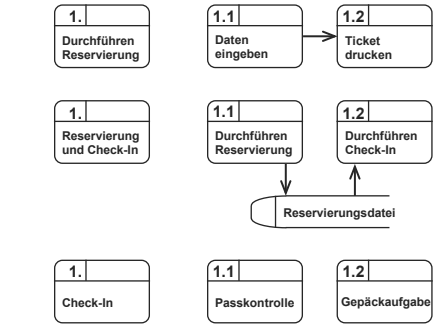
Durchführen des Check-Ins:

Der Passagier ist im Besitz einer Flugkarte und fragt um einen Check-In an. Wenn Plätze frei sind, bekommt er eine Bordkarte. Sonst wird sein Flug auf einen späteren Zeitpunkt umgebucht.



Grundregeln:

Datenmodellierung: Strukturierte Analyse

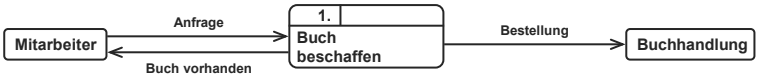


Aufgabe und Kontextdiagramm

Datenmodellierung: Strukturierte Analyse

Buchbeschaffung

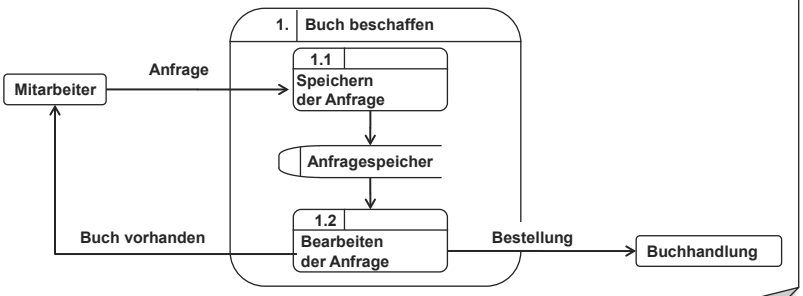
Ein Mitarbeiter fragt an, ob ein Fachbuch im Unternehmen vorhanden ist. Die Anfrage wird im System abgespeichert und anschließend bearbeitet. Im Rahmen der Bearbeitung erfolgt die Überprüfung, ob das Buch in der Abteilungsbibliothek oder in der Unternehmensbibliothek existent ist. Gegebenenfalls wird das Buch in einer Buchhandlung bestellt.



Lösung:

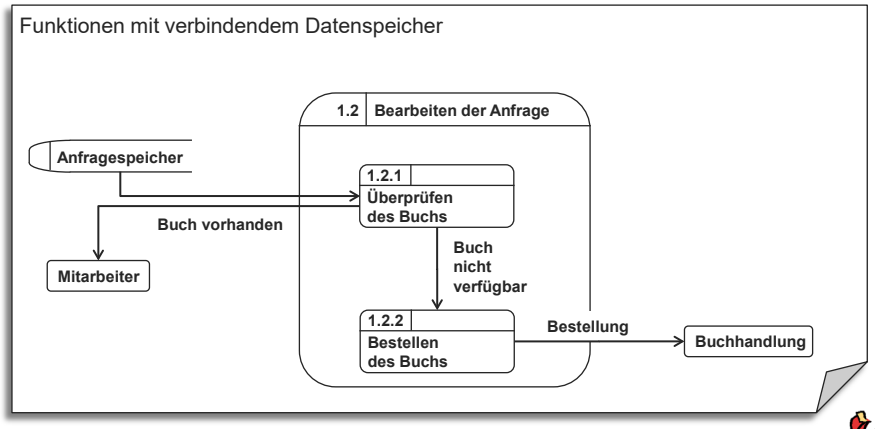
Datenmodellierung: Strukturierte Analyse

Funktionen mit verbindendem Datenfluss:



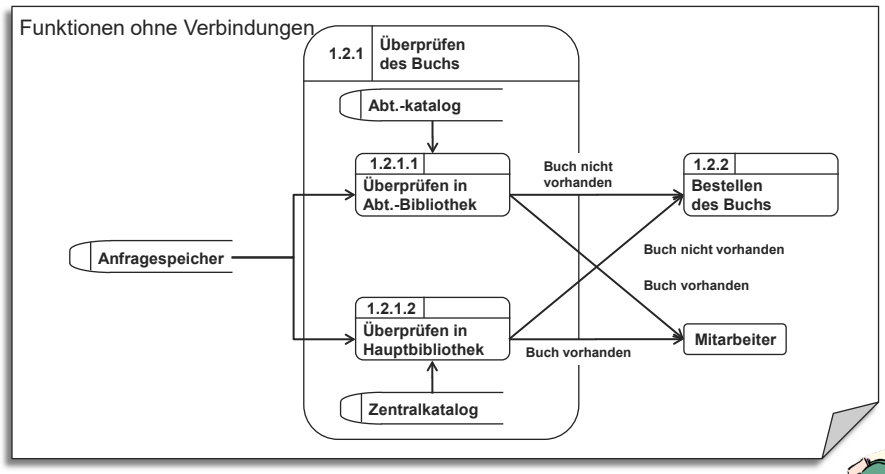
Lösung:

Datenmodellierung: Strukturierte Analyse

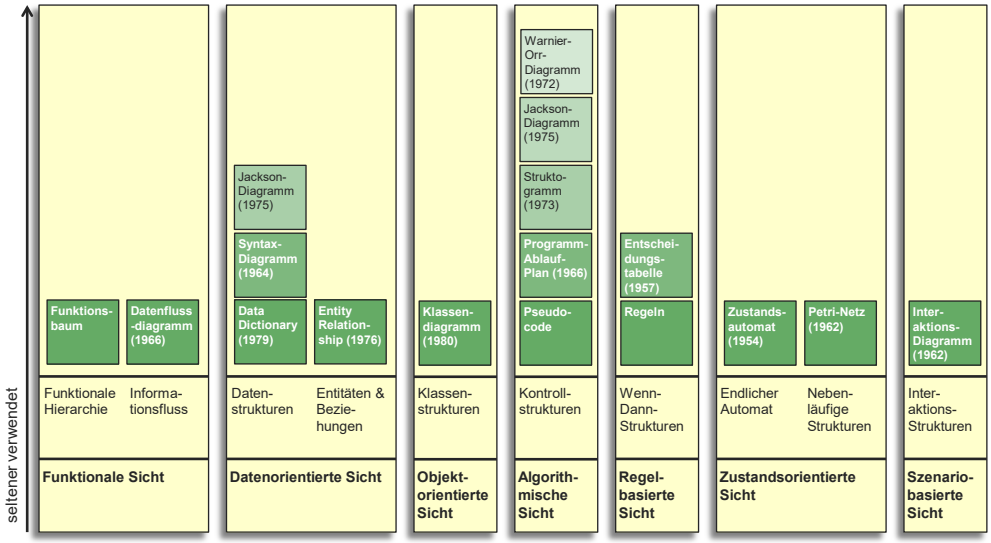


Lösung:

Datenmodellierung: Strukturierte Analyse



Datenmodellierung: Überblick



9 Entity-Relationship-Modell

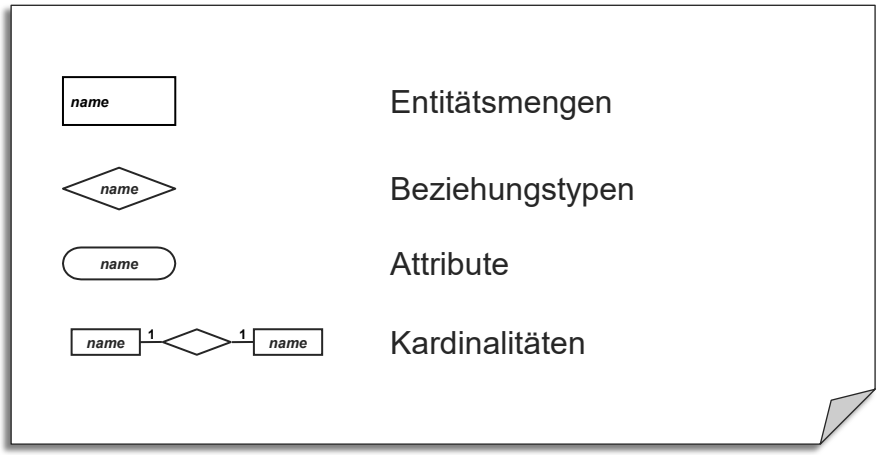
9.1 Grundlagen

Entity-Relationship-Modell

Ein Entity-Relationship-Modell (ER-Modell) ist eine anschauliche und leicht kommunizierbare Beschreibung der Datenwelt. Es ist Ergebnis der konzeptionellen Modellierung und stellt einen Ausschnitt der Realität dar, der für eine Problemstellung wichtig ist. Es beschreibt Objekte der Realwelt und ihre Beziehungen zueinander.

Entity-Relationship-Modell

Bestandteile:



Entity-Relationship-Modell: Relationenmodell

Datenmodellierung im Entity-Relationship-Modell:

1. Problemrahmen transparent machen
2. Entitätsmengen ermitteln
3. Beziehungen & Kardinalitäten festlegen
4. Attribute definieren
5. Schlüssel (Primär, Sekundär) festlegen
6. Überführen in das Relationenmodell
   - Redundanz beseitigende Normalisierung,
   - Wertebereiche und
   - n:m - Beziehungen auflösen
7. Beschreibung der Integritätsbedingungen
8. Überprüfung des Modells.

ERM

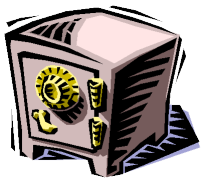
↓

Relationenmodell

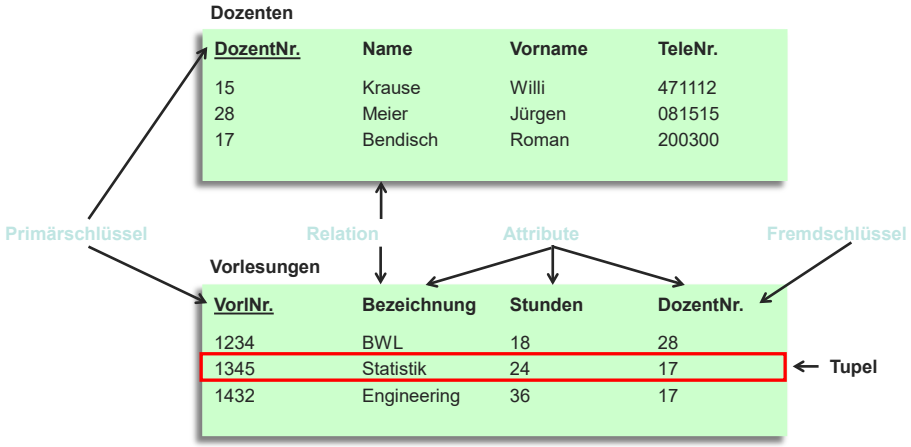
Entity-Relationship-Modell: Nach der Erstellung

Kontrolle

- Die Entitätsmengen müssen mindestens ein Attribut besitzen
- Entitätsmengen = Substantive; Beziehungstypen = Verben (Singular) beschrieben
- Attribute im Singular
- Sind zwei Entitätsmengen identisch (Kardinalität 1:1) / durch die selben Relationen mit der Umwelt verbunden / besitzen sie die selben Schlüsselelemente bzw. die selben Attribute
- Bringt jede Entitätsmenge bzw. Beziehungstyp neue Informationen in das Modell

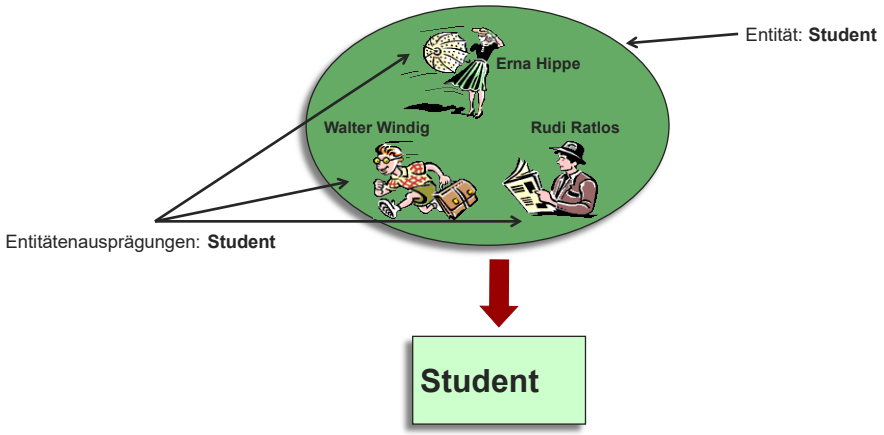


Entity-Relationship-Modell: Grundbegriffe



9.2 Entitäten

Entity-Relationship-Modell: Entitätsausprägungen



Entity-Relationship-Modell: Entitäten

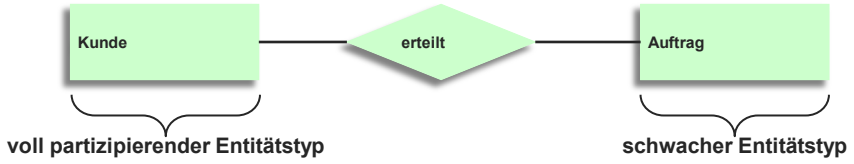
Beschreibung

- Entitätsmenge: Synonym Entity, Entitätstyp, Objekttyp, Entität
  - Eine Entitätsmenge enthält Entitäten (Ausprägungen)
- Entitätsmenge: Zusammenfassung von Entitäten mit gleichen Eigenschaften
  - Entität: Individuelle, identifizierbares Exemplar von Dingen, Personen oder Begriffen der realen oder Vorstellungswelt, wird durch Eigenschaften beschrieben.
- Substantiv (Singular)

Entity-Relationship-Modell: Schwache Entitäten

Beschreibung

- Enthalten Entitäten, die nur in **Abhängigkeit** einer **anderen** Entität bestehen können.



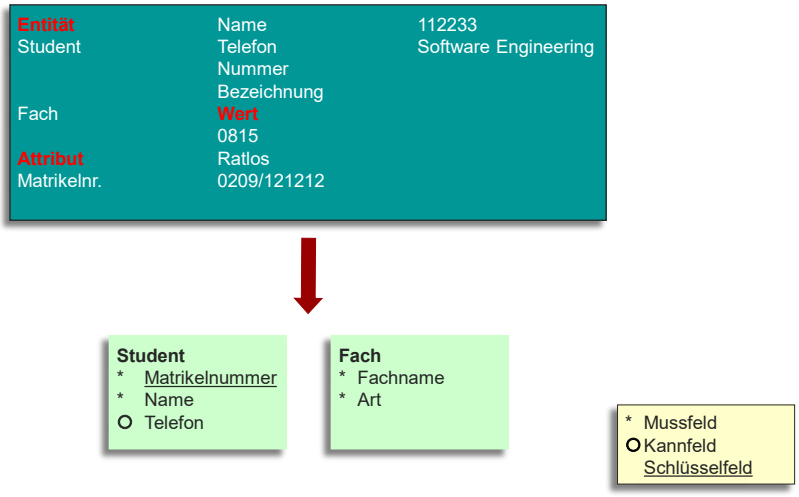
Entity-Relationship-Modell: Zirkulierende Entitäten

Beschreibung

- sind nach Möglichkeit zu vermeiden
- können zu Problemen bei der Implementierung führen

9.3 Attribute

Entity-Relationship-Modell: Attribute (Muss, Kann)



Entity-Relationship-Modell: Attribute

Beschreibung

- Attribut: Synonym Property, Eigenschaft
- Eigenschaft, die allen in einer Entitätsmenge enthaltenen Entitäten gemeinsam ist
- Substantiv (Singular)
  - Attribute werden im ERM häufig aufgrund der Übersichtlichkeit nicht dargestellt (nur Primary Keys und Foreign Keys)

Entity-Relationship-Modell: Zusammengesetzte Attribute

Beschreibung

- Zusammengesetzte Attribute werden durch eine Kombination von mehreren Attributen gebildet.
- Beispiel: $\underbrace{\text{Straße, Plz, Ort}}_{\textbf{Adresse}}$

Entity-Relationship-Modell: Mehrwertige Attribute

Beschreibung

- **Attribute** die mehr als einen Wert aus einer definierten Wertemenge (Werteliste, Ausprägungen, Domänen) annehmen können.
  - Beispiel: feste Zeitintervalle (2,4,6 Stunden)
- **Plausibilitätsprüfung** im Anwendungsprogramm

9.4 Domänen

Entity-Relationship-Modell: Domänen

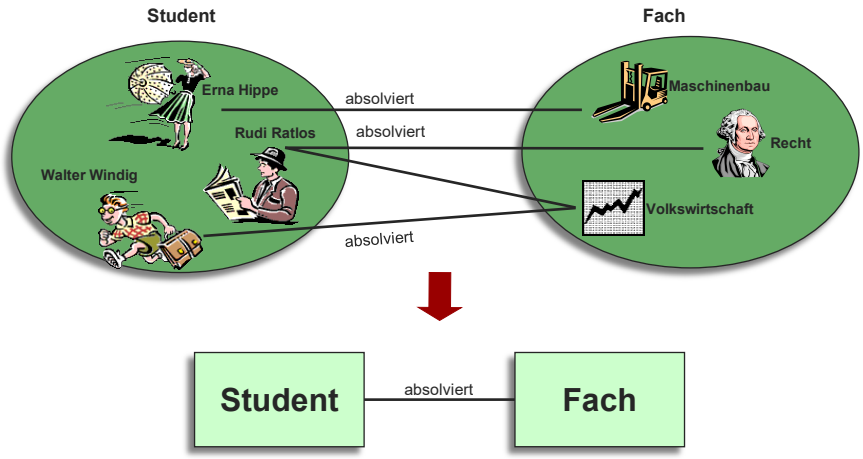
Beschreibung

- Festlegen der Wertebereiche von Attributen
- Wertebereiche sind **entitätsmengenübergreifend**
- Die Elemente eines Wertebereiches müssen **atomar** sein
- Sinn: **Beschreibung oder Beschränkung** der Attributausprägung
  - Sicherstellen der **Verknüpfbarkeit** von Entitätsmengen
  - **Absicherung** vor falschen Eingaben

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00479

9.5 Beziehungen

Entity-Relationship-Modell: Beziehungen



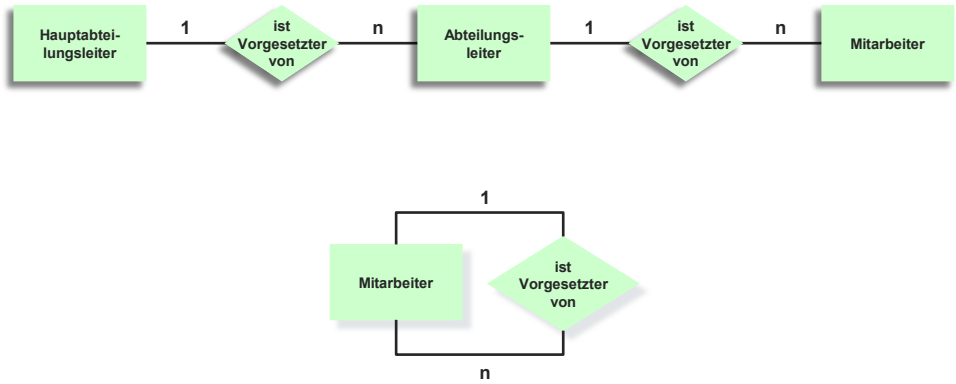
Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00480

Entity-Relationship-Modell: Beziehungen

Beschreibung

- **Beziehungstyp**: Synonym Relationstyp, Assoziation, Relationship
- **Verb (Singular)**
- Zum Beziehungstyp gehört die **Kardinalität**
- Setzt ein oder mehrere Entities zweier Entitätsmengen in **Verbindung** miteinander.

Entity-Relationship-Modell: Rekursive Beziehungen



9.6 Kardinalitäten

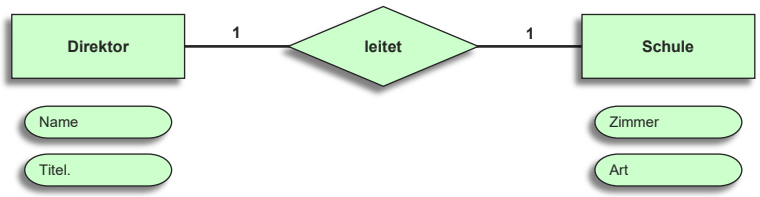
Entity-Relationship-Modell: Kardinalitäten

Beschreibung

- Kardinalität: Synonym Komplexitätsgrad
- Gibt an, mit wie vielen A-Entitäten eine B-Entität in Verbindung stehen kann
- Kombinationen
  - 1:1
  - 1:n
  - n:m

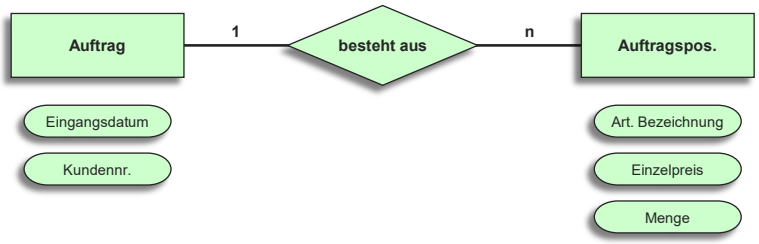
- Merke: Kardinalität immer von beiden Seiten betrachten

Entity-Relationship-Modell: 1:1

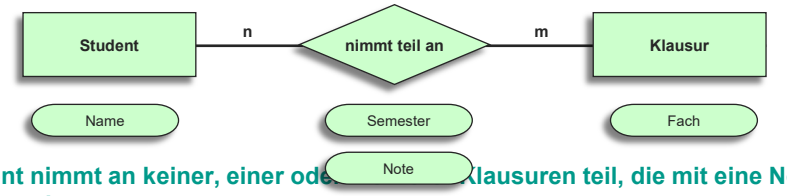


Ein Direktor leitet eine Schule.
Eine Schule wird von einem Direktor geleitet.

Entity-Relationship-Modell: 1:n



Entity-Relationship-Modell: n:m

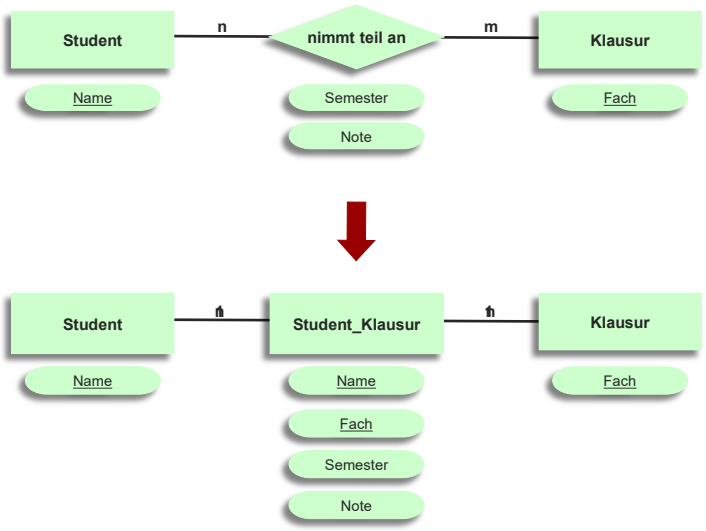


Ein Student nimmt an keiner, einer oder mehreren Klausuren teil, die mit eine Note bewertet werden.

An einer Klausur in einem Semester nehmen n Studenten teil.

1:n würde bedeuten: Ein Student nimmt an n Klausuren teil. An einer Klausur nimmt aber jeweils nur ein Student teil.

Entity-Relationship-Modell: Auflösung n:m

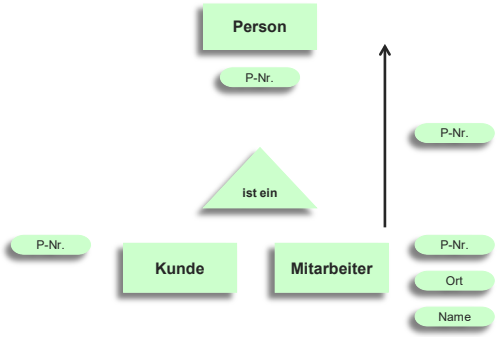


9.7 Spezielles

Entity-Relationship-Modell: Generalisierung

Beschreibung

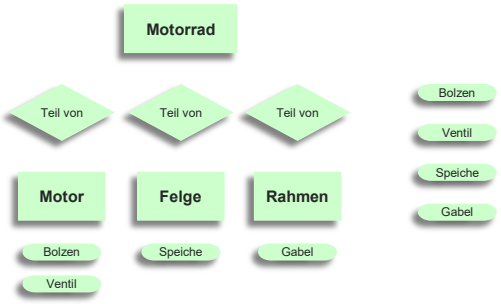
- **Generalisierung**
  Vererbung von gemeinsamen Attributen von Entitäten zu einer gemeinsamen übergeordneten Entität



Entity-Relationship-Modell: Aggregation

Beschreibung

- **Aggregation**
  eine Beziehung mit Über-/Unterordnung zwischen Entitäten.



10 Relationen Modell

10.1 Grundlagen

Relationales Datenbankmodell

Das relationale Modell

- Relationale Datenmodell, 1970 von **Edgar F. (Ted) Codd**
- weit **verbreitetste** Datenmodell
- Grundelement ist die **Tabelle**
- unterstützt einfache, mächtige **Datenbanksprache** (erstellen, manipulieren, ...)

Anforderungen an den DB-Entwurf

- **Redundanzfreiheit** (Vermeidung von Mehrfachspeicherungen)
- **Datenintegrität** (Vollständigkeit / Nutzbarkeit der Daten)
- **Datenkonsistenz** (Widerspruchsfreiheit)

Das relationale Datenmodell I/IV

- Edgar F. (Ted) Codd (1970):
  "A Relational model of Data for Large Shared Data Banks"
- Datensicht als zweidimensionale Tabelle
- Basierend auf der mathematischen Theorie der Relationen und Prädikatenlogik 1. Ordnung
- Anwendung von systematischen Methoden

- Aufbauende Verfahren
  - Normalisierungstheorie
  - Relationale Abfragesprache

Information Retrieval — P. BASHKOAKE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A small collection of operators on normal relations is put a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Keeping users and application programs free from such effects is called data independence. Future demands for such independence will arise as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, relational models of data are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchy of data, networks of data, relations, decomposition, redundancy, consistency, composition, join, universal language, predicate calculus, first order logic
CR CATEGORIES: 3.70, 3.73, 3.75, 4.22, 4.33, 4.39

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question answering systems. Levein and Maltzan [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of data independence—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of data inconsistency which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it forms the basis for a data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a firm foundation for the problems of redundancy and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of mechanisms, not the least of which is mistaking the derivation of connections for the derivation of relations (see Section 2 for the distinction between these).

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a given framework. This last point is of particular importance in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables is recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables are a kind of catalog—they contain information about the information stored in a data bank. However, the variety of data representation characteristics which can be changed without logically impacting some application programs is still quite limited. Further, the model of data with which a user or programmer is faced is still a fairly complicated one, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In most current systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to have a position which is fixed with respect to every other element (to participate in several orderings, List or member-chain existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses, for example, the sequence of the storage locations or tracks). To access a collection of records in some particular order, most existing systems require application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Volume 13 / Number 6 / June, 1970 — Communications of the ACM 377

Communications of the ACM
Volume 13, Issue 6 (June 1970)
Pages: 377-387
Year of Publication: 1970
ISSN:0001-0782
http://portal.acm.org/citation.cfm?doid=362384.362685

Entity-Relationship-Modell

Relationale Integritätsregeln

Integrität

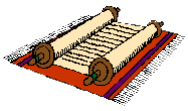
- Unversehrtheit:
  - **Keine widersprüchlichen Daten.**
  - **Nicht: Sicherstellung der Richtigkeit eingetragenen Daten.**

Entity Integrität

- Keine Datenobjekte ohne eindeutigen Schlüssel dürfen eingegeben werden.

Referenzielle Integrität

- Beziehung zwischen Entitäten sind unversehrt.
- Beispiel:
  - **PK-Datensatz wird gelöscht und abhängige Datensätze bleiben erhalten.**



Entity-Relationship-Modell

12 Regeln von Codd für relationale Datenbanksysteme

1. **Logische Informationsdarstellung erfolgt nur durch Tabellen.**
2. **Zugriff auf Attribute erfolgt über Tabellen- und Attributname.**
3. **Der Wert "Null" muss darstellbar sein.**
4. **Tabellenschemata müssen ebenfalls als Tabelle verfügbar sein.**
5. **Allumfassende (Abfrage-, Manipulations-, Definition-) Sprache muss verfügbar sein.**
6. **Views müssen permanent aktuelle gehalten werden.**
7. **Einfüge-, Lösch- und Pflegeoperationen müssen genauso mächtig sein wie die Abfragesprache.**
8. **Physische Unabhängigkeit der Anwendungen.**
9. **Logische Unabhängigkeit: Logische Einzelsichten auf die Daten sind von der logischen Gesamtheit unabhängig.**
10. **Integritätsbedingungen sind in der zu Grunde liegende Sprache zu formulieren.**
11. **Verteilte Datenbanken sind transparent darzustellen.**
12. **Integritätsbedingungen dürfen nicht umgangen werden können.**

Entity-Relationship-Modell

Tupel

Beschreibung

- Eine relationale DB besteht aus Tabellen.
- Objekte und Beziehungen der Realwelt (Entities) werden als Elemente der Relationen (Tupel, Tabellenzeilen) abgebildet.
- Entität (eine Ausprägung/Datensatz) wird als Tupel vom Grad n bezeichnet (n = Anzahl der Attribute).
- Es darf zu keinem Zeitpunkt zwei Tupel mit identischen Attributwerten geben.
- Die Reihenfolge der Tupel ist nicht definiert.

Entity-Relationship-Modell

Felder

Beschreibung

- Attribute dürfe nicht zusammengesetzt, müssen **atomar** sein.
- Jedes Attribute kann einen **Wertebereich** haben.
- Die Reihenfolge der Attribute ist nicht definiert.
- **identifizierend** (eindeutige Abgrenzung eines Objektes zu anderen Objekten) z.B. (Personalnummer).
- **klassifizierend** (Zuordnung Beziehungen zu Klassen) z.B. Zuordnung einer Kostenart (oder Mitarbeiter leitende/nicht leitend).
- **chronologische** Attribute (Zuordnung Beziehungstyp zu Zeitpunkt (Periode) z.B. Personal bearbeitet Vorgang für einen Zeitraum).
- **deskriptive** Attribute (beschreibend). Eigenschaften eines Objektes. z.B. Materielle Anforderung oder Modellierung.

Entity-Relationship-Modell

3 - Ebenen – Architektur (I/II)

Externes Schema (Sicht)

- Wird auch **Benutzersicht** genannt
- Teilausschnitt des logischen Schemas für jeweils unterschiedliche Anwender
- **Datenschutz, Datensicherheit**
- **Reduktion** der **Komplexität** für Benutzer
- **Anwendungsprogramme** und **Benutzerdaten**

Internes Schema (Sicht)

- Regelt **wie** und **wo** die Daten gespeichert werden sollen
- Zugriffspfade
- physische Datenstrukturen, Dateien
- Datenverteilung

Entity-Relationship-Modell

3 - Ebenen – Architektur (II/II)

Konzeptionelles/Logisches Schema (Sicht)

- Beschreibung sämtlicher Datenobjekte
- Darstellung der Beziehung zwischen Datenobjekten
- Definition der Integritätsbedingungen
- Unabhängig vom verwendeten Datenbanksystem
- Logische Gesamtsicht auf die Daten

Entity-Relationship-Modell

Information/Data Dictionary

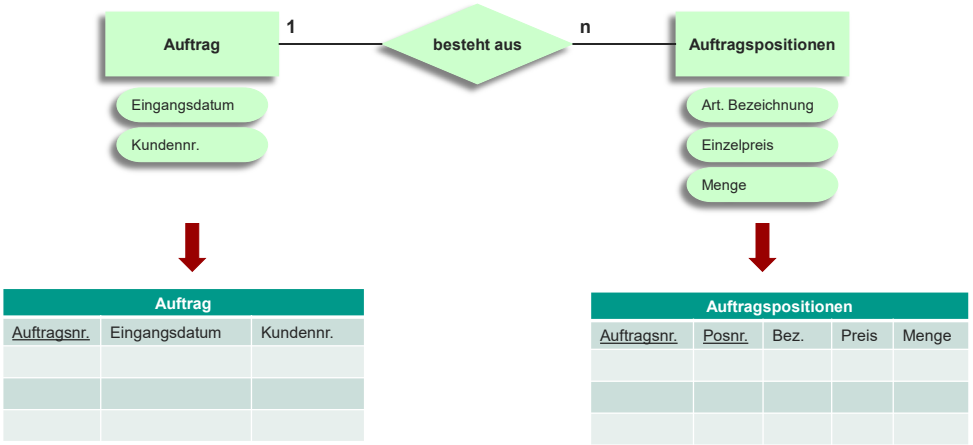
- **Strukturiertes zweckbezogenes Wissen (automatisch weiterverarbeitbar).**

- **Dokumentation der Datenstrukturen (Beziehungen zwischen den Daten), Datenfeldern (Wertebereiche, Format, Typ) und Datenverwendung (in welchen Anwendungen, Zugriffe).**
  Synonyme: Meta-Datenbanken, Respository, Datenkatalog.

Entity-Relationship-Modell

Überführung ERM/Relationenmodell (Theorie) – 1:n

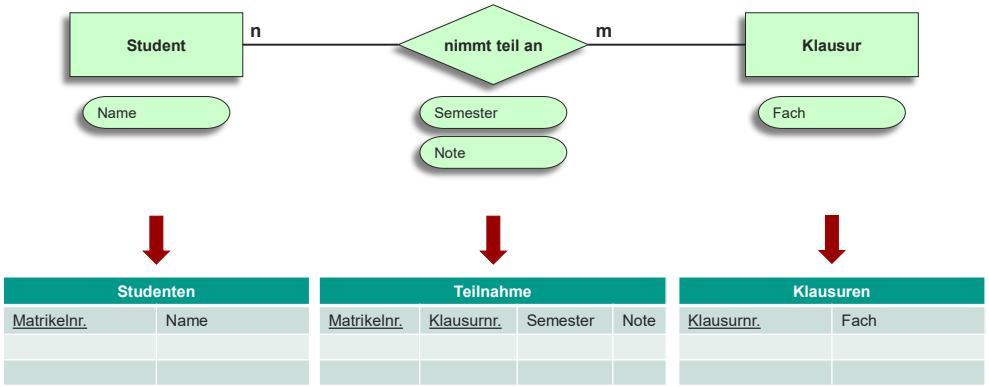
1:n



Entity-Relationship-Modell

Überführung ERM/Relationenmodell (Theorie) – n:m

n:m



10.2 Normalisierung

Entity-Relationship-Modell

Normalisierung

Als Normalisierung bezeichnet man den Prozess der Verbesserung der Datenstruktur um Qualitätskriterien (= den Normalformen) gerecht zu werden. Die Qualitätskriterien sollen sogenannte Anomalien verhindern.

Prof. Dr.-Ing. Peter Steininger
2026-SS
DBM-0001-2012-00502

Definition von Relationenschemata

Formen von Anomalien

Einfügeanomalie:

- Ein Tupel kann nur eingetragen werden, wenn alle beteiligten Schlüsselwerte vorliegen (d.h. ein neues Buch vorliegt, dass ausgeliehen wird)

Löschanomalie:

- Beim Löschen eines Tupel geht Information von Autor + Buch + Ausleiher verloren. Wünschenswert?

Änderungsanomalie:

- Bei der Änderung eines Attributs müssen mehrere Tupel geändert werden (z.B.: Änderung eines Namens), andernfalls Inkonsistenzen

Prof. Dr.-Ing. Peter Steininger
2026-SS
ITB-0001-2011-00198

Entity-Relationship-Modell
Normalisierung

Normalformen (BOYCE/CODD):

1. Normalform (1. NF)
2. Normalform (2. NF)
3. Normalform (3. NF)

Definiert höhere Qualität

Warum?

- Primärer Einsatz bei der **Datenmodellierung**
- Vermeidung unerwünschter **Abhängigkeiten**
- Eliminierung von **Redundanzen**
- Vermeidung von **Anomalien** (Delete, Insert, Update)

Entity-Relationship-Modell
Normalisierung

Funktionale Abhängigkeit

Ein Attribut oder eine Attributkombination Z wird als funktional abhängig von einem Attribut oder einer Attributkombination Y der gleichen Relation R bezeichnet, wenn zu einem Wert von Y höchstens ein Wert von Z möglich ist. Wir schreiben auch Y → Z(R) oder einfach kurz Y → Z.

Pers# ← Name

Der Name ist funktional abhängig von der Personalnummer

Entity-Relationship-Modell
Normalisierung

Y bestehe aus einem oder mehr als einem Attribut. Ist nun Z von Y, jedoch nicht von einzelnen Teilen der Attributkombination Y funktional abhängig, so ist Y voll (funktional) abhängig.

Pers# ← Name

Der Name ist voll funktional abhängig von der Personalnummer

Name, ISBN ← Lehrbuch

Das Lehrbuch ist nicht voll funktional abhängig von dem Namen und ISBN, weil das Lehrbuch nur von einem Teil (ISBN) abhängig ist.

Entity-Relationship-Modell
Normalisierung

Beispiel: Ausgangssituation

- Die Informationen, welcher Dozent unterrichtet welches Seminar fehlt.

Informationen über Studierende
Student 0815 heißt Meier. Er belegt Seminare in WI und VWL aus dem Fach Wiwi.
Müller mit der Matrikelnummer 4711 belegt Seminare in Marketing, WI (Wiwi) und alte Geschichte.
Schmidt mit der Matrikelnummer 1188 belegt ein Seminar in Trigonometrie aus dem Fach Physik.
Student 1234 Schulz hat bisher kein Seminar belegt.

Entity-Relationship-Modell
Normalisierung

Erste Strukturierung

- Meier besucht Seminare zu WI und VWL-Theorie, die beide zum Fach Wiwi gehören.

MatrNr./Name	Seminar	Fach	Professor
0815, Meier	WI, VWL	Wiwi	
4711, Müller	Marketing, WI, Alte Geschichte	Wiwi, Geschichte	
1188, Schmidt	Trigonometrie	Physik	
1234, Schulze			

Entity-Relationship-Modell
Normalisierung

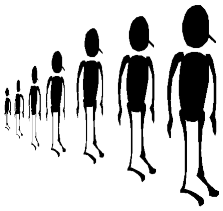
1. Normalform

Eine Relation befindet sich in der ersten Normalform, wenn die Attribute klar definiert sind und wenn alle Attribute nur atomare Werte (nicht weiter zerlegbar sind) beinhalten.)

Entity-Relationship-Modell
Normalisierung

1. Normalform

- Betrachtung einzelner Tupel (Zeilen)
- Atomare Werte schaffen
- Auflösung von Wiederholungsgruppen
- Eventuelle Erweiterung des Primärschlüssels
- Eventuell zusätzliche **Informationen** in das Modell einfügen
- Informationen **korrigieren**
- **Null-Werte** im PK werden nicht erlaubt



Entity-Relationship-Modell
Normalisierung

1. Normalform

- Die Relation befindet sich in der ersten Normalform!

MatrNr	Name	Seminar	Fach	Professor
0815	Meier	WI	Wiwi	Mittel
0815	Meier	VWL	Wiwi	Klein
4711	Müller	Marketing	Wiwi	Groß
4711	Müller	WI	Wiwi	Mittel
4711	Müller	Alte Geschichte	Geschichte	Hinz
188	Schmidt	Trigonometrie	Mathematik	Kunz

Entity-Relationship-Modell
Normalisierung

2. Normalform

Eine Relation befindet sich in der zweiten Normalform, wenn diese sich in der 1. Normalform befindet, und jedes nicht zum Schlüssel (= Primärschlüssel) gehörende Attribut von diesem voll funktional abhängig ist.

Entity-Relationship-Modell
Normalisierung

2. Normalform

- Erweiterung des Beispiels um das Datum Fachbereichsnummer (FBNr.).

MatrNr	Seminar
0815	WI
0815	VWL
4711	Marketing
4711	WI
4711	Alte Geschichte
1188	Trigonometrie

MatrNr	Name
0815	Meier
4711	Müller
1188	Schmidt
1234	Schulze

Entity-Relationship-Modell
Normalisierung

2. Normalform

- Erweiterung des Beispiels um das Datum Fachbereichsnummer (FBNr.).

Seminar	Professor	FBNr.	Fach
WI	Mittel	03	Wiwi
VWL	Klein	03	Wiwi
Marketing	Groß	03	Wiwi
Alte Geschichte	Hinz	16	Geschichte
Trigonometrie	Kunz	17	Mathematik

Entity-Relationship-Modell
Normalisierung

3. Normalform:

Eine Relation ist in dritter Normalform, wenn sie in 2. Normalform ist und es kein Attribut, welches nicht Teil des Schlüssels ist, gibt, welches transitiv vom Schlüssel abhängt.

Entity-Relationship-Modell

Normalisierung

3. Normalform

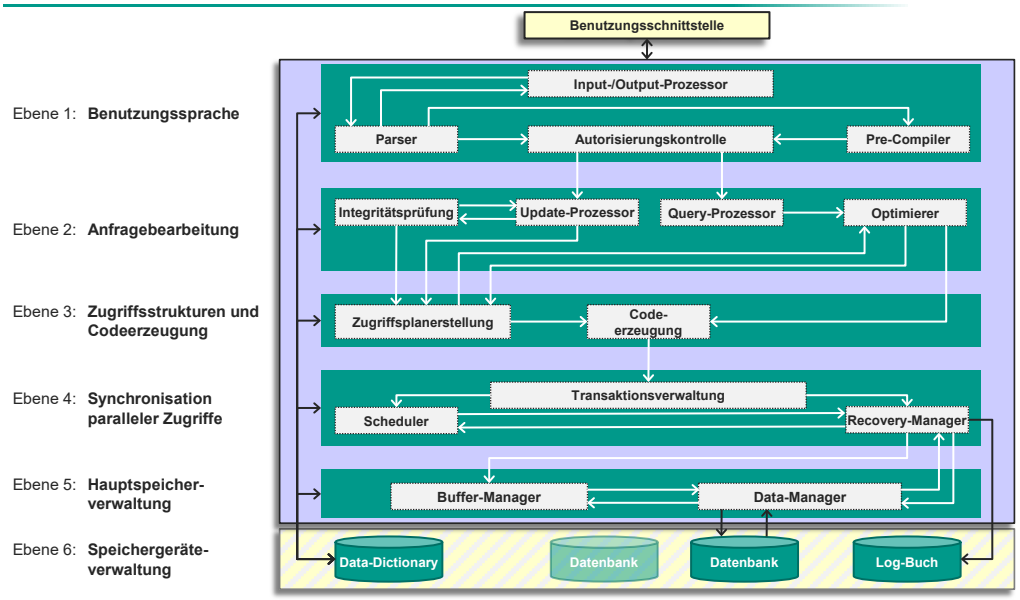
Seminar	FBNr.
VWL	03
Marketing	03
WI	03
Alte Geschichte	16
Trigonometrie	17

FBNr.	Fach
03	WiWi
16	Geschichte
17	Mathematik

11 Datenbank-Systeme (DBS)

11.1 Einführung

Datenbanksystem (DBS) – Blockdiagramm



Prof. Dr.-Ing. Peter Steininger
2026-SS

i.A.a. VOSSEN 2008
DBM-0001-2014-01035

11.1.1 Grundbegriffe

Grundbegriffe:

Definition: Datenbank/DBMS/DBS

Unter einer Datenbank (DB) versteht man einen zentral verwalteten Datenbestand, der über anwendungsunabhängige Zugriffsverfahren nutzbar gemacht wird.

Das DBMS verwaltet diesen Datenbestand und ermöglicht gleichzeitigen Zugriff mehrerer Anwendungsprogramme/Nutzer.

Das Datenbanksystem (DBS) besteht au seinem DBMS, einer DB sowie zusätzlichen Programmen, die die Bearbeitung, Verwaltung und Auswertung gespeicherter Daten vereinfachen.

Grundbegriffe: Bedeutung der Daten

Bedeutung der Daten für das Unternehmen:
"Die Daten des Unternehmens sind ein solch wichtiger und wertvoller Produktionsfaktor, dass der Verantwortliche für die Daten für so wichtig wie der Finanzchef zu erachten ist." (MARTIN 1981, S. 299)

Prof. Dr.-Ing. Peter Steininger

Quelle: MARTIN 1981

DBM-0001-2012-00565

Grundbegriffe

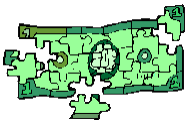
Datenbankmanagementsystem (DBMS)

Bestandteile:

- Data Dictionary
- Datenmanipulationssprache (Data Manipulation Language (DML))
- Datendefinitionssprache (Data Definition Language (DDL))
- Datenkontrollsprache (Data Control Language (DCL))

- Werkzeuge für Reports, Maske, Grafiken usw.

"Software", die zum Zugriff und Modifikation der Datenbank benötigt wird

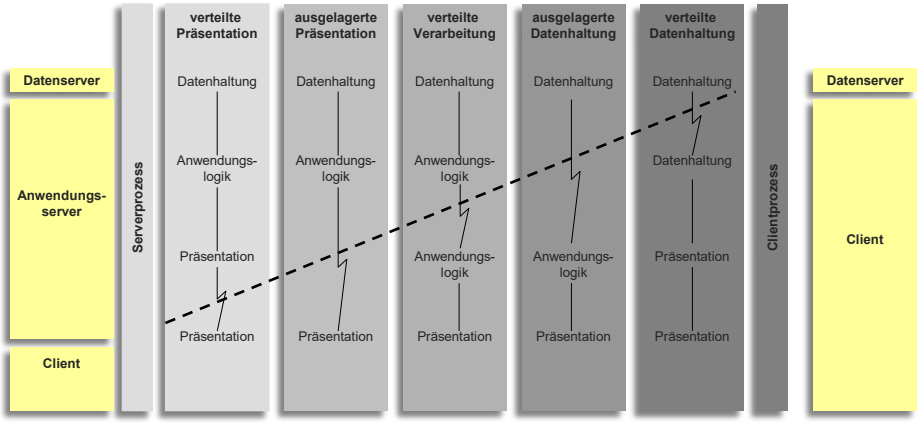


Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00566

Grundbegriff

Client-Server-Konzept



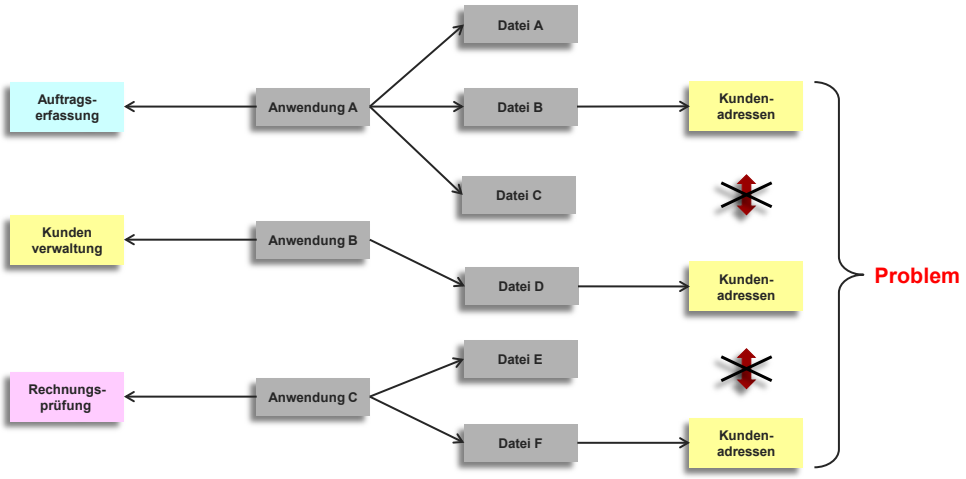
Grundbegriffe:

Definition: Constraint/Trigger

- Ändert sich ein Datensatz (oder wird er eingefügt oder gelöscht), lösen manche Datenbanken einen Alarm aus, einen sogenannten Trigger.
- Ein Constraint (engl.: Einschränkung) wird häufig in Programmiersprachenverwendet, um den Wertebereich einer Variablen einzuschränken.

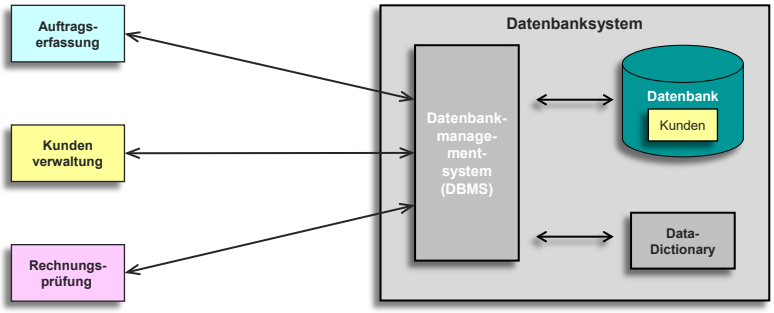
Grundbegriffe

Dateiorientierte Datenorganisation



Grundbegriffe

Datenbankorientierte Datenorganisation



Grundbegriffe

Vergleich zwischen Dateisystem und Datenbank

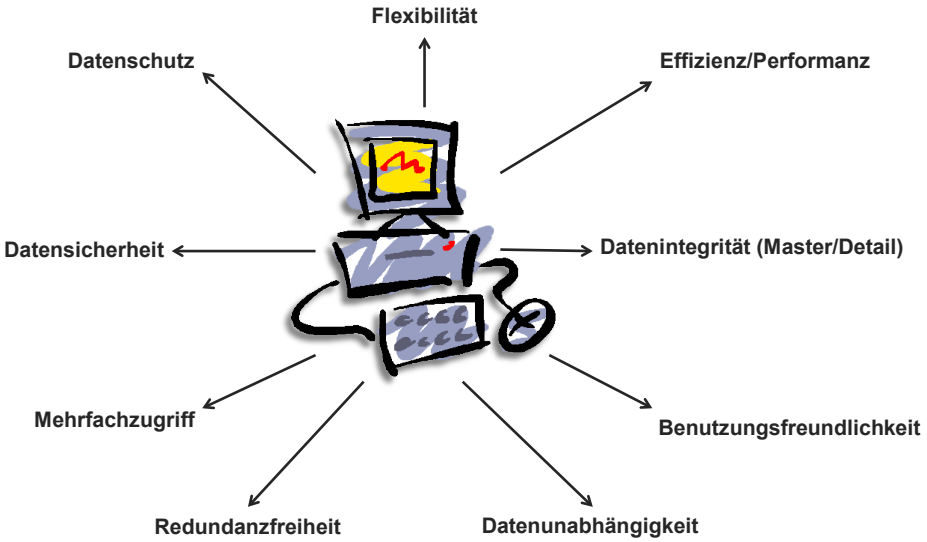
Eigenschaft	Dateisystem	Datenbanksystem (relational)
Datenunabhängigkeit	Keine, da Datenstruktur im Programm	Datenunabhängigkeit durch Projektion und Views sowie durch Trennung in externes, konzeptionelles und physisches Schema
Datensicherheit	Tagessicherung	Logging aller Datenänderungen
Zugriffsschutz	Dateiebene	Auf Satz- und Feldebene, inhaltsbezogen
Beziehungen	–	Durch Fremdschlüssel
Redundanz	hoch	gering
Konsistenz	–	Erhaltung wird durch Transaktionskonzept unterstützt
Integrität	–	Referentielle Integrität; Definition von Integritätsregeln
Aktualität	weitgehend	Ja, durch direkte Änderung
Abfragesprache	–	SQL
Schneller Zugriff	B*-Baum, Hash	Indizes
Zugriffsform	Einzelsatz-Zugriff	Mengenzugriff
Synchronisation im Mehrbenutzerbetrieb	Sperren auf Satzebene	Sperren auf Transaktionsebene
Tools	wenige	(Viele) Tools für die Anwendungserstellung

Grundbegriffe: Entwicklungsumgebung

Oftmals durch firmeninterne Gegebenheiten oder Anforderungen des Auftraggebers vorgegeben.

Produkt	Hersteller	Typ
Access	Microsoft	Relational
ADABASE	Software AG	Relational
DB2	IBM	Relational
ASE	Sybase	Relational
Informix	Informix	Netzwerk
Oracle	Oracle	Relational
UDS	SNI	Netzwerk
SQL-Server	Microsoft	Relational

Anforderungen an ein DBMS



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00574

11.1.2 Verteilung und Speicherung

Verteilung/Speicherung

Definition: Datenverteilung

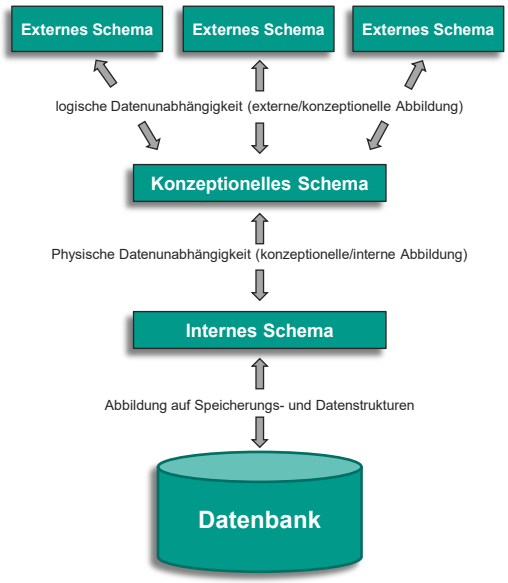
Logisch zusammenhängende Datenbestände werden physisch verteilt (Verwaltung erfolgt übergeordnet durch das Datenbank-Management-System (DBMS))

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00575

Verteilung/Speicherung

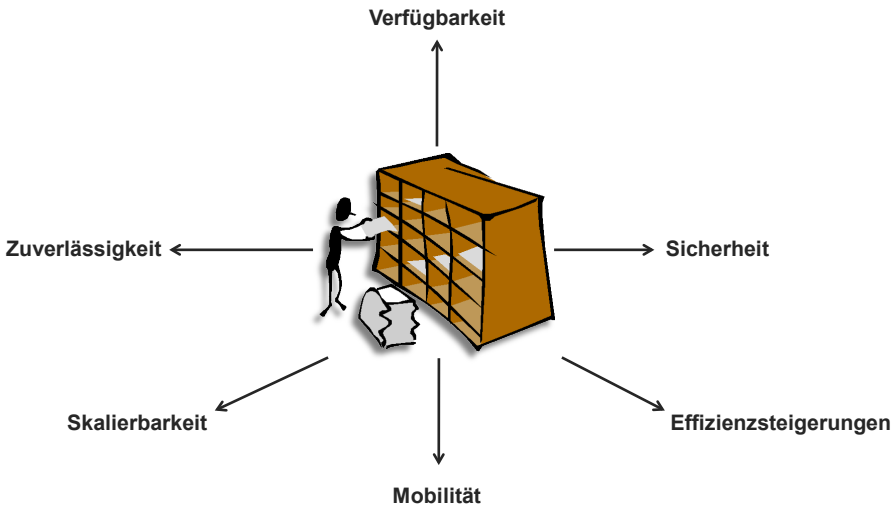
ANSI-SPARC-Dreischichtenmodell



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00576

Verteilung/Speicherung: Gründe für die Datenverteilung



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00577

Einführung: Grundbegriffe – unstrukturiert

- Word-Dokument mit Personen bezogenen Daten

```
Business:

Willi Meier, Schillerstr. 3, 023123

Müller, Hans, 80000 München, Geburtstag: 23.4.1987, Schuhverkäufer

Jansen, Paul; Handy: 0172 65 78 78; 182 m; Schuhgröße 47
Aalen

Privat:

Thomas Berg, 45886 Gelsenkirchen, 100 Euro offen
```

Einführung: Grundbegriffe – strukturiert

- Dateien:

Name	Vorname	Straße	Nr.	PLZ	Ort	Datum
Meier	Willi	Schillerstr.	3	8000	München	23.04.02
Müller	Hans	Fischerstr.	3	45884	Gelsenkirchen	23.04.87

- sequenzielle Abarbeitung der Dateien (z.B. Magnetbänder, -kassetten)
- Sprachen zur Verwaltung großer Datenbestände: COBOL, FORTRAN, PASCAL
- Applikationen speichern "eigene Daten" isoliert in "eigenen Files"
- Datenspeicherung und Zugriff auf Daten nur durch spezielle Anwendungen

Verteilung/Speicherung

Strukturierte Datenspeicherung in einer Datenbank

Anwendung und Daten sind voneinander unabhängig

Möglichkeiten eines DBMS

- anwendungsneutralen Zugriff auf Daten
- Regelung der Zugriffsrechte
- Mehrbenutzerfähigkeit über Netzwerke
- Vermeidung von Redundanz, Gewährleistung von Konsistenz
- Recovery-Konzept (Schutz vor Datenverlust-Datensicherheit)

DBMS als neue Schicht zwischen Anwendung und Daten

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00580

11.1.3 Historie

Historie

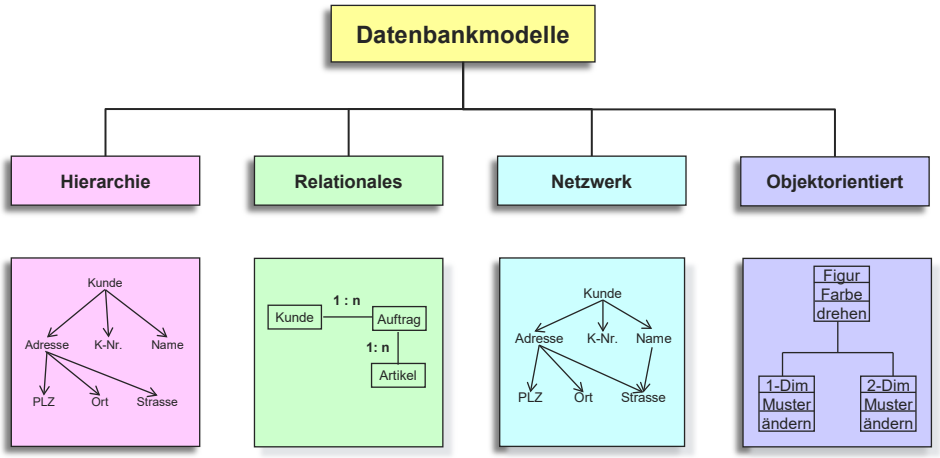
Jahrzehnt	Technik	Zugriffsart
1950	Lochkarte(n)/Magnetband	sequenziell
1960	Festplatte	random access
1970	Netzwerk, Hierarchisches DBMS	random access
1980	Relationale DBMS	random access
1990	Postrelationale DBMS	random access
	• objektrelationale (ORDBMS)	
	• objektorientierte (OODBMS)	

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00581

11.1.4 Datenbankmodelle

Datenbankmodelle: Übersicht



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00582

Datenbankmodelle: Hierarchisches Datenbankmodell

Baumartige Verknüpfung verschiedener Sätze

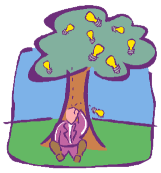
- Jeder Entitytyp hat nur **einen Vorgänger** (Ausnahme 1. Ebene)
- Jeder Entitytyp kann **mehrere Nachfolger** haben
- Beispiel: IMS von IBM

Vorteile

- Effiziente "**computergerechte**" Datenorganisation
- sehr schnell

Nachteile

- Benutzer muss die Datenstruktur genau kennen
- **Redundanzen** sind möglich (da keine n:m Beziehung möglich ist)



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00583

Datenbankmodelle: Hierarchisches Datenbankmodell

Beispiel:

Eine Abteilung kann mehrere Mitarbeiter haben. Jeder Mitarbeiter kann in mehrere Projekten arbeiten.

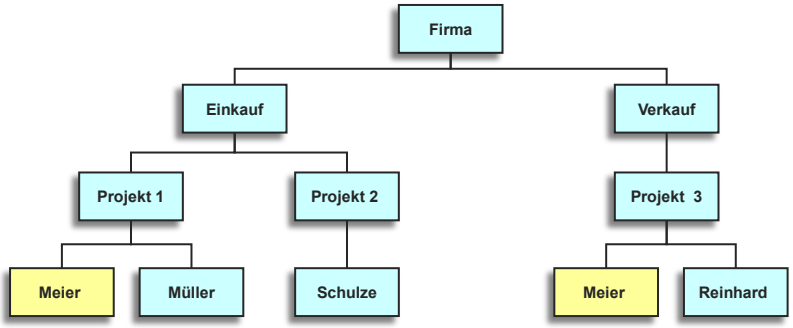
In jedem Projekt arbeiten ein oder mehrere Mitarbeiter.

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00584

Datenbankmodelle: Hierarchisches Datenbankmodell

Beispiel:



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00585

Datenbankmodelle: Netzwerk Datenbankmodell

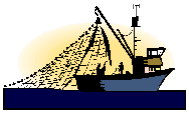
Entitytypen können mehrere Vorgänger und Nachfolger haben
Es kann mehrere Wege zu einer Informationen geben.

Vorteile:

- Relativ **einfache** Modellierung, da n:m Beziehung erlaubt ist
- **Komplexe** Strukturen können abgebildet werden

Nachteile:

- **Unübersichtlichkeit**
- Benutzer muss die Datenstruktur genau kennen.

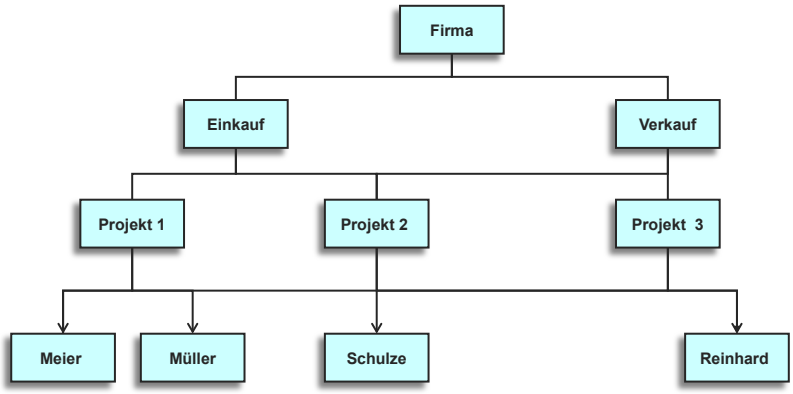


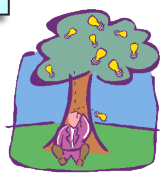
Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00586

Datenbankmodelle: Netzwerk Datenbankmodell

Beispiel:





Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00587

Datenbankmodelle: Objektorientiertes Datenbankmodell

Integration objektorientierter Eigenschaften in DBS

- Kapselung,
- Vererbung und
- Polymorphismus

Kenntnisse der Objektorientierung erforderlich

Darstellung komplexer Strukturen

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00588

11.1.5 Rechte und Sicherheit

Rechte/Sicherheit: Vergabe von Benutzerrechten

Rechte:

- **Insert** (einfügen von Datensätzen → erstmaliges anlegen)
- **Update** (aktualisieren von Datensätzen → überschreiben)
- **Delete** (löschen von Datensätzen → unwiderrufliches entfernen von Datensätzen)
- **Select** (anzeigen von Datensätzen)

Möglichkeiten:

- Datenbankgesteuert
- Tabellengesteuert (Individualsoftware)
- Rollenbasiert

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00589

Rechte/Sicherheit: Vergabe von Benutzerrechten

Datenbankgesteuert:

```
grant insert, delete, select, update Benutzer on benutzerrechte to SH00002;
```

Tabellengesteuert:

```
create table benutzerrechte (
    benutzer varchar(20) constraint primary key,
    s number(1) not null,
    u number(1),
    d number(1),
    i number(1));
```

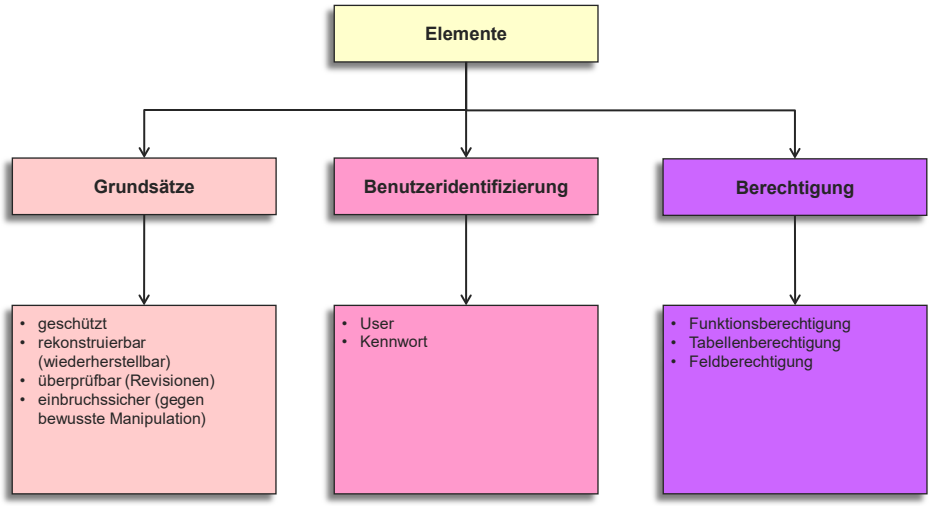
Benutzer	S	U	D	I
SH0002	1	1	0	1
WW0034	1	0	0	0
LE00123	1	0	0	1
HH0111	1	1	1	1

Rechte/Sicherheit: Vergabe von Benutzerrechten

Profilgesteuert

Plant Maintenance Transaktionsliste	Transaktion	Stammdaten	Leiter	Vorarbeiter	Arbeiter ohne Auftragsbearbeitung	Arbeiter mit Auftragsbearbeitung
Plant Maintenance						
Technische Objekte						
Lokation						
Anlegen	IL01	X				
Ändern	IL02	X				
Anzeige	IL03	X	X	X	X	X

Rechte/Sicherheit: Datensicherung



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00592

11.2 Relationenmodell

11.2.1 Locking und Locks

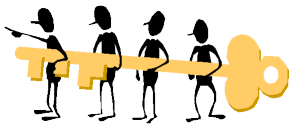
Locks: Mehrfachzugriff/Locks

Überschneidungen:

- mehrere Benutzer greifen zum gleichen **Zeitpunkt** auf Datenbestände zu (2 Reisebüromitarbeiter wollen den letzten freien Platz buchen)
- Datenbankobjekte können (kurzzeitig) für eine Transaktion **gesperrt** werden

Lock-Möglichkeiten

- ein **Feld**
- einen **Datensatz**
- komplette **Tabelle** (Administration)
- komplette **Datenbank** (Administration)



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00659

Locks

Arten:

- **Exclusive Locks**: Auf das gesperrte Datenbankobjekt ist kein Zugriff möglich
- **Shared Locks**: Lesen erlaubt, keine Manipulation

Logfiles:

- Alle durchgeführten Änderungen werden in einem **Logfile** protokolliert.
- Im Falle eines Ausfalls kann der verloren gegangene Inhalt **rekonstruiert** werden
- existiert eine Ausgangszustand der DB zum Zeitpunkt **t**, so kann mit Hilfe der Logfiles der Zustand der DB zu jedem beliebigen Zeitpunkt **t + x** wieder hergestellt werden

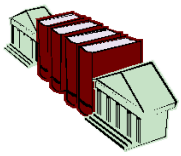
Locks: Erstellung Sicherheitskonzept

- **Views als Sicherheitsmedium einrichten (Achtung Performance)**
- **Zugriffsregelung konzipieren und vergeben**
- **Sicherheitsverfahren konzipieren**
- **Backup konzipieren**
- **Berechnung der Datenbankmengen inkl. Puffer durchführen**
- **Eingabefehler durch Plausibilitätsprüfungen vermeiden**

11.2.2 Datenorganisation

Datenorganisation: Sequenzielle Speicher

- **Beschreibung:**
  - Datensätze werden unmittelbar nacheinander abgespeichert
  - können nur in der gespeicherten Folge verarbeitet werden
  - Verwendung : z. B. Backup–Medium
- **Suchverfahren:**
  - sequenziell



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00662

Datenorganisation: Adressierbare Speicher

- **Beschreibung:**
  - jeder beliebige Datensatz mit Kenntnis der Adresse sofort
  - gelesen,
  - geändert,
  - gelöscht oder
  - eingefügt
  - Magnetplatte, Diskette, CD, RAM, ZIP-Medien
- **Suchverfahren:**
  - sequenziell
  - indizierte Organisation
  - gestreute Organisation

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00663

Datenorganisation: Indizierte Organisation

- **Unter einem Index versteht man in der Informationsverarbeitung eine Hilfsdatei oder -tabelle, deren Datensätze neben den Schlüsseln der Hauptdatei die Adressen der zu diesem Schlüsseln gehörenden Datensätze beinhalten.**

Datenorganisation: Indizierte Organisation

Beispiel:

Hauptdatei

Adresse	Name	Name	Ort
1	Otto	Gross	Essen
2	Boris	Klein	Bochum
3	…	…	…
4	…	…	…

Indexdatei

Name	Adresse
Klein	2
…	…
Gross	1
...	…

Überlaufbereich

Der Adressverweis stellt die Verbindung zwischen Haupt- und Indexdatei dar.
Zugriff von Index auf Hauptdatei

- unsortierter Index (gesamter Index muss durchsucht werden)
- sortierter Index (Suchprozess wird verkürzt)

Datenorganisation: Physisch sortierter Index

Inhalt:

- Reihenfolge der Speicherung entspricht der Sortierreihenfolge
- auf dem Datenspeicher wird der Index physisch gespeichert
- muss ständig aktualisiert werden

Einfügen von Datensätzen:

- Überlaufbereich, damit Daten nicht immer neu sortiert werden müssen
- wird ein Datensatz gesucht, 1. suche im oberen Teil per Index 2. sequenzielle Suche im Überlaufbereich
- regelmäßige Reorgansisation muss durchgeführt werden, damit alle Daten wieder sortiert sind

Datenorganisation: Physisch sortierter Index

- **Binäres Suchen:**
  - Schnell, effizient, Voraussetzung = Daten sind sortiert
  - Halbieren der Datenmenge bis man das Ergebnis gefunden hat (Vergleich der Adressenschlüssel)

  - Beispiel:
    - Suche der Zahl 297 (Zahlen von 1 - 1023)
    - 1 - 1023 (halbieren) → 512 Suchwert größer oder kleiner?
    - 0 - 512 (halbieren) → 256 Suchwert größer oder kleiner?
    - 256 - 512 (halbieren) → 384 Suchwert größer oder kleiner?
    - …

Datenorganisation: Physisch sortierter Index

- **M-Wege Suchen:**
  - Alternative zum Binären Suchen
  - der sortierte Datenbestand wird in Blöcke eingeteilt
  - der Suchschlüssel wird zuerst mit dem letzten Element des erste Blockes verglichen
    - Ist der Suchschlüssel größer → Übergang um nächsten Block
    - Ist der Suchschlüssel kleiner wird durch das binäre Suchverfahren weitergesucht

Datenorganisation: Physisch sortierter Index

M-Wege Suchen:

Indexdatei

	rel. Adresse	Name	Adressverweis
Block 1	1	Becker	4
	2	Hansen	6
	3	Müller	1
	4	Rehagel	8
Block 2	5		2
			7

Hauptdatei

Nr.	Adressverweis	Autor	Titel
344	4	Hansen	Hallo
239	6	Müller	Tag
234	1	Becker	Abend
245	8	Rehagel	Morgen
222	2	Schön	Nacht

Datenorganisation: Physisch sortierter Index

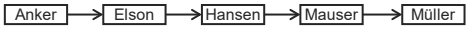
Logisch versus Physische sortierter Index:

- Problem physisch sortierte Datenbestände → **Reorganisation**
- seltene Reorganisation → Suchzeiten **verlängern** sich, durch das **langsame** Suchen im Überlaufbereich
- regelmäßige Reorganisation → Reorganisationsläufe brauchen **erhebliche** Rechnerzeiten
- Logisch sortierter Index → **keine Reorganisation** ist notwendig
  - **Kette**
  - **Baum**

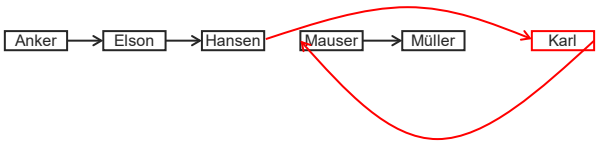
Datenorganisation: Physisch sortierter Index

Kette:

- in jedem Datensatz wird die Adresse (Pointer) des in der Sortierreihenfolge nachfolgenden Satzes gespeichert
- **Anker** = Der Zeiger, der auf den ersten Datensatz zeigt.
- Beispiel (Ausgangspunkt):



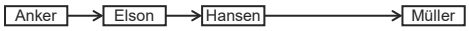
- **Einfügen**: Neuer Datensatz wird ans Ende des physischen Datenbestandes geschrieben. Es wird nur der Zeiger verändert.



Datenorganisation: Physisch sortierter Index

Kette:

- Es werden nur die Zeigeradressen geändert. Speicherplatz des gelöschten Datensatzes wird als überschreibbar gekennzeichnet. Hier wird "Mauser" gelöscht.



- **Vorteil:** einfacher Änderungsdienst
- **Nachteil:** nur sequenzielles Suchen möglich (von Adresse zu Adresse) z. B. bei Arbeitsspeichern

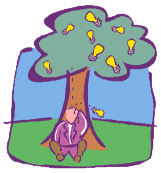
Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00672

Datenorganisation: Logisch sortierter Index

Baum:

- Eine Baumstruktur besteht aus einer Menge von **Knoten** und **Kanten**. Jede Kante zeigt auf einen Knoten. Der Baum weist drei Eigenschaften auf :
  - **Es gibt genau einen Knoten, der keinen Vorgänger hat → Wurzel (Root).**
  - **Jeder Knoten, außer der Wurzel, hat genau einen unmittelbaren Vorgänger.**
  - **Zu jedem Nichtwurzelknoten gibt es genau einen Weg von der Wurzel zum Knoten.**

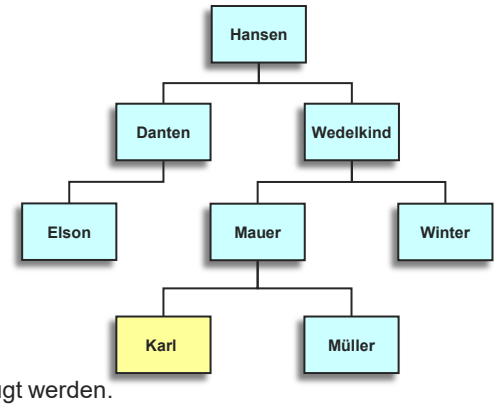


Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00673

Datenorganisation: Logisch sortierter Index

Baum:



- **Einfügen:**
  Karl sollte eingefügt werden.
  Karl größer als Hansen. Ja → nach rechts.
  Karl größer als Wedelkind. Nein → nach links.
  Karl größer als Mauer. Nein → nach links.
  Einfügen des Knotens an der logischen Position des Baumes

Datenorganisation: Logisch sortierter Index

Baum:

- **Löschen**:
  Einfaches streichen von Müller (wenn der zu löschende Knoten keinen oder einen Nachfolger hat). Wedelkind löschen → Vertauschoperation. Der nächst kleinere oder nächst größere Knoten ersetzt den zu löschenden Knoten → aufwendige Vertauschoperation
- **Suchen:**
  Einfache Vergleichsoperationen
- Es kann sein, dass die Bäume auf beiden Seiten sehr unausgeglichen sind. Es kommt auf die eingefügten Datensätze an. → Daten müssen doch reorganisiert werden.
- Besser als Kette, schlechter als binäres und M–Wege–Suchen
- Einsatz in DB–Systemen

Datenorganisation: Gestreute Organisation (I/V)

- **Bei der gestreuten Organisation wird direkt aus dem jeweiligen Schlüssel die Adresse des zugehörigen Datensatzes errechnet.**

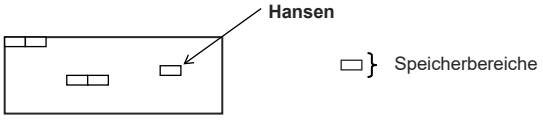
Datenorganisation: Gestreute Organisation (II/V)

- **Probleme:**
  - Welche Funktion soll für die Berechnung verwendet werden ?
  - Da die Anzahl der verfügbaren Speicherplätze i. d. R. geringer ist als die der möglichen Schlüssel, muss eine Funktion gewählt werden, die eine Doppelbelegung (**Kollision**) einer Adresse zulässt. Die Art der Behandlung derartiger Kollisionen beeinflusst die Zugriffsdauer wesentlich.

- **Hash-Funktion:**
  - Hier wird aus alphabetischen, numerischen oder alphanumerischen Schlüsseln eine Menge von Adressen **berechnet**.
  - Bei Kollisionen → sequenziell hintereinander schreiben (Überlaufbereich)

Datenorganisation: Gestreute Organisation (III/V)

- **Beispiel:**



- **10-stelliges alphabetisches Schlüsselfeld mit dem Inhalt "Hansen".**
- **Zeichen 8 Bit = 256 Werte (0-255)**

Datenorganisation: Gestreute Organisation (IV/V)

Beispiel:

	1	2	3	4	5	6	7	8	9	10
	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Buchstabe	H	A	N	S	E	N				
EBCDIC	100	193	213	226	197	312	64	64	64	64
Wert	$100*2^9$	$193*2^8$	$213*2^7$	$226*2^6$	$197*2^5$	$312*2^4$	$64*2^3$	$64*2^2$	$64*2^1$	$64*2^0$

$100 * 2^9 + 193 * 2^8 + \ldots + 64 * 2^0 =$
$100 * 2569 + 193 * 2568 + \ldots + 64 * 2560 =$
948048930278549498118208 → Schlüssel "Hansen"

- **Schlüsseltransformationsfunktion (engl. Hash function)**:
  riesige Anzahl von möglichen Schlüsseln auf ein akzeptable Menge von Adressen abbilden. Rest der Division 948048930278549498118208 / 1117 = 403 [Adresse]

Datenorganisation: Gestreute Organisation (V/V)

DML:

- **Einfügen** = sehr einfach (über Formel)
- **Löschen** = sehr einfach (löschen aus Speicherraum)
- **Suchen** = sehr einfach (effizient) sofern keine Kollisionen entstehen z. B. Arbeitsspeicher

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00680

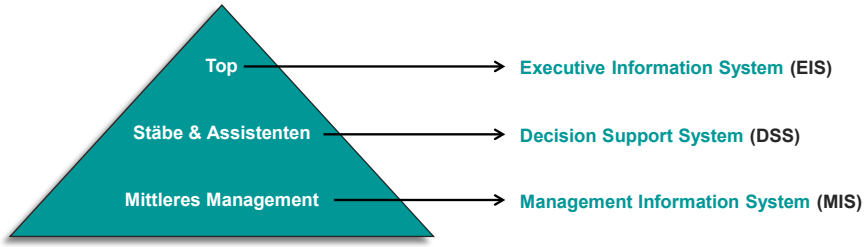
11.3 Online Transaction Processing (OLTP), Data Warehouse (DWH) und Data Mining

11.3.1 Management Information System (MIS)

Managementunterstützungssysteme

Managementunterstützungssysteme:

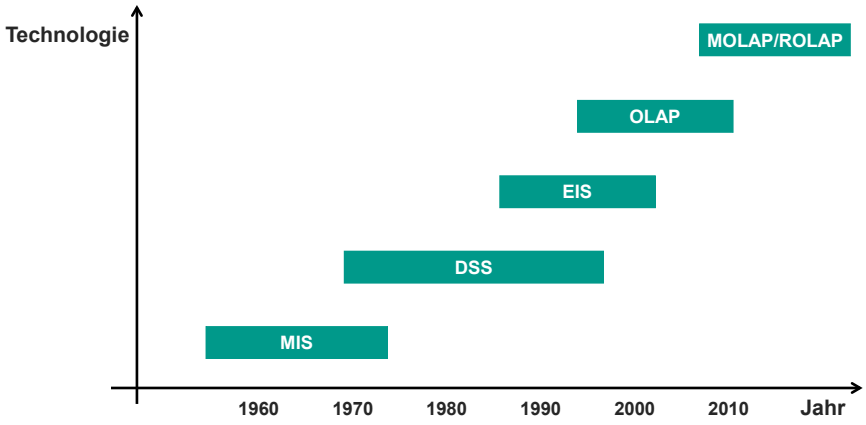
- sind die Summe aller Informationssysteme zur **Unterstützung** sämtlicher Managementebenen.



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00681

Entstehung



MIS

Beschreibung:

- sind Informationssysteme, die Mitarbeiter bei de Erledigung ihrer täglichen anfallenden Fachaufgaben unterstützen (**Routineaufgaben**).
- Orientierung auf operative, meist interne Informationen (harte Informationen)
- **Berichtsorientierung** mit Standardberechnungen (z.B. Summen-, Durchschnittsbildung)
- Basis für **DSS** und **EIS**

DSS

Beschreibung:

- computergestütztes Informationssysteme, das einen oder mehrere **Entscheidungsträger** bei semistrukturierten Entscheidungsprozessen **unterstützten**
- Algorithmische Orientierung
- Modell-und Methodendatenbanken
- Sensitivitätsbetrachtungen/Zielwertberechnungen

DSS

Methoden im DSS:

- **Finanzmathematische Methoden:**
  Beispiel: Kapitalwertmethode, Annuitätenberechnungen, Interne Zinsfußmethode, Diskontierungs-/ Inflationsberechnungen.
- **Statistische Methode:**
  Risiko-Analyse (Mittelwertbetrachtungen (Modus, Median usw.), Regressionsanalysen/Prognose (Fore-casting)
- **What-if-Methoden:**
  Sensitivitätsbetrachtungen, Zielwertanalysen, Signifikanztests
- **Umfassende Modelle:**
  Modelle für multidimensionale Konsolidierungen, Modelle zur Investitionsanalyse, Modelle für Vertriebs-/Marketingauswertungen

EIS

Beschreibung:

- computergestütztes Informationssystem, das es einem **Top Manager** (Executive) in Abhängigkeit von seinen individuellen kritischen Erfolgsfaktoren (critical success faktor (CSF)) erlaubt, auf einfache Art und Weise auf externe und interne Daten zuzugreifen

11.3.2 Data Warehouse (DWH)

Data Warehouse

Ist ein umfassendes Konzept zur Entscheidungsunterstützung von Mitarbeitern aller Bereiche und Ebenen. Kern ist eine integrierte DB mit entscheidungsrelevanten Informationen über die Geschäftsfelder, die aus den operativen DB und externen Quellen bedarfsgerecht destilliert wird. Der direkte Zugriff wird den Endbenutzern durch einen Informationskatalog erleichtert, der über die Inhalte, Formate und Auswertungsmöglichkeiten des Data Warehouse Auskunft gibt. Eine Dritte wesentliche Komponente sind die Softwarewerkzeuge und Anwendungsprogramme, mit denen Daten des Data Warehouse abgefragt, transformiert, analysiert und präsentiert werden können

Data Mining (Neuronale Netze)

Bezeichnet man die softwaregestützte Ermittlung bisher unbekannter Zusammenhänge, Muster und Trends in sehr großen DB. Dabei kann der Benutzer bestimmte Ziele vorgeben, für die das System angemessene Beurteilungskriterien ableitet und damit die Objekte de DB analysiert. Oder das System teilt automatisch auf eine vage Frage hin eine gegebene Menge von Objekten in Cluster auf.

Data Warehouse

Merkmale:

- **Subjektorientierung:**
  Operative Daten werden von Applikationen weiterverarbeitet. Diapositive Daten werden vom Manager zur Unterstützung seiner Entscheidungen benötigt.
- **Integration:**
  Data-Warehouse nutzt Daten aus verschiedenen Quellen und formalisiert Sie für die Auswertung (Report) neu.
- **Zeitbezogenheit:**
  Operative Daten entsprechen aktuellem Stand. Diapositive Anwendungen benötigen auch Daten aus der Vergangenheit
- **Beständigkeit:**
  Operative Daten werden laufend durch die Anwendungen überschrieben. Data-Warehouse liest nur Daten

Data Warehouse

Gründe:

- **Wie laufen unsere Kerngeschäfte?**
  - Unternehmensfakten (-wissen),
  - bestehen Möglichkeiten, auf die man reagieren soll?
- **Ich brauche diese Information sofort?**
  Herkömmliche Berichterstattung funktioniert nicht; Ad-hoc Abfragen gibt es nicht in herkömmlichen Operativsystemen
- **Wie sah die Entwicklung vor einem Jahr aus?**
  - historische Daten,
  - subjektorientierte Daten
- **Warum habe ich keine Zugriff auf diese Informationen?**
  keine gewöhnliche DB, verschiedene Informationsquellen

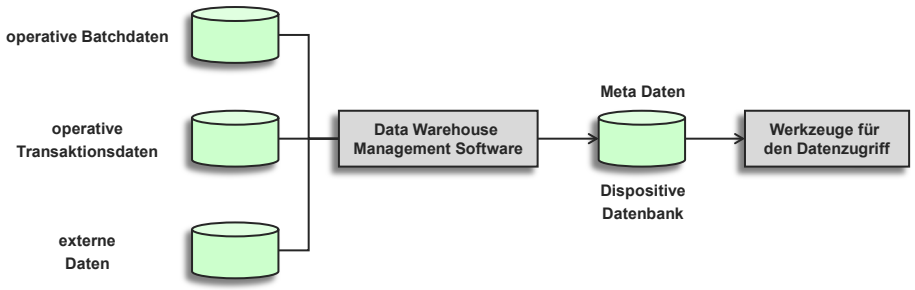
Data Warehouse

Ziele:

- Unternehmensweite **konsistente** Daten = eine Datenquelle für Berichte und Analysen
- **Bessere Performance** bei ad-hoc-Anfragen = Trennung zwischen OLTP-Tabellen und Tabellen des Abfragesystems
- **Einfache Zugriffsmöglichkeiten** durch den Endbenutzer = leistungsfähige Werkzeuge für das Reporting, Zugriff ohne vorherige Einmischung der EDV-Abteilung

Data Warehouse: Architektur (I/II)

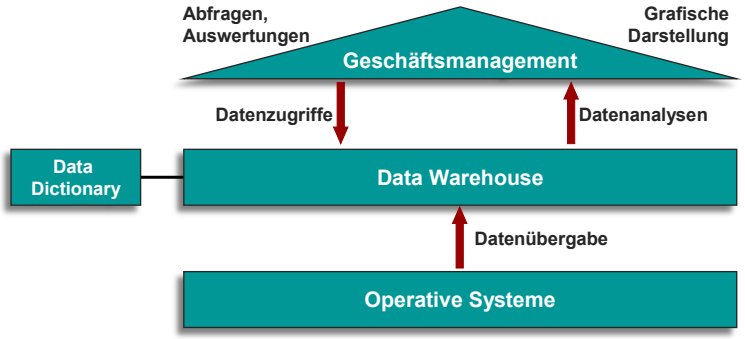
Werkzeuge formen die operativen Daten um, speisen sie in eine dispositive Datenbank ein und stellen sie dem Endanwender bereit. Die benötigten Definitionen sind in den Meta-Daten festgehalten.



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00692

Data Warehouse: Architektur (II/II)



Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00693

11.3.3 Online Analytical Processing (OLAP) und Online Transaction Processing (OLTP)

Vergleich

	Operative Datenbanken	Informative Datenbanken
	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Transaktionsvolumen	hohes Volumen	mittleres bis niedriges Volumen
Antwortzeit	sehr schnell, im Sekundenbereich	normal, bis mehrere Minuten
Update	hohe Frequenz, permanent	niedrige Frequenz
Betrachtungsperiode	aktuelle Periode	Vergangenheit – Zukunft
Umfang	anwendungsintern	anwendungsübergreifend
Aktivitäten	operativ, detailliert	analytisch, taktisch
Abfragen	vorhersehbar, periodisch	unvorhersehbar, "ad hoc"
Niveau der Daten	detailliert	aggregiert, aufbereitet
Verarbeitungseinheit	Datensatz, eindimensional	Matrizen (Array), multidimensional
Zeithorizont	1-3 Monate	mehrere Jahre bis zu Jahrzehnten
Datenaktualität	permanent gegeben	nur nach Update gegeben

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00695

OLAP

Kennzeichen:

- OLAP – Regeln
- Historisch Weiterentwicklung von EIS
- Mehrdimensionale Sichten auf die Daten
- Flexibles bewegen durch unterschiedliche Verdichtungsebenen
  - **Slice & Dice** = Benutzer können einen bestimmten Ausschnitt der im Hypercube aggregierten Daten entlang jeder vorgesehenen Dimension "schneiden" oder "drehen", um so einen Überblick aus verschiedenen Blickwinkeln zu erhalten.
  - **Drill–Down** = Ein Drill Down ermöglicht dem Benutzer die Auffächerung (verfeinern) aggregierter Informationen, um mehr Details zu erfahren. → Drill–up.

Prof. Dr.-Ing. Peter Steininger

DBM-0001-2012-00696

OLAP

Kennzeichen:

- Wenig Benutzer
- Zugriff auf historische Daten notwendig
- Anwendungsbeispiel: Kaufverhalten eines Kunden eines Einzelhandelkaufhauses

OLTP

Kennzeichen:

- Viele Benutzer die vordefinierte SQL-Anweisungen ausführen
- Kurze Transaktionen
- Anwendungsbeispiel: Auftragsverwaltungs- oder Reservierungssystem