Computergestützte Methoden

Konzepte moderner Datenhaltung

Meik Teßmer

Inhalt der heutigen Vorlesung

- Arten der Datenhaltung

  mögliche alternative Formen
- Konzepte moderner Datenbanken

Fallstudie Fahrrad-Verleih: Probleme bei der Datenhaltung

- eindeutige Identifikation eines Datensatzes
- Veränderungen der Datenstruktur (zusätzliche Daten?)
- Modifizieren einzelner Datensätze (bspw. Anzahl Verleihvorgänge während im Laufe eines Tages)
- Verknüpfung mit anderen Daten (Grunddaten $\rightarrow$ aggegrierte Daten)
- große Datenmengen

beide vorgestellten Lösungsalternativen haben hier Schwierigkeiten

Anforderungen an ein Datenhaltungssystem

Welche Eigenschaften sollte ein solches System haben?

Anforderungen an ein Datenhaltungssystem

was muss sie bieten:

- Persistenz
- Flexibilität bei der Suche
- Auswahl bei den Ergebnissen
- geringe/keine Redundanz
- Verwaltungsfunktionen
- ggf. Mehrbenutzerbetrieb

... ...

Blick über den Tellerrand: Bekannte Konzepte

- auf Rechnern legen wir Daten im *Dateisystem* ab
- funktioniert auch für große Datenmengen
- Konzept:
  - Speicherung von Daten in *Dateien*
  - Organisieren von Dateien in *Verzeichnissen*
  - Organisieren von Verzeichnissen in Verzeichnissen

  $\rightarrow$ hierarchische Organisation
- Verarbeitung:
  - Zugriff unter Angabe des *Dateipfads*
  - Öffnen mit einem passenden Programm
- besondere Formate für *strukturierte* Daten:
  - CSV, JSON, YAML
  - XML
  - für Big Data: HDF5

Bekannte Konzepte: Datenbanken

zwei große Gruppen:

1. klassische *relationale* Datenbanken

   populäre Vertreter sind MySQL/MariaDB, PostgreSQL, Oracle Database

2. sog. NoSQL-Datenbanksysteme:

   - Dokumentorientiert: MongoDB
   - Graph-Datenbanken: Neo4j
   - Key-Value-Store: Redis
   - Object Stores: Ceph (+ S3-Protokoll für Zugriff)

Fallstudie Fahrrad-Verleih: Probleme bei der Datenhaltung

- Tabellenkalkulation und programmierte Lösung: Schwierigkeiten bei großen Datenmengen und Mehrbenutzerbetrieb
- Ansatz der programmierten Lösung bisher:
  - Grunddaten: eine oder mehrere CSV-Dateien
  - Verknüpfung von Daten über ein Programm
  - Veränderung der Daten mit Hilfe eines Editors oder des Programms

Fallstudie Fahrrad-Verleih: Probleme bei der Datenhaltung

- Abgleich mit Anforderungen:
  - Persistenz: ja, über das Dateisystem garantiert
  - Flexibilität bei der Suche: hängt vom Programm ab
  - Auswahl der Ergebnisse: hängt vom Programm ab
  - Redundanz: hängt vom Entwurf der CVS-Dateien ab
  - Verwaltungsfunktionen: über das Dateisystem (Datei erstellen/löschen) oder Programm
  - Mehrbenutzerbetrieb: nicht vorhanden

Suche und Änderungen sind abhängig von Geschwindigkeit des Dateisystems und des Programms! Verbesserungsvorschläge?

Vorschau Relationale Datenbanken: Tabelle

Daten werden wieder in Tabellenform festgehalten (wie bei Tabellenkalkulation):

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11
2022-01-01	10th & H St NE	3
2022-01-01	10th & K St NW	8
2022-01-01	10th & Monroe St NE	9

$\rightarrow$ jede Zeile entspricht einem *Datensatz*

Relationale Datenbanken: Begriffe

- Zeile = Objekt oder Entität (heißt auch *Datensatz*)
- Spalte = *Attribute* eines Objekts
  - Spaltentitel = *Attributname*
  - Feldinhalt = *Attributwert*
- wichtig: Attributnamen sind *eindeutig*

→ Attribut „Station“ für ersten Datensatz hat den Wert „10th & E St NW“

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
...	...	...

Relationale Datenbanken: Begriffe

(vorläufige) Definition:

> *Eine Menge von Datensätzen, die durch die Werte ihrer gleichen Attributkombination vollständig und eindeutig beschrieben werden, heißt* ***Relation****.*

Daher kommt auch der Name *Relationale Datenbanken*.

Relationale Datenbanken: Grundlage Mengenlehre

- Mengen
  - Z: Menge der ganzen Zahlen
  - N_0: Menge der natürlichen Zahlen mit 0
  - Q: Menge der rationalen Zahlen
  - M := {-1, 2, 3} (eine eigene Menge)
- Kombination zweier Mengen heißt *kartesisches Produkt* oder *Kreuzprodukt*
  - A, B: zwei Mengen
  - $A \times B$ ist die Menge aller Paare (a, b) mit $a \in A$ und $b \in B$

$A \times B := \{(a, b) | a \in A \wedge b \in B\}$

- Elemente dieser kombinierten Mengen heißen *Tupel*

Relationale Datenbanken: Grundlage Mengenlehre

- man kann auch mehr als zwei Mengen kombinieren:
  $A \times B \times ... \times Z$

  $\rightarrow$ heißen dann *n-Tupel*

- Übertragung auf Tabellen:
  - jede Spalte entspricht einer Menge
  - Anordnung der Spalten ist in der Tabelle egal(!)
  - Relation ist *Teilmenge* eines kartesischen Produkts:
    $R \subseteq A \times ... \times Z$

Relationale Datenbanken: Grundlage Mengenlehre

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11
...	...	...

Sei

- D: Menge aller Datumseinträge
- S: Menge aller Stationen
- C: Menge der Zahl der verliehenen Fahrräder

Relation ist dann: Verleih $\subset D \times S \times C$

Unterschiede Relation-Tabelle

- Relation kann *unendlich* viele Elemente enthalten

  Tabellen haben immer *endlich* viele Elemente,
- Relationen haben eine Anordnung der beteiligten Mengen

  Tabellenspalten können bei Bedarf umgeordnet werden,
- ein Element kann in einer Relation nur *einmal* vorkommen

  in Tabellen kann es *Duplikate* geben (sollte es aber nicht),
- in Tabellen kann ein Attributwert NULL sein

Was kann man mit Tabellen machen?

- 9 wichtige Operationen
  - auf einer Tabelle (2)
  - auf strukturgleichen Tabellen (4)
  - auf unterschiedlichen Tabellen (3)

Verarbeitung einer einzelnen Tabelle

Was kann man mit Tabellen machen?

- Beispiel: Menge aller Namen ausgeben

  $\rightarrow$ Projektion auf bestimmte Attribute; macht die Tabelle „schmaler“

- Projektion heißt PROJECT

  $A \subseteq M \times ... \times N, R := \{M', ..., N'\}$: Auswahl aus $M, ..., N$

  PROJECT(A, R) $:= \{a \in A | \text{a ohne Attribute R}\}$

- Datensätze können mehrfach im Ergebnis auftreten

Verarbeitung einer einzelnen Tabelle

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11
...	...	...

Projektion auf eine Spalte (Station):

Station
10th & E St NW
10th & Florida Ave NW
10th & G St NW
...
10th & G St NW

Verarbeitung einer einzelnen Tabelle

Was kann man noch mit Tabellen machen?

- Auswahl von Elementen aus einer Relation anhand einer Bedingung, z.B. Geschlecht

  $\rightarrow$ macht die Tabelle „kürzer“

- Selektion heißt RESTRICT
  - $A \subseteq M \times ... \times N$
  - $\chi : A \rightarrow \{\text{True}, \text{False}\}$ (Prüffunktion)
  - $\text{RESTRICT}(A, \chi) := \{a \in A | \chi(a) = \text{True}\}$

Verarbeitung einer einzelnen Tabelle

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11
...	...	...

Restriktion auf die Zeilen mit `count>5`:

Datum	*Station*	*count*
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11

Verarbeitung strukturgleicher Relationen

- Beispiel: bestehende Liste mit weiteren Verleih-Vorgängen erweitern

  $\rightarrow$ zwei strukturgleiche Tabellen (haben dieselben Attributnamen) werden vereinigt

- Mengenvereinigung heißt UNION:

  $A \subseteq M \times ... \times N$ und $B \subseteq M \times ... \times N$

  UNION$(A, B) := A \cup B$

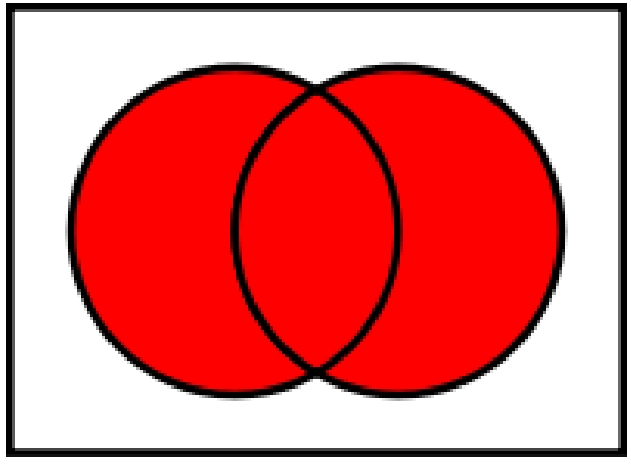


Figure 1: Vereinigung

Verarbeitung strukturgleicher Relationen

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24

Datum	*Station*	*count*
2022-01-01	Sennestadt	4
2022-01-01	Lohmannshof	21

$\rightarrow$ Ergebnistabelle:

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	Sennestadt	4
2022-01-01	Lohmannshof	21

Verarbeitung strukturgleicher Relationen

- Beispiel: Vergleich zweier Listen zwecks Suche nach gleichen Mitgliedern
- Durchschnitt heißt INTERSECTION; entspricht einer gewöhnlichen Schnittmenge zweier Mengen

  $A \subseteq M \times ... \times N$ und $B \subseteq M \times ... \times N$

  INTERSECTION$(A, B) := A \cap B$

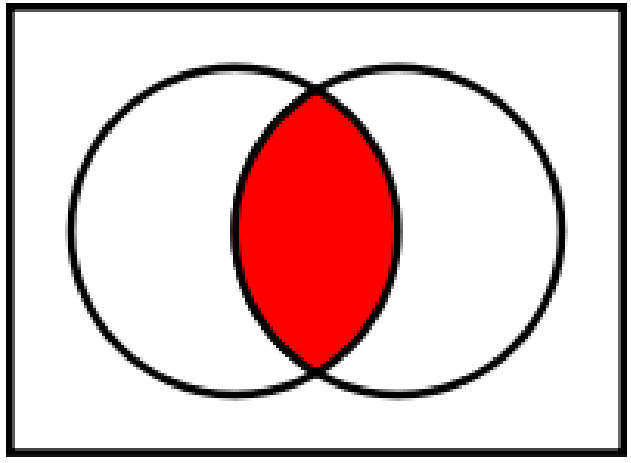


Figure 2: Schnittmenge

Verarbeitung strukturgleicher Relationen

- Beispiel: Vergleich zweier Listen auf Unterschiede

  $\rightarrow$ Ergebnis enthält die Elemente, die nur in der einen Liste vorkommen

- Differenz heißt DIFFERENCE; entspricht der Differenzmenge zweier Mengen

  $A \subseteq M \times ... \times N$ und $B \subseteq M \times ... N$

  DIFFERENCE$(A, B) := A \backslash B$ (A ohne B)

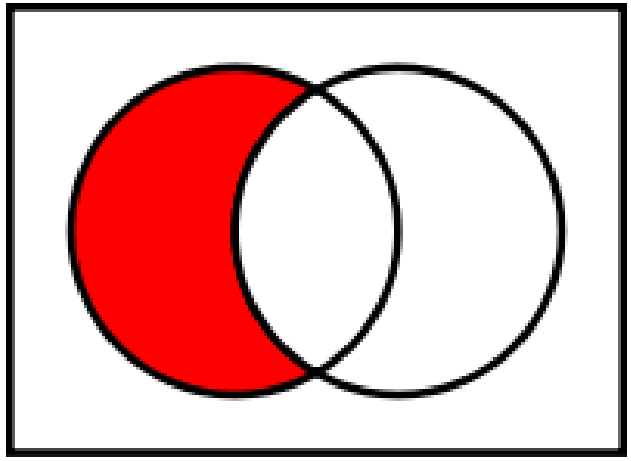


Figure 3: Differenz

Verarbeitung strukturgleicher Relationen

- Beispiel: Kombination zweier Listen ohne doppelte Datensätze

  $\rightarrow$ Ergebnis enthält die Elemente, die nur in der einen Liste *oder* der anderen vorkommen

- Symmetrische Differenz entspricht

  $A \subseteq M \times ... \times N$ und $B \subseteq M \times ... \times N$

  SYMM. DIFFERENCE$(A, B) := A \triangle B$

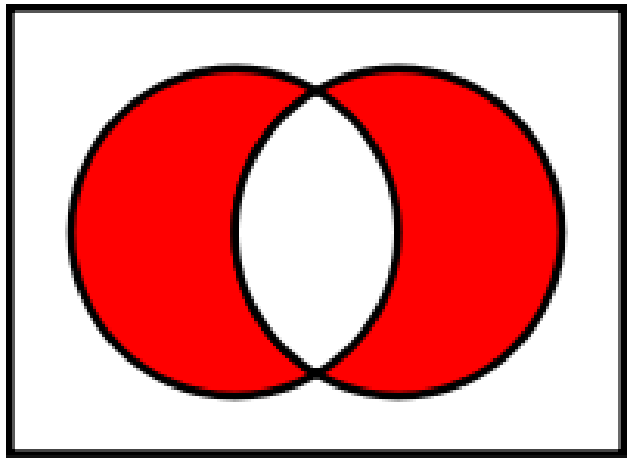


Figure 4: Symmetrische Differenz

Zusammenfassung Relationale Operatoren

auf einer Tabelle:

- Projektion
- Selektion (Restriktion)

auf strukturgleichen Tabellen:

- Vereinigung
- Durchschnitt
- Differenz
- symmetrische Differenz

Verarbeitung strukturverschiedener Relationen

Was kann man mit Tabellen machen?

- Beispiel: bestehende Sammlung um Informationen wie *Genre* erweitern

  $\rightarrow$ Ergebnis: Tabelle mit erweiterten Datensätzen

- kartesisches Produkt: PRODUCT

  Erweiterung einer Tabelle in der „Breite“

  $A \subseteq M \times ... \times N$ und $B \subseteq O \times ... \times P$

  PRODUCT$(A, B) := A \times B \subseteq M \times ... \times N \times O \times ... P$

- Anzahl Datensätze: Ergebnis der Multiplikation der Zeilenanzahlen der Ausgangstabellen

Verarbeitung strukturverschiedener Relationen

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24

Linie	*Haltestelle*
4	10th & E St NW
1	10th & Florida Ave NW

$\rightarrow$ Ergebnis:

Datum	*Station*	*count*	*Linie*	*Haltestelle*
2022-01-01	10th & E St NW	1	4	10th & E S...
2022-01-01	10th & E St NW	1	1	10th & Flo...
2022-01-01	10th & Florida Ave NW	24	12	10th & G S...
...				

Verarbeitung strukturverschiedener Relationen

- Join: „seitliches Anfügen“ einer Tabelle
- entspricht dem kartesisches Produkt mit anschließender Selektion

einfachste Form: Equi-Join

- kartesisches Produkt mit anschließender Restriktion, dass der Inhalt bestimmter Spalten gleich sein muss (Gleichheitsoperator)
- Beispiel: Zuordnung der Linien zu einer Station
  - Tabelle R: Liste der Verleih-Stationen
  - Tabelle S: Liste der Linien samt Haltestellen

  Ziel: Anbindung der Stationen auflisten

Verarbeitung strukturverschiedener Relationen

- Sei $R \subseteq A \times ... \times D$ und $S \subseteq E \times ... \times G$.
- Sei $\Phi : R \times S \rightarrow \{\text{True}, \text{False}\}$.
- Equi-Join: EQUI_JOIN(R, Φ, S)

  entspricht RESTRICT(PRODUCT$(R, S), \Phi)$
- Beispiel: EQUI_JOIN$(R, (R.A == S.E), S)$

R:

A	B	C	D
1	2	3	4
4	5	6	7
7	8	9	0

S:

E	F	G
1	2	3
7	8	9

R x S:

A	B	C	D	E	F	G
1	2	3	4	**1**	2	3
4	5	6	7	1	2	3
7	8	9	0	1	2	3
1	2	3	4	7	8	9
4	5	6	7	7	8	9
7	8	9	0	**7**	8	9

JOIN(R, R.A = S.E, S):

A	B	C	D	E	F	G
1	2	3	4	**1**	2	3
7	8	9	0	**7**	8	9

Figure 5: Equi-Join

Verarbeitung strukturverschiedener Relationen

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24

Linie	*Haltestelle*
4	10th & E St NW
1	10th & Florida Ave NW

$\rightarrow$ Ergebnis (Prüffunktion: `Stations==Haltestelle`):

Datum	Station	count	Linie	Haltestelle
2022-01-01	10th & E St NW	1	4	10th & E S...
2022-01-01	10th & Florida Ave NW	24	1	10th & Flo...

Verarbeitung strukturverschiedener Relationen

- Verallgemeinerung des Join:
  - Θ-Join (Theta-Join): kann alle Vergleichsoperatoren nutzen
  - Natural Join: verbesserter Equi-Join, weil die zweite Vergleichsspalte ausgeblendet wird

  es gibt noch andere Joins (Semi-, Outer-) ...

Beispiel Natural Join: A ist in beiden Tabellen vorhanden

R:

A	B	C	D
1	2	3	4
4	5	6	7
7	8	9	0

S:

A	F	G
1	2	3
7	8	9

NATURAL JOIN (R, S):

A	B	C	D	F	G
1	2	3	4	2	3
7	8	9	0	8	9

Figure 6: Natural-Join

Verarbeitung strukturverschiedener Relationen: Natural Join

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24

Linie	*Haltestelle*
4	10th & E St NW
1	10th & Florida Ave NW

$\rightarrow$ Ergebnis (Prüffunktion: `Stations==Haltestelle`):

Datum	Station	count	Linie
2022-01-01	10th & E St NW	1	4
2022-01-01	10th & Florida Ave NW	24	1

Verarbeitung strukturverschiedener Relationen

- Division: „dividiert“ eine Relation durch eine andere
- Beispiel: „Wähle *alle* Eltern aus (Vater, Mutter), die ein Kind mit dem Namen Maria und dem Alter 4 *und* ein Kind mit dem Namen Sabine und dem Alter 2 haben.”
- Attribute der Ergebnistabelle: R ÷ S = (Vater, Mutter)

R:

Vater	Mutter	Kind	Alter
Franz	Helga	Harald	5
Franz	Helga	Maria	4
Franz	Ursula	Sabine	2
Moritz	Melanie	Gertrud	7
Moritz	Melanie	Maria	4
Moritz	Melanie	Sabine	2
Peter	Christina	Robert	9

S:

Kind	Alter
Maria	4
Sabine	2

R÷S

Vater	Mutter
Moritz	Melanie

Figure 7: Division

Zusammenfassung Relationale Operatoren

auf strukturverschiedenen Tabelle:

- Produkt
- Join (Equi, Theta, Natural)
- Division

Problem: Eindeutiges Identifizieren von Datensätzen

- Rückblick: Identifizieren einzelner Datensätze in einer Tabellenkalkulation?
- Wie gelingt dies in einer relationalen Datenbank?
- Gibt es eigentlich doppelte Datensätze?

Identifizieren von Datensätzen

Beispiel: Mehrfach vorkommende Datensätze

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11
...	...	...

$\rightarrow$ Projektion auf Station:

Station
10th & E St NW
10th & Florida Ave NW
10th & G St NW
...
10th & G St NW

Lösung: Schlüssel

- Wert des Attributs kann ein sog. *Schlüsselkandidat* sein
- Schlüsselkandidat: Attribut(e) mit tabellenweit *eindeutigen* Werten (Eindeutigkeit)
- Minimalität: entfernt man die Schlüsselattribute, geht die Eindeutigkeit verloren
- es kann mehr als einen Schlüsselkandidaten geben
- mehr als ein Attribut: zusammengesetzter Schlüssel

Schlüssel: Liste mit und ohne Schlüsselattribut

Ausgangstabelle:

Datum	*Station*	*count*
2022-01-01	10th & E St NW	1
2022-01-01	10th & Florida Ave NW	24
2022-01-01	10th & G St NW	11
...	...	...

Liste ohne wichtiges Schlüsselattribut `Datum`:

Station	*count*
10th & E St NW	1
10th & Florida Ave NW	24
10th & G St NW	11
...	...
10th & G St NW	3

Primärschlüssel

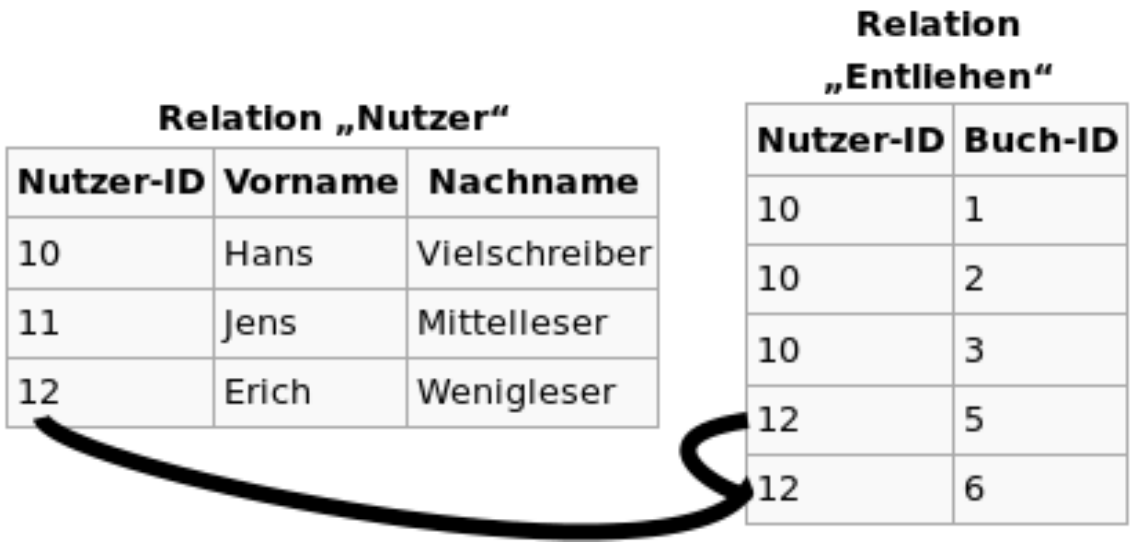
- Primärschlüssel: beim Tabellenentwurf „angedachter“ Schlüssel
- oft als „ID“ bezeichnet, hat keine Bedeutung für den Anwender

ID	Datum	Station	`count`
1	2022-01-01	10th & G St NW	11
2	2022-01-01	10th & K St NW	8
3	2022-01-01	10th & Monroe St NE	9
4	...	...	...

$\rightarrow$ Primärschlüssel: *ID*

Fremdschlüssel und referenzielle Integrität

- Beziehung zwischen zwei Tabellen: Werte bestimmter Attribute
- Fremdschlüssel: Schlüsselkandidat einer *anderen* Tabelle
- referenzielle Integrität: zu jedem Fremdschlüsselwert in einer Tabelle *muss* es einen Wert in einer zugehörigen Tabelle geben



Relation „Nutzer“

Nutzer-ID	Vorname	Nachname
10	Hans	Vielschreiber
11	Jens	Mittelleser
12	Erich	Wenigleser

Relation „Entliehen“

Nutzer-ID	Buch-ID
10	1
10	2
10	3
12	5
12	6

Figure 8: Referenzielle Integrität

Notationsweise

Schema der Tabelle: Beschreibung der Attribute und ihrer Funktion für die Datenbank

Beispiel: Tabelle *Mitarbeiter*

Mitarbeiter(PersonalNr#, Name, Vorname, Titel, Geschlecht, GebDatum)

- Schlüsselattribute erhalten ein # nachgestellt (auch Fremdschlüssel!)
- Primärschlüssel einer Tabelle werden unterstrichen

Beispiel: Tabellen *Leistung* und *LeistungsPos*

Leistung(LeistungsNr#, Datum, PersonalNr#)
LeistungsPos(LeistungsNr#, AuftragsNr#, vonZeit, Dauer, Tätigkeit)

Normalisierung/Normalformen

- atomare Werte: jeder Attributwert ist „atomar“, d.h. es stehen dort keine Listen o.ä.

  Bsp.: Name = „Pan, Peter“ ist nicht atomar, sondern packt Vor- und Nachname in einen Wert

- 1. Normalform: es werden nur atomare Werte verwendet
- nur auf Tabellen in der 1. Normalform kann sauber gearbeitet werden

Normalisierung/Normalform

Beispiel: Primärschlüssel ist (Belegnr, Matrikelnr)

Belegnr	*Matrikelnr*	*Nachname*
310059	1234567	Mueller
316000	7654321	Schmidt
370009	1234567	Mueller
270001	1234567	Mueller

Warum ist diese Tabelle „ungeschickt“?

Normalisierung/Normalform

- sog. *funktionale Abhängigkeit* des Nachnamens von der Matrikelnummer
- Nachname ist hier überflüssig und fehleranfällig
- 2. Normalform ist erfüllt, wenn:
  - 1. NF ist erfüllt
  - jedes Nichtschlüssel-Attribut ist nur vom Primärschlüssel abhängig, nicht aber von einer Teilmenge desselben.

    Im Beispiel: Nachname ist abhängig von Matrikelnr, einer Teilmenge des Primärschlüssels
- es gibt noch weitere Normalformen (3., 4., Boyce/Codd, 5.)

Beispiel Verleih-Tabelle

ID	Datum	StationId	count
1	2022-01-01	1	11
2	2022-01-01	2	8
3	2022-01-02	2	4
4	2022-01-01	3	9
...	...	...	

StationId	Station
1	10th & G St NW
2	10th & K St NW
3	10th & Monroe St NE

Verwaltung einer Datenbank

- Menge aller Datensätze == *Datenbank*
- Verwaltung erfolgt durch *Datenbankmanagementsystem* (DBMS)
- je DBMS mehrere Datenbanken möglich
- Anforderungen an ein DBMS:
  - Datenbank einrichten
  - darin Tabellen erzeugen und löschen
  - Datensätze hinzufügen, ändern und löschen
  - Daten abfragen
- SQL (Structured Query Language): verbreitete Abfragesprache für relationale Datenbanken

Ende

- Wiederholung Mengenlehre: Wikipedia
- zu Datenbanken:
  - Schicker (2017)
  - Steiner (2017)
  - Geisler (2014)

Literatur

Geisler, Frank. 2014. *Datenbanken: Grundlagen Und Design*. 5. ed. Mitp Professional. http://web.b.ebscohost.com/ehost/detail/detail?vid=0/&sid=a6841ba5-067d-4b51-8b70-555c9c351781/%40sessionmgr104/&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ/%3d/%3d/#AN=979056/&db=nlebk.

Schicker, Edwin. 2017. *Datenbanken Und SQL: Eine Praxisorientierte Einführung Mit Anwendungen in Oracle, SQL Server Und MySQL*. 5. ed. Wiesbaden: Springer Vieweg. https://link.springer.com/content/pdf/10.1007/%2F978-3-658-16129-3.pdf.

Steiner, René. 2017. *Grundkurs Relationale Datenbanken: Einführung in Die Praxis Der Datenbankentwicklung Für Ausbildung, Studium Und IT-Beruf*. 9. ed. Wiesbaden: Springer Vieweg. https://link.springer.com/content/pdf/10.1007/%2F978-3-658-17979-3.pdf.