DATENSTRUKTUREN

3 Abstraktionsebenen (geordnet nach Abstraktionsgrad):

- **Mathematikebene**: **WAS** wird berechnet (**Spezifikation**)
  - **Algorithmische Begriffszuordnung**: **Funktion**, **Relation**
  - **Datenbegriffszuordnung**: **abstrakter Datentyp**, **Datentyp** (**Algebra**)
- **Algorithmische Ebene**: **WIE** wird berechnet (**allgemeine, rechnerunabhängige Implementation**)
  - **Algorithmische Begriffszuordnung**: **ALGORITHMUS** (**Struktogramm**)
  - **Datenbegriffszuordnung**: **DATENSTRUKTUR**
- **Programmierungsebene**: konkrete Darstellungsebene (**konkrete Implementation**)
  - **Algorithmische Begriffszuordnung**: **Prozedur**, **Funktion**
  - **Datenbegriffszuordnung**: **Klasse**, **Typ**, **Modul**

Algebra:

- **System**; **mathematisches Gebilde**
- Eine **Menge von Objekten** (Zahlen, Zeichenketten), die sog. **Trägermenge** mit **zugehörigen Operationen** (=**Verknüpfungen**), die auf dieser Menge erklärt sind (T,OP)
- Zwei verknüpfte Elemente der Objektmenge ergeben ein **Element der Objektmenge** oder einer **neuen Menge** (=Ergebnismenge); man erzeugt eine **Struktur** auf der Menge!
- Zur Verständlichmachung von **allgemeinen**, **realisierungsunabhängigen Gesetzmässigkeiten**
- **Definiert / spezifiziert Datentyp auf hoher Abstraktionsebene** (=mathematische Abstraktionsebene); sie ist **Modell eines Datentyps**

→ Begriffe: zweistellige (T,OP), dreistellige Algebra (T,OP,OP)

Halbgruppe:

- **Algebra mit assoziativer Operation** (z.B. Zeichenverknüpfung getrennt betrachtet)

Monoide:

- **Halbgruppe um neutrales Einselement erweitert** (z.B. Leerzeichen oder Zahl 0)

Geordnete Menge:

- **Ordnungsrelation einer Trägermenge** (Vergleich </>, z.B. Telefonbuch)
- Geltende Gesetze: **Reflexivität**, **Antisymmetrie**, **Transitivität**

Verband:

- **Geordnete Menge erweitert um logische Operatoren & (logisches UND) und | (logisches ODER)** (z.B. Netzplantechnik, optimierte Ablaufplanung)

- Geltende Gesetze: **Kommutativität**, **Assoziativität**, **Idempotenz**, **Absorption**

Gruppe:
- **Monoid mit inversem Element für jedes Element der Trägermenge** (in der Informatik unbedeutend)

Sorten:
- **Wertearten der Trägermenge** (natürliche Zahlen, Wahrheitswerte)

Universelle Algebra:
- **Ergebnismenge = Trägermenge**

Mehrsortige (heterogene) Algebra:
- **Mehrere Mengen / Objektklassen beteiligt**

Warum Spezifikationssprache?
- Angabe von Trägermenge und Operatorenmenge (T,OP) reicht nicht aus → Charakterisierung der Algebra erfordert weitere Beschreibungselemente

2 Teile der Algebra in der Spezifikationssprache:
- **Syntaktischer Teil**: sog. **Signatur**; Bezeichnungen (T,OP), Anzahl und Art Objekte als Argumente für Operationen, Art der Ergebnismenge; enthält Terminologie **algebra**, **sorts**, **operators**
- **Semantischer Teil**: Definition (T,OP) sowie deren Wirkung; enthält Terminologie **sets**, **variables**, **functions**, **end**

Rolle der Spezifikationssprache bei der Algebra / ADT Darstellung:
- Mit wenig vorgegebenen Sprachelementen und einem syntaktischen Regelwerk exakte und widerspruchsfreie Aussagen formulieren
- Jeder Sprachmächtige kann diese Aussagen ohne Störung empfangen
- **Abstraktion trennt das Wesentliche vom Unwesentlichen**
- Logische Axiome schliessen Kandidaten von Algebrenmodellen aus
- Wichtigste Eigenschaft der Spezifikation: **Eindeutigkeit**

Datentyp:
- Zusammenfassung einer Menge von Werten und Operationen, die auf diese Werte angewendet werden dürfen
- **Hierarchische Algebra** D=(R,VD,OP) mit einer **Repräsentantenmenge R** vom Datentyp D (=ausgezeichnete Trägermenge) sowie einem **abstrakten Fehlerelement error**, **VD als Menge vorausgesetzter Datentypen** und **OP als Menge von Operationen**
- Hierarchisch: baut auf VD auf, um komplexere DT zu schaffen
- Vererbung: neuer DT erbt sämtliche Eigenschaften der VD (OOP-Bezug)
- Wird **durch Datenstruktur implementiert / realisiert**

Abgrenzung Algebra / Datentyp:
- DT ist eine **spezielle Algebra**; jedem DT kann eine Algebra **als Modell unterlegt** werden

Abstrakter Datentyp:

- **Äquivalenzklasse aller Datentypen mit gleicher Signatur und gleichem äusseren Verhalten**
- Äusseres Verhalten: **Menge der Wechselwirkungen der Operatoren** auf die innere Datenstruktur (=**Axiome**; definieren nur die Wirkung der OP, nicht aber die algorithmische Definition der OP selbst)
- Menge der Axiome: bilden Äquivalenzrelation einer Klasse äquivalenter DT
- Entsteht durch **realisierungsunabhängiger Definition** der Eigenschaften des DT (=implementationsunabhängig=erlaubt unterschiedliche Implementierungen)
- Trägermenge wird mittels **Konstruktor** definiert
- **Kapselung**: verborgene innere Datenstruktur im Implementierungsmodul des ADT versteckt; Nutzerzugriff über definierte Schnittstellen (=Signatur der Algebra)
- Vorteile: polymorphe DT können spezifiziert werden, präzise formale Beschreibung
- Spezifikationssprache Substitution: **algebra** → **adt**; **functions** → **axioms**

Zusammenhang ADT / Algebra:
Eine Algebra heisst Modell eines ADT, falls

- Algebra die gleiche Signatur wie der ADT hat und
- Operationen der Algebra den Axiomen des ADT genügen

Unterschied DT / ADT:

DT	ADT
	Schwerpunkt auf Eigenschaften, die Operationen und Wertebereiche besitzen
	Unabhängig von einer konkreten **Programmiersprache**
	Obermenge der DT

Warum ist DT ein so grundlegendes Konzept bei der Modellierung von Prozessen?

- In der GPM werden komplexe reale Abläufe abstrahiert (=relevante Objekte und Prozessschritte abgeleitet
- Wesentliches von Unwesentlichem trennen
- Abstrakte Gebilde können relativ einfach in Rechneranlagen abgebildet werden

Datentypen in Programmiersprachen:

- **Konkrete DT**
  - **Atomare DT**:
    - **Standard DT**:
      - **Ordinale DT**: es gibt exakte Vorgänger und Nachfolger, aufzählbare Ordnung
        - **Integer DT**
        - **Char DT**
        - **Boole DT**
        - **Aufzählungs DT**
        - **Zeiger DT**
      - **Real DT**: nicht aufzählbar, keine Ober- und Untergrenze, zwischen 2 reellen Zahlen gibt es unendlich weitere
    - **Prozedurtyp**
  - **Dynamische DT**: Instanzen werden durch Konstruktor erzeugt, die zu jedem beliebigen Zeitpunkt verändert oder zerstört werden können (new / destroy); werden über Zeiger realisiert
    - **Lineare DT**: Komponenten total geordnet
      - **Datei**: **Sequenz**, **Folge**
      - **Liste**: Knoten einer Liste=meist Verbund und Verweis auf nächsten/vorigen Knoten → (einfach/doppelt) verkettete Listen; letztes Listenelement durch Selbstreferenz oder Nullzeiger realisiert; werden im Heap gespeichert (hohe Performance)
      - **Prioritätsschlange** (P-Queue) / **Kellerspeicher** (Stack)
    - **Nichtlineare DT**: Ordnungsrelation ist eine Halbordnung (= es gibt unvergleichbare Komponenten); Speicherung im Heap
      - **Baum**: jeder Knoten hat zwei oder keinen Nachfolger
      - **Graph**: beliebig viele Wurzelelemente, Zugriff „Kanten“
  - **Aggregierte statische DT**: Zusammenfassung (Aggregierungen) von Objekten atomarer DT zu neuen DT; statisch bedeutet ein existierender Gültigkeitsbereich (Grösse und Anzahl der Elemente stehen fest); besitzen Typkonstruktor zur Deklaration
    - **Reihe**: **Feld**, **Array**; geordnete Reihung der Objekte eines einzigen DTs im Speicher hintereinander; feste Elementzahl; Zugriff über Index; Verschieben von Elementen sehr inperformant
    - **Verbund (Satz)**: **Record**, **Struct**; Reihung unterschiedlicher logisch zusammenhängender DT Objekte unterschiedlicher Länge als Satz
    - **Menge**: endliche Zusammenfassung von Objekten eines DT ohne vorgeschriebene Reihenfolge; keine gleichen Elemente möglich; kann auch leer sein
- **Abstrakte DT**

Dynamische Datentypen:

- **Werden aus Instanzen atomarer und aggregierter DT nutzerkontrolliert zur Laufzeit eines Programms aufgebaut und verändert**
- Zur **Strukturierung** werden Instanzen des **Zeiger** DT verwendet
- **Speicher** muss **dynamisch allokiert** werden (new)
- Beim Löschen müssen Zeigerreferenzen umgesetzt werden, der Speicher wird dann freigegeben (destroy)
- Dynamische Objekte werden im **Heap** abgelegt (dieser wächst dynamisch)

Unterscheidung statische / dynamische DT:

Statische DT	Dynamische DT
Feste Speicherausdehnung	Entstehung zu beliebigem Zeitpunkt
Fester Platz im Speicher	Speicherplatz nicht vorhersehbar
Konstruktor verbindet Instanz (Platz) und Variablenname	Dynamische Adressen nicht an vorher deklarierte feste Namen bindbar

Datei:

- **Linearer dynamischer DT**
- Wird im Gegensatz zur Liste nicht im Heap, sondern auf **externen Datenträgern** verwaltet (Permanenz)
- Zugriff ist von der Organisationsform abhängig:
  - **Sequentielle Organisationsform**:
    - Auf Basis des **DT Sequenz**: Einzelfolgen hintereinander
    - Zugriff nur durch Überlesen der davorliegenden Information
    - Ungeeignet für Manipulation und Aufsuchen von Daten
    - Anwendung zur Absicherung von Datenbeständen, meist auf Magnetbänder
  - **Indexsequentielle Organisationsform**:
    - Auf Basis des **DT Dictionary**:
    - Jeder Datensatz erhält Schlüsselinformation zum Auffinden
    - **Sequentielle Speicherung** auf externen Datenträgern
    - Direkter Zugriff: Physische Adresse (Zylinder etc.) wird mit Schlüsselinformation im Index festgehalten
    - Indexbaum im Hauptspeicher geladen
  - **Untergliederte bibliotheksorientierte Organisationsform**:
    - Deklaration der Datei als **Zeiger**; Zugriff über „Directory“

Definition Datenstruktur:

- **Implementation eines (abstrakten) Datentyps auf der algorithmischen Ebene** (mit Hilfe einer Programmiersprache), d.h. die Operatoren des Datentyps werden durch Algorithmen realisiert, und für die Objekte der Trägermengen (der vorliegenden Modell-Algebra) wird eine interne Repräsentation festgelegt.

Unterschied Datenstruktur zu DT/ADT:

- durch spezielle Realisierungen lassen sich unterschiedliche Datenstrukturen festlegen, die dem ADT entsprechen, aber unterschiedliche Ressourcen in Anspruch nehmen

Datenmodell:
- **Beschreibung der Realität über Daten und ihre Beziehungen**
- Umfasst eine **Menge von Regeln zur Bildung von Datenstrukturen** und eine **Menge von Operationen**, die über einer Datenbasis definiert werden

3 Ziele eines Datenmodells:
1. Abbildung von (realen) Informationsprozessen
2. Abstrahierung von der Realität (komplexe reale Probleme → **konzeptionelles Modell** → Grundlage für logisches Datenmodell)
3. Strukturierung von Informationszusammenhängen

Warum Abstraktion?
- Nicht alle Probleme relevant: „Schaffung eines Ausschnitt der realen Welt" (=konzeptionelles Modell)
- Rechenanlagen haben endliche Kapazitäten

5 Stufen/Ebenen des Abstraktionsprozess der Datenmodellierung:
- **Reale Welt**
- **Informationen über Teile der realen Welt**: Konkreter Sachverhalt (konzeptionelles Modell)
- **Logische Datensysteme**: Verallgemeinerung, Datenmodellebene (Datenstrukturentwurf für Geschäftsprozesse)
- **Physische Datensysteme**: Filesystemaspekte
- **Computer**: binäre, magnetisierte Information

→ *Analysierung* → *Strukturierung* → *Modellierung* → *Implementierung*

3 Arten Datenmodelle:
- **Semantische Datenmodelle**:
  - **Synonym Infologische DM**
  - Realitätsausschnitt-Beschreibung als **statisches Modell** mit Elementen und Beziehungen
  - Hochabstrakte **Informationsstrukturierung** unter den Aspekten **Bedeutungsgehalt** und **Informationsinhalt**
- **Logische Datenmodelle**:
  - **Synonym Formale / Datalogische DM**
  - Beschreibung in **Abhängigkeit** von der **Informationsorganisation** einer **Klasse von DBMS**
  - **Konkrete Form**, da DBMS i.d.R. immer nur ein logisches DM zulässt, 3 Modellierungsformen:
    - **Hierarchisches DM**
    - **Netzwerkorientiertes DM**
    - **Relationales DM**
  - Entwurf basiert auf semantischem DM
- **Unternehmensdatenmodelle**:
  - Basieren auf semantischen DM
  - Integration **sämtlicher unternehmensrelevanter Daten** (Gesamtarchitektur) sowie der **Datenschnittstellen** auf **konzeptioneller Ebene**

- **Rechner- und DBMS-unabhängig**

Problem Realwelt → semantisches DM → logisches DM → DBMS

Entity Relation Ship Modell (ERM):
- Beschreibt Entitäten und -mengen sowie deren Beziehungen und –mengen
- Grafische Darstellung der semantischen Datenmodellierung

Entität, engl. entity:
- **Abgrenzbares / identifizierbares Objekt der realen Welt** (Ding, Gegenstand)
- **Physisch existierendes** Objekt oder auch eine **gedankliche Abstraktion**
- Individuelle identifizierbare „Exemplare"; auch Beziehungen

Entitätsmenge: 
- Zusammenfassung von **Entitäten mit gleichen oder ähnlichen Merkmalen**
- Besitzt **eindeutigen Namen**

Entitätstyp:
- In der Literatur häufig mit Entitätsmenge gleichgesetzt
- Beinhaltet gewählte **Bezeichnung** und **Strukturbeschreibung**

Attribute / Entitätseigenschaften: 
- Spezielle Attributswerte von Entitäten
- Anzahl und Art der Attribute sind Bestimmungsgrösse der Entitätsmenge

Eigenschaftswerte:
- Spezielle Ausprägung eines Attributs der Attributsmenge einer Entitätsmenge
- Eindeutige Beschreibung

Primärschlüssel, engl. key:
- Zur **eindeutigen Identifikation** einer Entität
- Einziges Attribut oder Attributskombination mit **Minimaleigenschaft** (Wegfall des Schlüssels → verlorene Eindeutigkeit)
- Wahl des am Besten geeigneten Schlüsselkandidat

Beziehung, engl. Relationship:
- 3 grundsätzliche Arten von Beziehungstypen: 1:1, 1:n, n:m

Beziehungsmengen, engl. relationship sets: 
- Zusammenfassung von hinsichtlich Art und beteiligten Entitätsmengen übereinstimmenden Beziehungen
- Gekennzeichnet durch **Entitätsmengen in Relation** und **Attributen**
- Attribute sind mindestens immer die identifizierenden Merkmale der beteiligten Entitätsmengen der relevanten Beziehung

Attributierte Beziehungen:
- Beschreiben typische Eigenschaften von Beziehungen

Kardinalität:

- Festlegung des **Beziehungsgrad** (=Beziehungstypen)
- Wieviele Elemente der einen Entitätsmenge mit wie vielen Elementen der anderen Entitätsmenge in Beziehung stehen
- **3 grundsätzliche Arten** von Beziehungstypen, sog. **Assoziationstypen**: **1:1**, **1:n**, **m:n** (=multiple, keine eindeutige Zuordnung)
- bzgl. der Verbindlichkeit Unterscheidung der Kardinalität:
  - **obligatorische Beziehungen**: jedes Element hat mindestens einen Beziehungspartner
  - **konditionale Beziehungen**: nicht jedes Element muss einen Partner haben, Kennzeichnung durch „**c**“

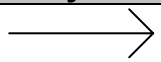
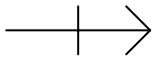
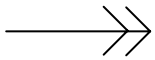
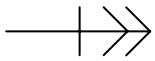
Komplexität:

- **minimale und maximale Anzahl** der Beziehungen jeder der beiden Entitätsmengen, die in Beziehung stehen
- Angabe von Grenzwerten – durch Komma getrennt – in runden Klammern
- Beide Richtungen müssen betrachtet werden

Zusammenhang Komplexitätsgrad und Assoziationstyp:

- **Ableitung** ist möglich
- **Kanten werden revers beschriftet**!

Symbolische Schreibweise der Komplexitätsgrade:

Komplexitätsgrad	Symbol
1: einfache Komplexität	
c: konditionale komplexität	
m: multiple Komplexität	
mc: multipel-konditionale Komplexität	

Konditionale Beziehungen:

- (3 grundsätzliche Beziehungstypen nennt man **obligatorisch**)
- Kandidaten einer Entitätsmenge ohne Beziehung haben eine **wahlweise** oder konditionale Beziehung
- Bezeichnung: Kardinalität c (bei m oder n vorangesetzt, bei 1 ersetzt)

Warum Einführung von Konstruktionsoperatoren:

- Klassisches ERM reicht für komplexe Realität nicht aus
- Zur Systematisierung semantischer Information

Konstruktionsoperator Klassifizierung:

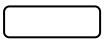
- OOP Konzept
- **Entitäten mit gleichen Attributen** zusammenfassen
- Bildung von **Entitätsmengen**

Konstruktionsoperator Spezialisierung/Generalisierung: 

- OOP Konzept
- Aus übergeordneten Klassen **Subklassen** bilden und revers
- Bedingungen:
  - Existenz als **Teilmenge**
  - Besitzt **gleichen Primärschlüssel**
  - **Attributsvererbung**
  - evtl. **weitere Attribute** vorhanden
  - **Partizipation an allen Relationen**
- Formulierung als **„is-a" Beziehung**
  - **Disjunkte Entitätsmengen**: keine Entität kommt in mehr als einer Teilmenge vor
  - **Überlappende Entitätsmengen**: Entitäten sind mehreren Teilmengen zugeordnet

Konstruktionsoperator Aggregation:

- OOP Konzept
- Beziehungsmenge wird mit zugehörigen Entitätsmengen **auf höherer Ebene zu neuen Objektklassen** zusammengefasst
- → so können **Beziehungsmengen Beziehungen eingehen**

Konstruktionsoperator Gruppierung: 

- aus Elementen einer Entitätsmenge diverse Gruppen bilden, wenn einzugehende Beziehungen nur für Teilgruppen gelten sollen
- Formulierung als „**Gruppe setzt sich zusammen aus**"

Rekursive Beziehungen: 

- Darstellung **existenzieller Abhängigkeiten**: Objekt einer Eintitätsmenge ist vom Vorhandensein eines Objektes einer anderen Entitätsmenge abhängig und hat ohne dieses keine Existenzberechtigung
- Sog. **Master-Detail** Beziehung

Vorgang ERM Visualisierung:

- (Ist-Zustand-Analyse)
- Bestimmung der Entitäts- und Beziehungsmengen
- Strukturierung der Entitäts- und Beziehungsmengen im ERD
- Bestimmung der Beziehungstypen
- Zuordnung der Attribute und Bestimmung der Schlüsselattribute

Hierarchisches Datenmodell:

- Entitätstypen und Beziehungen bilden **Baumstruktur**
- Zuordnung **Entität → Entitästypen**, **Set → Set-Typ**
- **Set Mitglieder** sind **hierarchisch** geordnet
- **Wurzel**: Existenz Baum-Entität ohne Vorgänger
- **Blätter**: Nachfolge-Entitäten
- **Zweige**: enden mit Blättern ohne Nachfolger
- **Abfragetechnik Zugriffspfade**: Entitäten sind von Wurzel aus über **gerichteten Graph** erreichbar
- 2 grafische Darstellungsebenen: **Typebene**, **Ausprägungsebene**

Vor- und Nachteile des hierarchischen DM:

- Vorteile: einfache Struktur, effiziente Verarbeitung (auch sequentiell)
- Nachteile: Entitätstypen nur in einem Baum, Zugriff über nur einen Weg; Redundanz; m:n Beziehung nur über Auflösung

Netzwerkdatenmodell:

- Entitätstypen und Beziehungen bilden zusammen **beliebig gerichtete Graphen**
- Zuordnung **Entität → Entitästypen**, **Set → Set-Typ**
- **Set Mitglieder** sind **geordnet**
- **Wurzel**: Existenz Baum-Entität ohne Vorgänger
- **Blätter**: Nachfolge-Entitäten
- **Zugriff**: von der Wurzel aus über jeden, nicht notwendig gerichteten Weg
- Begriffe:
  - **FIELD**: Attribut
  - **RECORD**: Entität
  - **SET**: Grundelement der Beziehungsbeschreibung für logische und physische Zugriffe, jedes SET besitzt 2 RECORD-Typen: OWNER (Parent) und Member (Child)
- **Navigation**: Datenobjektsuche auf genauem Weg (SET-Folge)

Vor- und Nachteile des Netzwerk DM:

- Vorteile: einfache Modellierung, Abbildungsregeln für 1:m und m:n Beziehungen; prozedurale Navigation (sehr gute Performance)
- Nachteile: komplex durch unbedingte Kenntnis von Zugriffspfaden und Datenstrukturen; eingeschränkte Datenunabhängigkeit; Entitätsbeziehungen bei DB Entwurf fix und schwer modifizierbar

Relationales Datenmodell RDM:

- **Verzicht auf grafische Darstellung** ggü. Netzwerk und hierarchischem DM
- Abbildung:
  - Auf **Mengentheorie basierende mathematische Schreibweise**: **Relationsname ($\underline{A_{PS}}$, A_1, A_2, ... A_n)**
  - **Tabellarische Darstellungsform**: zweidimensional; **Anzahl Tupel=Mächtigkeit**; **Anzahl Attribute=Grad**
- **Tupelorientierung** (=Zeilenorientierung)
- Informationsrepräsentation als **Relation**
- **3 Ebenen**: Typebene, Ausprägungsebene, Operationale Ebene

Unterschied Tabelle / Relation: Tupel kann in Tabelle mehrfach vorkommen → eine Tabelle ist im mathematischen Sinn daher keine Menge

Relation:

- Benannte Menge von n-Tupeln, wobei ein n-Tupel eine Anordnung von n atomaren, d.h. nicht weiter zerlegbaren Attributen ist

RDM Merkmale:

- Existenz einer **Menge von Relationen unterschiedlichen Grades über den Attributswerten**
- Jede Relation besitzt einen **Primärschlüssel**
- **Keine Tupel-Duplikate** einer Relation
- **Tupel-Ordnung / Reihenfolge belanglos**
- **Attributsreihenfolge egal**, da eindeutiger Name
- **Attributswerte** in Relationen sind **elementar** (=eine Menge von Werten ist unzulässig)
- **Primärschlüssel identifiziert eindeutig**
- **Relationen** sind **untereinander gleichberechtigt**
- **Relationen** sind **zeitlich veränderlich** (Einfügen, Löschen)

Vor- und Nachteile RDM:

- Vorteile: einfach, da nur ein Strukturelement: Relation; nicht-prozedurale Manipulation (kein WIE, sondern WAS ist das Ziel)
- Nachteil: Performance schlecht durch Tabellenverknüpfung

Darstellung von Beziehungen im Relationalen Datenmodell:

- Abbildung durch Entitäten und Relationen
- Primärschlüssel organisieren Verbindung (**join**)
- Primärschlüssel bilden in Beziehungsrelation sog. Fremdschlüssel (foreign key)
- Fremdschlüssel in Beziehungsrelation verweisen auf jeweiligen Primärschlüssel in der Basisrelation

Warum Transformation des Datenbankentwurf ERM → RDB?

- ERM als semantisches Datenmodell sehr abstrakt
- Beim ERM stehen **allgemeine Strukturen** im Vordergrund
- ERM gibt keine Auskunft über **Implementierung**
- RDM als logisches Datenmodell trifft **Annäherung an RDBMS (Datenbankschema)**

Transformationsregeln ERM → RDB:

Kardinalität	Obligatorisch=Verbindlich	Zahl der Relation
1:1	Ausprägung beider Entitytypen **→ Primärschlüssel wählbar**	1
1:c	Ausprägung eines Entitätstyps **→ Fremdschlüssel auf 1-Seite**	2 → workaround**:** NULL
c:c	Ausprägung keines Entitytyps **→ Hilfsrelation mit 2 FS**	3
1:m , c:m	Ausprägung der Entitytypen der m-Seite **→ FS auf m-Seite**	2
1:mc , c:mc	Ausprägung keines Entitytyps **→ Hilfsrelation mit 2 FS**	3
m:m , m:mc , mc:mc	Ausprägung keines Entitytyps **→ Hilfsrelation mit 2 FS**	3

→ Voraussetzung der Transformationsregeln:

- **Nur zweistellige Beziehungen** erlaubt (=zwischen 2 Entitätstypen)

- Generalisierung/Spezialisierung sowie Aggregationen **müssen aufgelöst** werden

Geschäftsprozess:
- In Unternehmen ablaufende Prozesse (entlang der Wertschöpfungskette), die an den Unternehmenszielen und kritischen Erfolgsfaktoren KEF ausgerichtet sind
- Eine **inhaltlich abgeschlossene**, **zeitlich und sachlogische Folge von Funktionen** (Aktivitäten), **die auf die Bearbeitung eines betriebswirtschaftlich relevanten Objektes ausgerichtet sind**
- **Objekt** kann **materiell** oder **informativ** sein
- Abgeschlossenheit: Auslösung durch externes Ereignis; Beendigung durch Ereignis
- Kennzeichen von GP: **Komplexität** (Unterschiedlichkeit der beteiligten Elemente, **Kompliziertheit** (Vielzahl der beteiligten Elemente)

Bedeutung der GP Modellierung:
- **Wirtschaftliche Bedeutung**:
  - Sicherung der Produktqualität
  - Verbesserung der Wirtschaftlichkeit
  - Schnelles und flexibles Reagieren auf Marktveränderungen
- Umsetzung der Ziele erfordert Erfassung, Analyse und Optimierung vorhandener Strukturen und Abläufe (=**Business Process Reengineering BPR**)
- Unterstützung durch rechnergestützte Werkzeuge (vorgangsorientierte Informationssysteme) mit offengelegter Ablauflogik
  - **Orientierung an Workflowmanagement**: Trennung zwischen Ablauflogik und Anwendungslogik
- Erfolg liegt in der **richtig dosierten Abstraktion**
- Zielsetzung muss mit Erfolgen zur Messung in Beziehung gesetzt werden

ARIS Modell:
- „**Architektur Integrierter Informationssysteme**“ von IDS Scheer
- Werkzeug für **Phase der IS-Architektur**
- **Bewältigung der Gesamtkomplexität** durch Beschreibungssichten
- Orientierung am Geschäftsprozess
  - **Folge von Prozessketten**
  - **Hierarchisierung der Prozessketten**
  - **Auf unterster Ebene Einzelprozesse: sog. Funktionen**
  - **Abgeschlossenheit: sog. Ereignisgesteuerte Prozessketten (EPK)**

Sichten der ARIS Modellierung:

- **Funktionssicht**: GP Funktionen, Prozesshierarchie, Detailprozesse aus Steuerungssicht
- **Organisationssicht**: Unternehmensstruktur als Organigramm
- **Ressourcensicht**: Sammlung von materiellen und immateriellen Ressourcen; Bereich der Informations- und Kommunikationstechnologie (IKT)
- **Steuerungssicht**: Modellkern: EPKs ergänzt um Organisationseinheiten
- **Datensicht**: Informationsobjekte und Beziehungen als semantisches DM (ERM)

ADDON

Algorithmus:
- **Eindeutig bestimmtes Verfahren zur schematischen Lösung einer Menge von gleichartigen Aufgaben**

Definition Daten:
- **Gespeicherte Information**
- Grössen, durch den **Lösungsalgorithmus weiterverarbeitet**
- Informationen mit **charakteristischer**, durch Zeichen- und Satzfolgen gekennzeichneter **Struktur**

Datenbasis:
- **Datenbankkern**
- **Strukturierte Sammlung von Information** zu einem bestimmten Ausschnitt der realen Welt
- auf **externen Speichermedien** abgelegt

Datenbank:
- ein **integrierter, persistenter Datenbestand** einschliesslich **Metainformation** (Integritätsbedingungen, Regeln), die einer Benutzergruppe in nur einem Exemplar zur Verfügung steht

Charakterisierung Datenbank:
- **Datenbasisdaten** sind unabhängig von Nutzerprogrammen und Verarbeitungsprozessen
- **Multiuserumgebung** mit **unterschiedlichen Benutzersichten** auf Daten
- Logische Struktur ist unabhängig von physischer Speicherungsform
- **Zentralisierte DBMS Verwaltung** vermeidet **Redundanz**, sichert inhaltliche Vollständigkeit (**Integrität**) und logische Korrektheit (**Konsistenz**)
- **DBMS Nutzermanagement**, **Zugriffskontrolle** und Datenschutz

Zeiger DT:
- Zeigt auf Instanzen eines **bestimmten Datentyps**
- Besitzt **Adresse als Wert**
- Existenz eines **NIL / NULL Zeigers** zur Entscheidung, ob ausgerichtete Instanz gültig
- **Referenzierung**: Bildung der Verweisadresse im Zeiger: **zv = &z**
- **Dereferenzierung**: Auslesen der Variablen der Verweisadresse: **v = *zv**

Stack DT:
- Linearer dynamischer DT
- Element von homogenem Grunddatentyp
- LIFO: Last In First Out
- Operationen: EINFÜGEN (PUSH), ENTNEHMEN (POP), Erzeugen (=Konstruktor), Löshen (=Destruktor), Istleer
- Implementierung als statische Reihe oder verkettete Liste

Queue DT:

- Wie Stack, aber FIFO: First In First Out

Geordneter Baum DT:

- Baum DT mit geordneten Söhnen (Durchnummerierung)
- Vorgänger Beziehung: **Vater-Funktion**
- Nachfolger Beziehung: **Jüngster-Sohn-Funktion**
- Sohnreihenfolge: **Nächst-Älterer-Bruder-Funktion**

Binärbaum DT:

- Jeder Knoten hat höchsten 2 Nachfolger
- Anwendung: schnelle **Suchalgorithmen**

Graph:

- Elemente einer **nichtleeren Menge** mit **homogenem Grunddatentyp**
- **Beliebig viele Wurzelelemente**
- Kanten: **beliebig binäre Relation ohne Halbordnungsrelation**
- Existenz von **mehreren Wegen** möglich (**Transitive Hülle** enthält alle Möglichkeiten)
- **Gerichteter Graph**: Kante ist mit Richtung versehen
- **Zyklischer Graph**: Ausgang und Rückkehr von und zu einem Element

Linearer dynamischer DT, der als Graph implementiert ist?
→ Liste

Unterschiede Baum / Graph:

Baum	Graph
1 Wurzelelement	Beliebig viele Wurzelelemente
Strukturelle Elementbeziehung: Halbordnung	Beliebig binäre Relation ohne Halbordnungsrestriktion