Datenbank-Systeme

Von unstrukturierten zu strukturierten Daten

Internationaler Frauenstudiengang Informatik

WiSe 2017/18

Renzo Kottmann

Kontakt:

- mail
- linkedin:http://www.linkedin.com/in/renzokottmann
- twitter: @renzokott

Organisatorisches

Naechste Vorlesung am 18.10.2017 faellt aus!

Wiederholung

- Definition von Daten
- Welche Kategorien von Daten gibt es?
- Was ist eine Datenbank im Allgemeinen?

Datenbank Entwicklung

Das Erstellen einer neuen Datensammlung erfordert dazu u.a. die Erfassung von Daten.

Von unstrukturierten zu strukturierten Daten

- Ein neue Datensammlung beginnt häufig mit der Erfassung der Daten
  - durch Digitalisierung
  - oder liegen schon vor
    - z.B. in einfachen Textdokumenten wie Word-files, PDFs oder auch Internetseiten.

Erfassung

Daten der Modulteilnehmer

Digital(isierung)

Wie?

Digital(isierung)

Wie?

Zur ersten Erfassung und Verwaltung bietet sich auch ein Tabellenkalkulationsprogramm wie z.B. Excel, OpenOffice oder online Tools wie Google Spreadsheets an.

Google Spreadsheet

Google Spreadsheet

URL: https://docs.google.com/spreadsheets/d/1-fYfxK-aszpVEa2_nW0eUcaqjXB7P3tKM5cjQ5C3FsQ/edit?usp=sharing

Erkenntnisse??

Meine Erkenntnisse:

- Als Datenbankstruktur ist eine Tabelle geeignet:
  - Ein Datensatz pro Zeile
  - Eine Eigenschaft (Attribut) pro Spalte
  - Erste Zeile enthält die Namen der Eigenschaften (anstatt eines Datensatzes)
  - Die Reihenfolge der Zeilen ist egal
  - Die Reihenfolge der Spalten ist egal

Daten Speicherung

- Files:
  - Nun befinden sich die Daten in einer Datei persistent im Dateisystem gespeichert, d.h. diese werden über die Laufzeit eines Programms oder des Computers hinaus existieren.
  - Nicht zwingend Excel-Format:
  - Excel ist ein binäres Format
  - In vielen Fällen reicht eine Textdatei
    - Z.B. das CSV-Format (Comma-separated values)

Comma Separated Values

- CSV-Dateien entsprechen Tabellen, gekennzeichnet durch:
  - jede Zeile durch ein Zeilenendezeichen
    - Spalten durch ein Trennsymbol wie
    - z.B. ein Komma ',' oder Semikolon ';'

Eine CSV-Datei kann man mit allen Textverarbeitungsprogrammen und auch Tabellenkalkulationsprogrammen bearbeiten werden.

Probleme dieser einfachen Struktur:

```
Viereck;Axel;26123;Oldenburg;
Huber;Ina;12345;FFM;0123/65235
Lustig;Olga;12345;Frankfurt;0123/45456
Mustermann;Erika;12345;Frankfurt;0123/45456
Henseler;Herwig;26197;Großenkneten;04435/388486 (Fax:388487)
Lustig;Peter;Frankfurt;0123/45456
Huber;Ina;3454;Dresden;0283/11111
Mustermann;Erika;;Bremen;436654
```

Probleme dieser einfachen Struktur:

```
Viereck;Axel;26123;Oldenburg;
Huber;Ina;12345;FFM;0123/65235
Lustig;Olga;12345;Frankfurt;0123/45456
Mustermann;Erika;12345;Frankfurt;0123/45456
Henseler;Herwig;26197;Großenkneten;04435/388486 (Fax:388487)
Lustig;Peter;Frankfurt;0123/45456
Huber;Ina;3454;Dresden;0283/11111
Mustermann;Erika;;Bremen;436654
```

- Mögliche Redundanzen
- Beziehungen werden nicht repräsentiert
- Keine Festlegung von Datentypen und Datenintgritätsbedingungen
- Unklare Eindeutigkeiten

Semistrukturierte Daten: XML

```xml
<?xmlversion="1.0"?>
<adressen>
    <adresse>
        <nachname>Lustig</nachname>
        <vorname>Peter</vorname>
        <plz>12345</plz>
        <ort>Frankfurt</ort>
        <telefon>0123/45456</telefon>
    </adresse>
    <!-- einige Eintraege ausgelassen -->
    <adresse>
        <telefon>436654</telefon>
        <nachname>Mustermann</nachname>
        <vorname>Erika</vorname>
        <ort>Bremen</ort>
    </adresse>
</adressen>
```

Semistrukturierte Daten: JSON

```
{
    "adressen": [{
        "nachname": "Lustig",
        "vorname": "Peter",
        "plz": "12345",
        "ort": "Frankfurt",
        "telefon": "0123/45456"
    },
    {
        "telefon": "436654",
    "nachname": "Mustermann",
        "vorname": "Erika",
        "ort": "Bremen"

    }]
}
```

Semistrukturierte Daten

- Für einfache Anwendungen kann diese Form der Datenspeicherung und verwaltung in Datei(en) durchaus ausreichen.
- Filesysteme unterstuetzen nicht (oder nur unzureichend):
  - effiziente Suche und Modifikation von kleinen Dateneinheiten,
  - komplexe Datenanfragen,
  - Transaktionen
  - effizientes buffering und caching von Daten im Hauptspeicher

Anforderungen an Datenbanken in der Regel deutlich höher.

Danke fuer die Zusammenarbeit