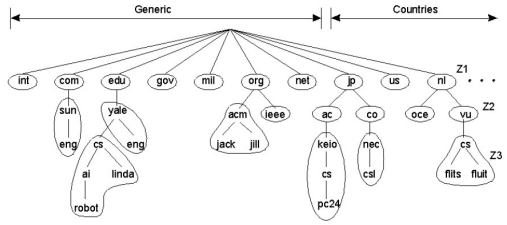
Datenstrukturen

Mittwoch, 8. Februar 2023 06:19

Datenstrukturen	• Daten : -Welche Art von Daten habe ich? -Gibt es irgendwelche Besonderheiten in den Daten? • Operationen: -Was will ich mit den Daten anfangen? -Wie häufig nutze ich die einzelnen Operationen? • (Speichern): -Wie kann ich effizient (= platzsparend) speichern? • (Besondere Anforderungen): -Einfach verständlich für den Menschen -...
Naming (Bsp. Namensgebung)	• Daten: -Menge an Objekten • Operationen: -Auffinden des Objektes -Identitätsvergleich • Speichern: --- • Besondere Anforderungen: -Dauerhaftigkeit -Lokation kann wechseln -Verständlichkeit für den Menschen -Einfache "Nummer" -um Daten zu benennen, kann man ihnen einfach Nummer geben -Zentrale Auflösung des Namens haben --> Bsp: ORCID (Open Researcher and Contributor Identifier) -Bsp: orcid.org/0000-0002-0964-4457 -Mehrere (hierarchische) "Nummern" -Daten sind in Hierarchie geordnet / man führt Hierachie ein --> Bsp: DOI (Digital Object Identifier) -Bsp: doi:10.1016/j.procir.2014.10.050 -nicht nur Nummern auch Text -Namensgebung wenn es um Verständlichkeit für den Menschen geht --> URL -Bsp: http://fusion.cs.uni-jena.de/ www.kit.edu -Teil des URL ist der Host-/Rechnername -> Rechner braucht Nummern um zu verstehen, deswegen muss der Name (Link) aufgelöst werden in eine Nummer -> wird aufgelöst durch das System DNS
DNS (Namen Adressen)	•Domain Name Service  ->Hierarchisch aufgebaut
Zusammenfassung	• Daten: -geordnete Menge von Objekten / bestimmte Menge von Objekten • Operationen: -Auffinden anhand der Position -Einfügen neuer Objekte -Löschen bestehender Objekte aus der Liste -(Ändern bestehender Objekte)

Objekte speichern mit Arrays und Listen	•1.Versuch: -> wir nehmen Felder, diese können sein: -es sind Arrays - sie haben eine feste Größe -Zugriff über Index -Ein- oder mehrdimensionale Arrays -Bsp: Bücher (Index 0-4) ->0 Passagier 21 ->1 Kinder der Freiheit ->2 Die Lebenden und die Toten ->3 Der Sohn ->4 Der Circle -Wie Speicherung der Felder? (Bsp: Speicher 0-7) ->im Speicher steht Array von oben (2-6), kontinuierlich ->es gibt Random Variable und zeigt auf Speicherzelle 2, mit ersten Element (Inex0) ->wenn man anderen Index haben will, dann ausrechnen 

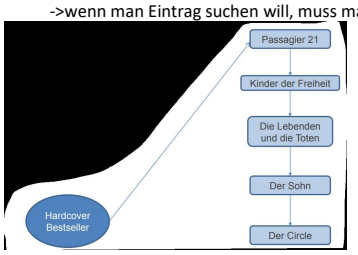
1. VERSUCH - BEWERTUNG

- **Daten**
  - geordnete Menge von Objekten ✔
- **Operationen**
  - Auffinden anhand der Position ✔
  - Einfügen neuer Objekte ✔
  - Löschen bestehender Objekte aus der Liste ✔
- **Speicher**
  - effizient ✔

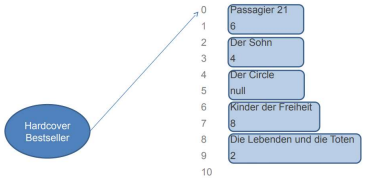
--> bei Einfügen und Löschen zu umständlich, weil man alles verschieben muss

•2.Versuch: -> Verkettete Listen

-können Beliebig lang
-Navigierender Zugriff , nicht durch anhand der Position
-Einfaches Einfügen und Löschen
-Bsp wie oben
->wir haben wieder Variable, die Anfang der Liste Zeigt
->Zeiger ist auf ersten Eintrag der Liste -> verweist dann immer auf jeweils nächste bis kein Verweis mehr drinne ist
->wenn man Eintrag suchen will, muss man alles durchgehen bis man Element gefunden hat



-Speicherung:



->Variable zeigt auf beliebigen Speicherplatz in Speicher
->Darin stehen Daten
->es gibt 2 Einträge -> der zweite ist 6, dann springe ich auf 6, usw bis Ende der Liste, bis es keinen Eintrag mehr gibt

2. VERSUCH - BEWERTUNG

- **Daten**
  - geordnete Menge von Objekten ✔
- **Operationen**
  - Auffinden anhand der Position ✔
  - Einfügen neuer Objekte ✔
  - Löschen bestehender Objekte aus der Liste ✔
- **Speicher**
  - effzient ✔

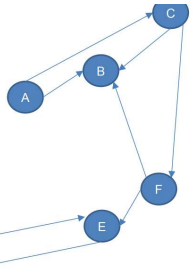
->Auffinden ist zwar möglich aber nicht mehr ausrechenbar
->nicht ganz effizient, weil man mehr Daten braucht, also auch mehr Speicher

->Möglichkeiten mit Listen und Arrays reichen nicht immer aus, man braucht also andere Datenstrukturen

Graphen	•Gegeben: ->Daten: -Objekten mit best. Beziehungen untereinander ->Operationen: -Auffinden aller direkt verbundenen Objekte -Auffinden indirekt verbundener Objekte (über 2,3,4… Verbindungen) -Sind X und Y verbunden? - … -bestehen aus Knoten -Kanten sind Verbindungen zw Knoten - es gibt gerichteten oder ungerichteten Graphen -Knoten sind benachbart, wenn sie Verbindung haben -Eingangsgrad und Ausgangsgrad sind die Anzahl der Kanten, die von Knoten hingehen bzw. Weggehen -wenn nicht gerichtet, dann ist der Grad die Anzahl der Kanten -Ordnung des Graphen ist bestimmt, wie viele Knoten ich habe -Größe des Graphen ist bestimmt, wie viele Kanten Ich habe -Pfad in Graphen ist Abfolge von Knoten, die durch Kanten verbunden sind -bei Schleife hat man Kante, von einem Knoten wieder zum gleichen Knoten hin -Zyklus geht über mehrere Knoten, aber man kommt wieder zum Ausgangsknoten zurück •Speicherung:

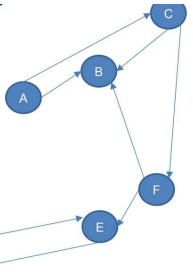
Adjazenzmatrix

	A	B	C	D	E	F	G
A		T	T				
B							
C		T				T	
D					T		
E				T			
F		T			T		
G							



Adjazenzlisten

A	B, C
B	
C	B, F
D	E
E	D
F	B,E
G	



BERWERTUNG

- **Daten**
  - Objekten mit best. Beziehungen untereinander ✔ ✔
- **Operationen**
  - Auffinden aller direkt verbundenen Objekte ✔ ✔
  - Auffinden indirekt verbundener Objekte (über 2,3,4… Verbindungen) ✔ ✔
  - Sind X und Y verbunden? ✔ ✔
  - …

Adj.-Matrix Adj.-Listen

->bei Auffinden direkter vO in Matrix müsste man in jeder Zeile nachschauen
->Auffinden indirekter vO, muss man bei beiden mehrfach suchen
->X und Y verbunden suchen bei Listen, muss man durch alle Listen durchgehen

Gerichtete Bäume

•Gegeben:
->Daten: -Objekten mit best. Beziehungen untereinander
-Hierarchie
->Operationen: -Auffinden aller Kindknoten
-Auffinden des Elternknoten
- …

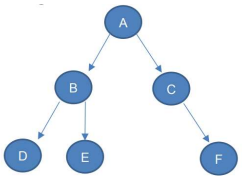
-Repräsentieren Hierarchien
-spezielle Form von gerichteten Graphen (Kanten haben Richtung
->genau 1 Knoten mit Eingangsgrad 0 (Wurzel)
->alle anderen Knoten: Eingangsgrad 1 (es geht immer nur eine Kante zu jedem Knoten hin)
->es gibt einen Pfad von der Wurzel zu jedem anderen Knoten (zusammenhängender Graph)

-Begriffe:
->Elternknoten (Knoten die über einem Knoten sind)
->Kindknoten (Knoten die unter einem Knoten sind)
->Teilbäume
->Unterbäume

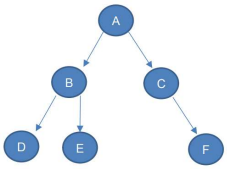
•Def. Graph: -Ansammlung von Knoten und Kanten
-Ordnung: Anzahl der Knoten
-Größe: Anzahl der Kanten
•Def. Baum: -kreisfrei (keine geschlossenen Pfade)
-zusammenhängend
•Gerichteter Graph/Baum: Kanten haben Richtung
•Gewurzelter Baum/Wurzelbaum: gerichteter Baum
•Binärbaum: Knoten haben maximal zwei direkte Nachkommen
•B-Baum / B+-Baum: komplexere Baumarten

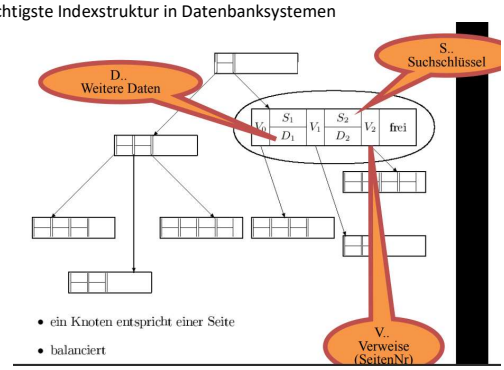
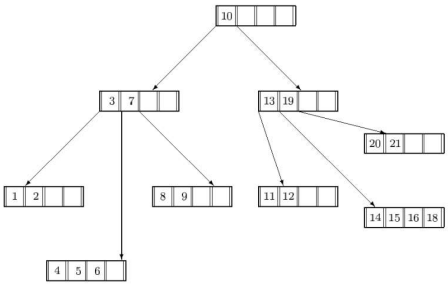
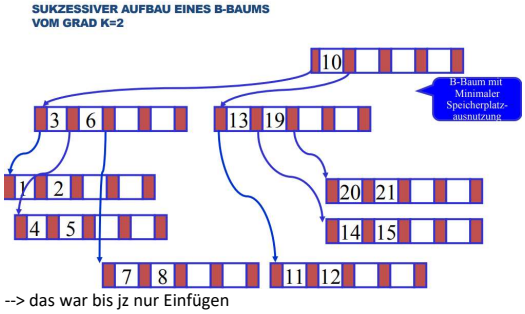
Binärbaum

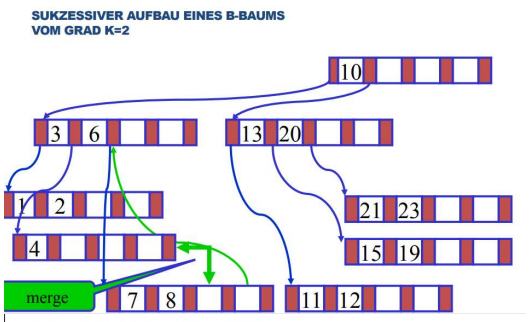
-Baum, in dem jeder Knoten maximal Ausgangsgrad 2 (= maximal 2 Kinder) hat.
-Geordnet: Für alle Knoten gilt: Rechtes Kind ist nach einer gegebenen Ordnungsrelation kleiner als linkes Kind.
-Wichtige Anwendung -> Binäre Suchbäume
-Es gibt maximal immer 2 Kinder

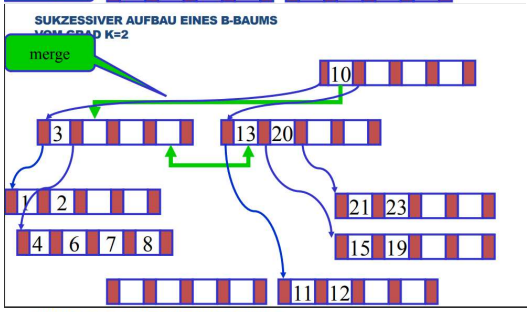


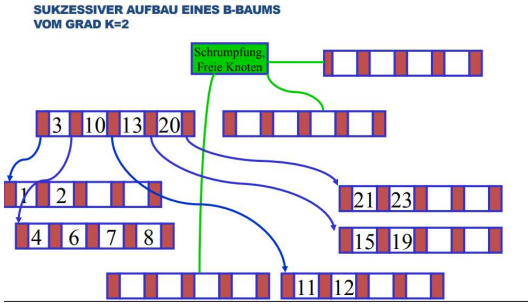
•Speicherung:

	1 A 2 B 3 C 4 D 5 E 6 - 7 F  Speicherung als Array • Knoten an Position X • Kindknoten an Position (2*X) und (2*X+1) -Ablauf von links nach rechts

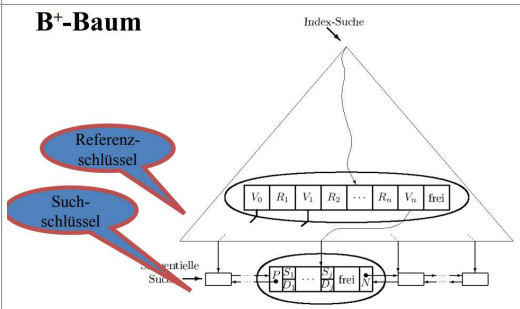
B-Baum	-Wichtigste Indexstruktur in Datenbanksystemen  •Eigenschaften: B-Baum von Grad k: 1. Jeder Weg von der Wurzel zu einem Blatt hat die gleiche Länge. 2. Jeder Knoten außer der Wurzel hat mindestens k und höchstens $2k$ Einträge. Die Wurzel hat höchstens $2k$ Einträge. Die Einträge werden in allen Knoten sortiert gehalten. 3. Alle Knoten mit n Einträgen, außer den Blättern, haben $n+1$ Kinder. 4. Seien $S_1, \ldots, S_n$ die Schlüssel eines Knotens mit $n+1$ Kindern. $V_0, V_1, \ldots, V_n$ seien die Verweise auf diese Kinder. Dann gilt: (a) V_0 weist auf den Teilbaum mit Schlüsseln kleiner als S_1. (b) V_i $(i = 1, \ldots, n-1)$ weist auf den Teilbaum, dessen Schlüssel zwischen S_i und S_{i+1} liegen. (c) V_n weist auf den Teilbaum mit Schlüsseln größer als S_n. (d) In den Blattknoten sind die Zeiger nicht definiert. •Beispiel:  -Bsp siehe Übung oder PDF  --> das war bis jz nur Einfügen --> es kommt noch Einträge löschen und Knoten mergen (Zsmfügen)







B+ -Baum



Ein B^+-Baum vom Typ (k, k^*) hat also folgende Eigenschaften:

1. Jeder Weg von der Wurzel zu einem Blatt hat die gleiche Länge.
2. Jeder Knoten – außer Wurzeln und Blättern – hat mindestens k und höchstens $2k$ Einträge. Blätter haben mindestens k^* und höchstens $2k^*$ Einträge. Die Wurzel hat entweder maximal $2k$ Einträge, oder sie ist ein Blatt mit maximal $2k^*$ Einträgen.
3. Jeder Knoten mit n Einträgen, außer den Blättern, hat $n + 1$ Kinder.
4. Seien $R_1, \ldots, R_n$ die Referenzschlüssel eines inneren Knotens (d.h. auch der Wurzel) mit $n + 1$ Kindern. Seien $V_0, V_1, \ldots, V_n$ die Verweise auf diese Kinder.
   (a) V_0 verweist auf den Teilbaum mit Schlüsseln kleiner oder gleich R_1.
   (b) V_i $(i = 1, \ldots, n-1)$ verweist auf den Teilbaum, dessen Schlüssel zwischen R_i und R_{i+1} liegen (einschließlich R_{i+1}).
   (c) V_n verweist auf den Teilbaum mit Schlüsseln größer als R_n.

Boolesche Werte	Datentyp: Boolean Werte: true, false Operationen: Or, And: Boolean x Boolean → Boolean Not: Boolean → Boolean Ture, false: → Boolean Gleichungen: Für alle x, y ∈ Boolean gilt: true or x = true true and x = x false or x = x false and x = false not true = false not false = true

Zahlen	• Natürliche Zahlen
	  • Meist kein eigener Datentyp: Integer
	• Ganze Zahlen
	  • Datentyp: Integer
	• Rationale Zahlen
	  • Meist kein eigener Datentyp
	• Reelle Zahlen
	  • Datentypen z.B. float oder double
	
	• Datentypen haben jeweils begrenzten Wertebereich und ggf begrenzte Genauigkeit!