INF

Datenbanken

Datenbank - Geschichte

- ca. 1972 präsentierte E. F. Codd sein Konzept der Trennung von Logik und physischer Speicherung
- 1976 P. Chen entwickelt das ERM (Entity Relationship Model)
- Anfang 1980 startet das erste kommerzielle Relationale Datenbanksystem mit Oracle Version 2 und SQL (Structured Query Language) wird zum Standard

Datenbanksystem

- Allgemeine Definition

  „System zur Beschreibung, Speicherung und Wiedergewinnung von Datenmengen"
- Grundlegende Gedanken von Datenbanken
  - Strukturierte Speicherung von Datenmengen
  - Vermeidung von Redundanz
  - Datenschutz
  - Datensicherheit
  - Datenspeicherung und Logik werden separat verwaltet

- Entwicklung
  - sequentielle Speicherung von Daten in Dateien (d.h. Verarbeitung erfolgt der Reihe nach)
  - Mehrfachzugriff über eine Adresse mittels Indexdateien oder Hashverschlüsselung an bestimmter Position
  - anstatt reiner Datenverarbeitung nun die Datenadministration

Datenbanken

- Eine **Datenbank** (DB, engl. Database) ist eine systematische Sammlung von Daten.
- Zur Nutzung und Verwaltung der in der DB gespeicherten Daten benötigt der Anwender ein **Datenbankverwaltungssystem** (DBMS, engl. Database Management System).
- Die Kombination aus DB und DBMS ist das **Datenbanksystem** (DBS, engl.: Data Base System), das jedoch häufig fälschlicherweise als Datenbank bezeichnet wird.

Relationale DB

- zum ersten Mal 1970 von Dr. Edgar F. Codd vorgestellt
- Konzept der Datenspeicherung in Datenbank und Operatoren zur Datenmanipulation (z.B. SQL - Structured Query Language)
- die wichtigsten Begriffe wurde geprägt:
  - Table (Tabelle), Column (Spalte), Row (Zeile, steht auch für Datensatz)
  - Integrity (Integrität)
  - Primary Key / Foreign Key (Primär-/Fremdschlüssel)
  - Unique (Unikat - Einmaligkeit)

Grundlegende Begriffe

- **Redundanz und Inkonsistenz**
  - Von Redundanz spricht man, wenn Daten (und Informationen) mehrfach gespeichert werden.
  - Werden diese Daten jetzt nur an einer Stelle geändert, kommt es zur Dateninkonsistenz
  - Daten, die isoliert in Dateien gehalten werden, speichern Informationen oft mehrfach, d.h. redundant
- **Integritätsverletzung**
  - bestimmte Datenverarbeitungsvorgänge sollen vom System „abgelehnt" werden (Abhängigkeitsbedingung – wenn, dann)
  - Betrifft auch Datentypen und Beschränkungen auf die Dateneingabe und -abhängigkeit

- Referentielle Integrität
  - ist eine spezielle Form der Integrität.
  - Es geht hierbei um die Beziehung zwischen FK und PK.
  - Ein FK kann nur existiert (eingefügt werden), wenn ein entsprechender PK existiert.
  - Ein PK kann nur geändert oder gelöscht werden, wenn der FK dies zulässt!
  - Änderung oder Löschung eines PK:
    - wird standardmäßig abgelehnt!
    - kann so eingestellt werden, dass der FK auch geändert oder gelöscht wird (oder auch auf NULL gesetzt wird).
    - Die Einstellungen in z.B. MySQL lautet:
      - CASCADE: Änderungen oder Löschungen am PK werden auch beim FK durchgeführt
      - RESTRICT, NO ACTION: Standardeinstellung, Löschungen oder Änderungen beim PK werden abgelehnt.
      - SET NULL: bei Änderung oder Löschung eines PKs wird der FK auf NULL gesetzt

- Entity
  - Objekt (der realen Welt), z.B. Schüler
  - Definition: Nach der Realität funktional zusammen gehörende Attribute
- Attribut
  - Eigenschaft eines Objekts/Entities, z.B. wird die Entität Schüler über ein Attribut Nachname näher beschrieben
- Relationship (Beziehung)
  - beschreibt die Zusammengehörigkeit und Abhängigkeit von Entitäten

- Schlüssel
  - **Primärschlüssel – Primary Key – PK**

    Zeichnet die Eigenschaft einer Entität aus. Als Primärschlüssel wird jenes Attribut gekennzeichnet, dass eine Relation eindeutig (einmalig) ausweißt. z.B. Sozialversicherungsnummer als Attribut der Entität Person.

    Ein PK kann auch zusammengesetzt sein, sofern die Werte der Attribute eindeutig (in Kombination unique) sind.
  - **Fremdschlüssel – Foreign Key – FK**

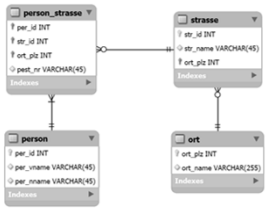
    Als Fremdschlüssel wird ein Attribut ausgewiesen, das der Primärschlüssel einer anderen Entität ist und durch Normalisierungsregeln in Zwischentabellen aufgenommen wurde.

SQL-Kommandos Kategorien

- Datendefinitionssprache (Data Definition Language, DDL)
  - Beschreibt Operationen zur Änderung der Datenstruktur (CREATE, ALTER, DROP, RENAME, TRUNCATE)
- Datenmanipulationssprache (Data Manipulation Language, DML)
  - Steht für Änderung von abgespeicherten Datenobjekten (INSERT, UPDATE, DELETE, MERGE)
- Datenkontrollsprache (Data Control Language)
  - Zur Vergabe von DB Rechten (GRANT, REVOKE)

Entity Relationship Model (ERM)

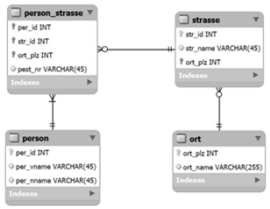
- auch ERD – Entity Relationship Diagram
  - zur graphischen Darstellung von Entitäten (Tabellen) und deren Beziehungen
  - kann in das relationale Modell übergeleitet werden



Non or identifying Relationship

- Identifying Relationship: damit ist gemeint, dass ein Entity nur mit Hilfe eines anderen Entities existieren kann, d.h.
  - der FK ist Teil des PKs in der Kindtabelle
  - oder der PK der Elterntabelle ist auch PK der Kindtabelle
- Non-Identifying Relationship: „normale" Beziehung zwischen den Entities.

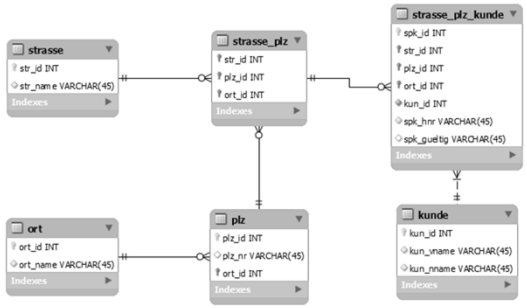
- **Identifying Relationship** kann zu einer vereinfachten Abfrage führen.
- Beispiel: Bezogen auf folgendes ERM



Personendaten und zugeordneter Ortsname sollen in einer Tabelle ausgegeben werden. Durch die Verwendung von identifying Relationship kann die nicht benötigte Tabelle STRASSE im JOIN ausgeschlossen werden:

```
select p.*,
       ort_name
from   person p natural join
       person_strasse
       natural join
       ort;
```


Beispiel DB kunde



Abfrage: Kundenname + PLZ und Ortsname

- ohne Berücksichtigung von Identifying:

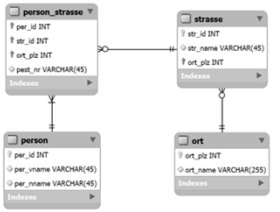
```
select   concat_ws(' ', kun_vname, kun_nname) as "Kunde",
         concat_ws(' ', plz_nr, ort_name) as "Ort"
from     strasse_plz sp, strasse_plz_kunde spk, kunde k, ort o,
plz p
where    o.ort_id = p.ort_id
and      p.plz_id = sp.plz_id
and      spk.str_id = sp.str_id
and      spk.plz_id = sp.plz_id
and      spk.ort_id = sp.ort_id
and      spk.kun_id = k.kun_id;
```

- mit Berücksichtigung von Identifying:

```
select   concat_ws(' ', kun_vname, kun_nname) as "Kunde",
         concat_ws(' ', plz_nr, ort_name) as "Ort"
from     kunde k, strasse_plz_kunde spk, ort o, plz p
where    spk.kun_id = k.kun_id
and      spk.ort_id = o.ort_id
and      spk.plz_id = p.plz_id;
```

Optional oder Mandatory

- Verbindungen zwischen den Entitäten können verpflichtend oder optional bestehen.
- Im Grunde geht es dabei um die Form der Beziehung, z.B. 0:N oder 1:N.
- Das hat vor allem eine Bedeutung für die spätere Umsetzung in z.B. eine GUI.
- Zeichnerische Umsetzung: Kreis optional, Längsstrich Pflicht.
- Erläuterung anhand des vorherigen Beispiels Personen und Adressen:



- Ein Ort kann eingetragen werden, ohne, dass er eine Strasse zugeordnet ist!
- Eine Strasse muss einem Ort zugeordnet sein.
- Eine Strasse kann einer Person zugeordnet sein.
- Eine Person muss einer Strasse zugeorndet sein.

Symbole für Händische Zeichnung

- Entität: abgerundetes Rechteck (softbox)
- Attribut: Text im Rechteck, diesem werden folgende Zeichen vorangestellt:
  - Primärschlüssel: Raute # (MySQL Workbench – MW - Schlüsselsymbole
  - Pflichtattribute: Stern * (MW Rautesymbol ausgefüllt)
  - Optionale Attribute: Kreis o (MW Rautesymbol leer)
  - Foreign Key: FK

- Relation: Linie mit Text bei jedem Entity – bei den Linien unterscheidet man nach den Beziehung durch strichlierte (non-identifying) oder durchgezogene Linie (identifying), bzw. eine Linie (1:1) oder "Krähenfüsse" am Ende der Linie (N)

- Symbolik im Unterricht:
- "Krähenfüsse" für das Anzeigen einer N-Beziehung
- Händisch: es gibt nur durchgezogene Linien (nicht strichliert)

Richtlinien für den Entwurf

- Ein ERM entspricht einer bestimmten Normalform (NF - Erläuterung folgt noch). Meistens der 3. NF.
- Daten werden nur einmal gespeichert!
- Daten werden logisch gespeichert, d.h. der Vorname einer Person wird nicht in einer Spalte Buchtitel gespeichert, sondern unter Vorname oder ähnlichem

Namenskonvention

- nach Oracle SQL
- Tabellen- und Spaltenbezeichnung:
  - zusammengesetzte Wörter:
    - den ersten Buchstaben des ersten Wortes
    - den ersten Buchstaben des zweiten Wortes (und jedes weiteren Wortes)
    - den letzten Buchstaben des letzten Wortes
    - Beispiel für Attribut: Tabelle strasse_plz ergibt als Präfix spz_
  - ein Wort:
    - erster, zweiter Buchstabe und
    - letzter Buchstabe
      - Beispiel für Attribut: Tabelle kunde ergibt Präfix kue_

- Beispiele ENTITY:
  - PERSON -> PEN
  - SCHULTYP -> STP
- Beispiele ATTRIBUTE:
  - Vorname -> VNE oder auch VOE
  - Nachname -> NNE oder auch NAE
  - Schulbezeichnung -> SBG
- Empfehlung: Tabellennamen lieber ausschreiben!

- Namensvergabe Schule:
  - Ein Wort als Tabellenname
    - Die ersten 3 Buchstaben
  - Bei zwei zusammengesetzten Tabellen-namen:
    - Die ersten zwei Buchstaben
  - Bei mehreren zusammengesetzen Tabellen-namen:
    - Der erste Buchstabe

  jeweils gefolgt von einem Unterstrich.

  Beispiele:

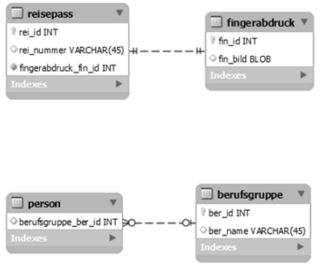
  PERSON: per_id, PERSON_ADRESSE: pead_id, PLZ_ORT_STRASSE: pos_id

Relationship

- Beziehung zwischen den Entitäten
- Eine Beziehung
  - besteht immer zwischen zwei Entitäten (oder zu sich selbst)
  - zeigt auch an, in welcher Beziehung Entitäten zueinander stehen
  - hat immer zwei Seiten
  - ist an beiden Seiten benannt / bezeichnet
  - hat eine Optionalität (Muss oder Kann eine Beziehung zu einem Datensatz bestehen)
  - hat eine Kardinalität (Wie oft kann ein Bezugswert – FK – zu einer anderen Entität bestehen)

Beziehungen

- 1:1
  - Ein DS der Tabelle A (PK) ist mit genau einem DS der Tabelle B verknüpft.
- 0:N
  - Ein DS der Tabelle A (PK) ist mit keinem oder mehreren DS der Tabelle B verknüpft. Ein DS der Tabelle B bezieht sich auf genau einen DS der Tabelle A.



- 1:N
  - Ein DS der Tabelle A (PK) ist mit einem oder mehreren DS der Tabelle B (FK) verknüpft. Ein DS der Tabelle B bezieht sich auf genau einen DS der Tabelle A.
- M:N
  - Ein DS der Tabelle A (PK) ist mit mehreren DS der Tabelle B verknüpft. Ein DS der Tabelle B bezieht sich auf mehrere DS der Tabelle A.