Script **generated by TTT**

Title: Einf_HF (22.04.2013)

Date: Mon Apr 22 14:14:41 CEST 2013

Duration: 91:53 min

Pages: 22

Datenbanksysteme

Informationsgesellschaft führt zu immer größeren Datenmengen ⇒ Speicherung der Daten in Datenbanksystemen.

Allgemeines

Anwendungsübergreifende Datenspeicherung und -organisation.

Datenbank = Datenmenge mit Strukturmodell (Datenbankschema, Datenmodell)

"Datenbanksystem" (DBS): Sammlung gespeicherter Daten (Datenbank), Software für Speicherung und zum Zugriff.

Operationen: Eintragen, Löschen, Suchen, Verknüpfen von Daten. Wichtig: Daten mit langer Lebensdauer, große Datenmengen (GBytes, TBytes).

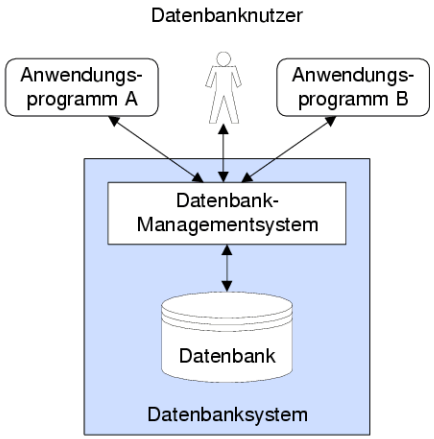
Logischer Aufbau

Datenbankmanagementsystem

Beispiele aus der Praxis

Anforderungen an ein DBS

Logischer Aufbau



Datenbankmanagementsystem

Das Datenbankmanagementsystem (DBMS) ist die Gesamtheit aller Programme für den Umgang mit den Daten. Es ist verantwortlich für

die sichere und einheitliche Verwaltung persistenter (langlebiger) Daten,

den Datenaustausch zwischen Datenbank und Anwendungsprogrammen,

die Verhinderung von unkontrollierten Zugriffen auf den Datenbestand und

die effiziente Zugriffsmöglichkeit auf die in der Regel sehr großen Datenbestände.

Universitätsdatenbank

Sammlung aller für Verwaltungsaufgaben an einer Universität benötigten Daten. I.a. Gliederung in Fachbereiche mit zugeordneten Studenten, Professoren, Mitarbeitern. Studenten belegen Vorlesungen von Professoren, legen Prüfungen ab. Typische Anwendungen: Immatrikulation, Rückmeldung, Ausfertigen von Studentenausweisen, Studienbescheinigungen, Stundenplänen. Raumplanung, Ausstellen von Zeugnissen, Exmatrikulation, Statistiken.

Datenbank einer Fluggesellschaft

Fluggesellschaft fliegt verschiedene Flughäfen an. Flugstrecken, Flugzeugtypen, Personal. Piloten haben Flugscheine jeweils nur für einige wenige Flugzeugtypen. Typische Anwendungen: Flugbuchungen, Anfertigen Passagierlisten, Personaleinsatzplanung, Materialeinsatzplanung, Flugplanerstellung, Überwachung der Wartefristen, Gehaltsabrechnung.

Kontrolle über Daten

Kontrolle der Datenintegrität

Automatisierte Zugriffskontrollen (Datenschutz).

Erhaltung der logischen Datenintegrität, d.h. Überprüfung der Richtigkeit von Daten.

Notwendigkeit des kontrollierten Mehrbenutzer-Betriebs, keine gegenseitige Störung durch parallel zugreifenden Benutzer.

Leichte Handhabbarkeit der Daten

Einfache Beschreibung der logische Aspekte der Daten (Bedeutung, Zusammenhänge), leicht erlernbare Sprache zum Zugriff bzw. Speichern von Daten in der Datenbank (z.B. SQL).

Datenunabhängigkeit

Anwendungen unabhängig von Organisation und Speicherungen der Daten.

Alle Daten können/müssen gemeinsam benutzt werden

keine verstreuten privaten Daten

Entwicklung neuer Anwendungen auf der existierenden Datenbank

Querauswertungen aufgrund inhaltlicher Zusammenhänge

Erweiterung und Anpassung der Datenbank

Eliminierung der Redundanz, damit Vermeidung von Inkonsistenzen

Datenbankadministrator hat zentrale Verantwortung für Daten

Kontrolle über Daten

Kontrolle der Datenintegrität

Automatisierte Zugriffskontrollen (Datenschutz).

Erhaltung der logischen Datenintegrität, d.h. Überprüfung der Richtigkeit von Daten.

Notwendigkeit des kontrollierten Mehrbenutzer-Betriebs, keine gegenseitige Störung durch parallel zugreifenden Benutzer.

Leichte Handhabbarkeit der Daten

Einfache Beschreibung der logische Aspekte der Daten (Bedeutung, Zusammenhänge), leicht erlernbare Sprache zum Zugriff bzw. Speichern von Daten in der Datenbank (z.B. SQL).

Datenunabhängigkeit

Anwendungen unabhängig von Organisation und Speicherungen der Daten.

Generated by Targeteam

Datenbanken und Informationssysteme

- Fragestellungen des Abschnitts:
  - Was unterscheidet Dateisysteme von Datenbanksystemen?
  - Wie kann die Struktur der Daten in einem Datenbanksystem dargestellt werden?
  - Was sind relationale Datenbanksysteme?
  - Was sind die grundlegenden Konstrukte von HTML?

Dateisysteme

Datenbanksysteme

Datenbankentwurf

Relationale Datenbanksysteme

WWW - Informationssystem

Wie entwirft man eine Datenbank?

Man geht schrittweise vor

1. Erstellung eines Modells der Daten und der Beziehungen zwischen diesen Daten ⇒ Datenmodell, z.B. ER-Modell
2. Abbildung des Datenmodells auf eine Menge von Tabellen (relationales Modell)
3. Normalisierung: Entfernen aller Redundanzen. Eine Redundanz liegt vor, wenn dieselbe Information an mehreren Stellen eingetragen ist, z.B. die PLZ in verschiedenen Tabellen der Datenbank. Bei einer Änderung der PLZ müssen alle Stellen in der Datenbank geändert werden; bei Vergessen einer Änderungsstelle können Fehler auftreten.
4. Eintragen der Daten in die Tabellen.
5. Realisierung der möglichen Abfragen ⇒ Sichten auf die Daten in den Tabellen.

Datenmodell

Die Daten in einem Datenbanksystem bilden ein abstrahiertes Spiegelbild einer Miniwelt. ⇒ Verwendung von Datenmodellen.

Definition: Ein **Datenmodell** ist ein (oft mathematischer) Formalismus

mit einer Notation zur Beschreibung und Definition der Datenobjekte und deren Struktur,

einer Menge von Operationen zur Manipulation der Daten.

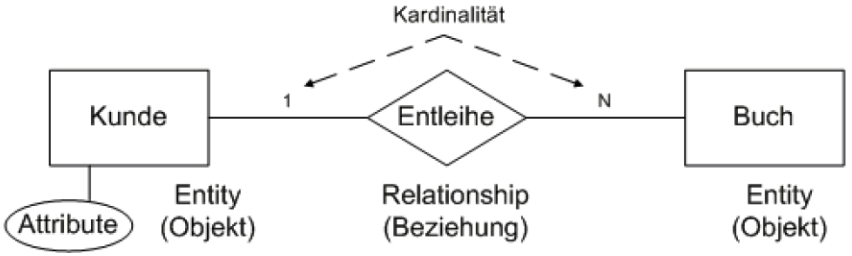
Nicht beschrieben werden Abläufe, Interaktion zwischen Objekten oder zeitliches Verhalten.

Beispiele

Entity-Relationship-Modell (ER-Modell), Objektorientiertes Modell, Hierachisches Modell.

ER-Modell

Entity-Relationship-Modell eignet sich zur Darstellung des Datenbankschemas.



Graphisches Hilfsmittel zur semantischen Modellierung eines Anwendungsgebietes, d.h. zum Entwurf einer Datenbank, unabhängig vom konkreten DBS.

Grundidee: Reale Welt (Mini-Welt) läßt sich durch Objekte und Beziehungen zwischen Objekten beschreiben (Objekte: Entities, Beziehungen: Relationships).

Gleichartige Entities (Objektinstanzen) werden zu Entity-Typen (vergleichbar Klassen) bzw. Relationships zu Relationship-Typen zusammengefasst.

Beispiele: Entity-Typ: "Bibliotheksbenutzer", "Buch", und Relationship-Typ: "entleiht".

Attribute bei Entities und Relationships, z.B. "Name" bei Entity "Bibliotheksbenutzer", "Entleihdatum" bei Relationship "leiht aus".

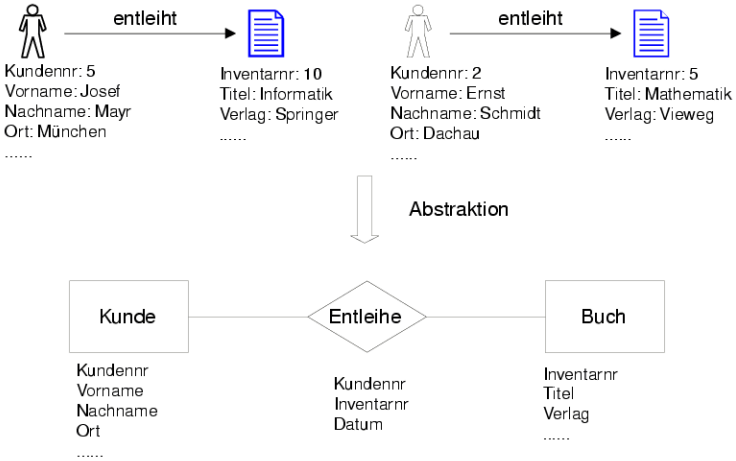
Entity-Typ

Relationship-Typ

Erweiterung auf n-stellige Beziehungen

Entity-Typ

Ein Entity-Typ fasst eine Menge von gleichartigen Objektinstanzen, die durch gleiche Attribute charakterisiert sind, zusammen.



[Schlüssel](#)

Schlüssel

Wahl eines oder mehrerer Attribute, die eine Entity (Objektinstanz) ***eindeutig*** identifiziert, z.B.

Kundennr identifiziert eindeutig einen einzelnen Kunden.

⇒ Primärschlüssel.

Ein Primärschlüssel kann aus mehr als einem Attribut bestehen.

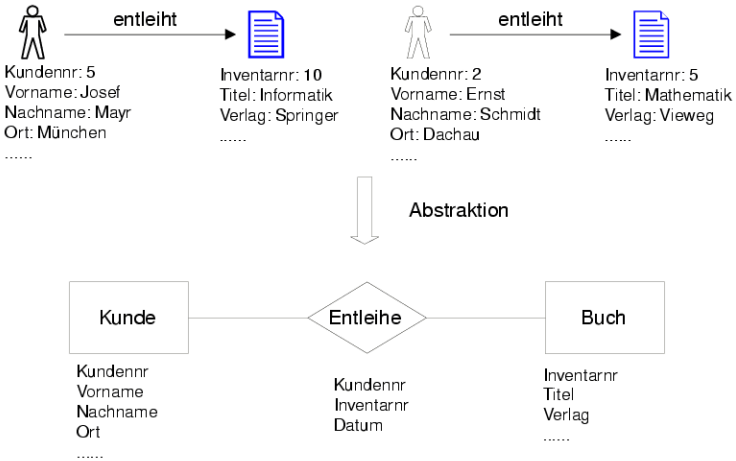
Die Entscheidung für einen bestimmten Primärschlüssel geschieht während der Modellierung des Anwendungsbereiches.

falls kein Attribut eindeutig ist, wird ein ***künstliches*** Attribut eingeführt, z.B. eine fortlaufende Nummer für jeden Eintrag.

Primärschlüssel werden im ER-Modell durch Unterstreichung gekennzeichnet.

Entity-Typ

Ein Entity-Typ fasst eine Menge von gleichartigen Objektinstanzen, die durch gleiche Attribute charakterisiert sind, zusammen.



[Schlüssel](#)

Relationship-Typ

Ein Relationship-Typ umfasst die Menge gleichartiger Relationships. Ein Relationship-Typ R stellt die Beziehung zwischen Entity-Typen E1 und E2 her, d.h. R ⊆ E1 * E2.

Kardinalität von Relationship-Typen zur Spezifikation der Art der Beziehung zwischen Entities.

Man kann verschiedene Kardinalitätsarten unterscheiden

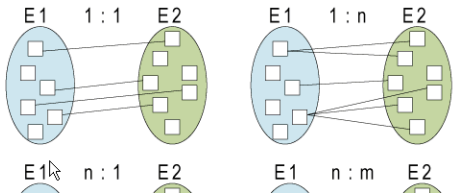
1:1 , ein Entity aus E1 kann mit höchstens einem Entity aus E2 über R in Beziehung stehen kann und umgekehrt.

n:1 , ein Entity aus E1 kann mit höchstens einem Entity aus E2 , aber ein Entity aus E2 mit beliebig vielen Entities aus E1 über R in Beziehung stehen.

1:n , ein Entity aus E1 kann mit beliebig vielen Entities aus E2 , aber ein Entity aus E2 mit höchstens einem Entity aus E1 über R in Beziehung stehen.

Beispiel: 1 Kunde kann n Bücher ausleihen; 1 Buch kann nur von einem Kunden ausgeliehen werden.

n:m , ein Entity aus E1 kann mit beliebig vielen Entities aus E2 über R in Beziehung stehen und umgekehrt.



Relationship-Typ

kanks. Produkt

1:1 , ein Entity aus E1 kann mit höchstens einem Entity aus E2 über R in Beziehung stehen kann und umgekehrt.

n:1 , ein Entity aus E1 kann mit höchstens einem Entity aus E2 , aber ein Entity aus E2 mit beliebig vielen Entities aus E1 über R in Beziehung stehen.

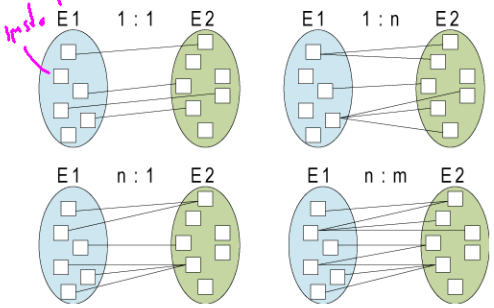
1:n , ein Entity aus E1 kann mit beliebig vielen Entities aus E2 , aber ein Entity aus E2 mit höchstens einem Entity aus E1 über R in Beziehung stehen.

Beispiel: 1 Kunde kann n Bücher ausleihen; 1 Buch kann nur von einem Kunden ausgeliehen werden.

n:m , ein Entity aus E1 kann mit beliebig vielen Entities aus E2 über R in Beziehung stehen und umgekehrt.

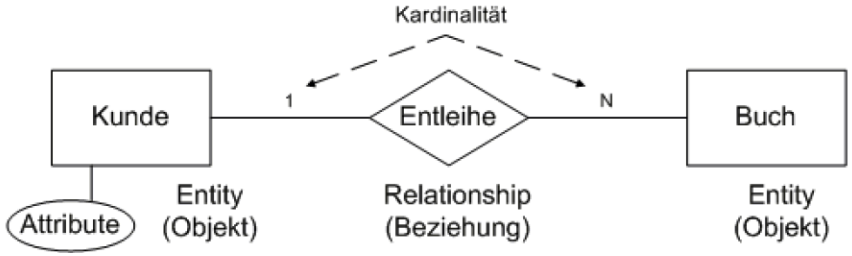
Typ

Instd.



ER-Modell

Entity-Relationship-Modell eignet sich zur Darstellung des Datenbankschemas.



Graphisches Hilfsmittel zur semantischen Modellierung eines Anwendungsgebietes, d.h. zum Entwurf einer Datenbank, unabhängig vom konkreten DBS.

Grundidee: Reale Welt (Mini-Welt) läßt sich durch Objekte und Beziehungen zwischen Objekten beschreiben (Objekte: Entities, Beziehungen: Relationships).

Gleichartige Entities (Objektinstanzen) werden zu Entity-Typen (vergleichbar Klassen) bzw. Relationships zu Relationship-Typen zusammengefasst.

Beispiele: Entity-Typ: "Bibliotheksbenutzer", "Buch", und Relationship-Typ: "entleiht".

Attribute bei Entities und Relationships, z.B. "Name" bei Entity "Bibliotheksbenutzer", "Entleihdatum" bei Relationship "leiht aus".

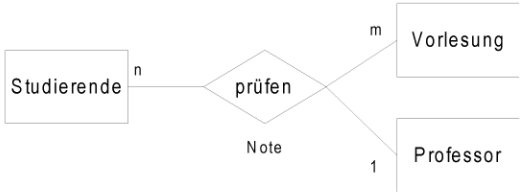
Entity-Typ

Relationship-Typ

Erweiterung auf n-stellige Beziehungen

Erweiterung auf n-stellige Beziehungen

Relationship-Typen können nicht nur 2-stellig (binär), sondern auch n-stellig sein. Beispiel für einen 3-stellige Relationship-Typen

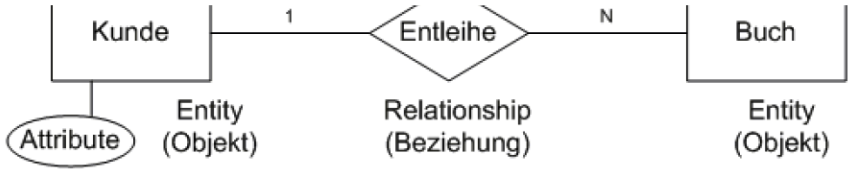


Darstellung der Funktionalität der Beziehung

prüfen: Studierende * Vorlesung ⇒ Professor.

Mehrere Studierende können über den Stoff von mehreren Vorlesungen von einem Professor geprüft werden.

ER-Modell



Graphisches Hilfsmittel zur semantischen Modellierung eines Anwendungsgebietes, d.h. zum Entwurf einer Datenbank, unabhängig vom konkreten DBS.

Grundidee: Reale Welt (Mini-Welt) läßt sich durch Objekte und Beziehungen zwischen Objekten beschreiben (Objekte: Entities, Beziehungen: Relationships).

Gleichartige Entities (Objektinstanzen) werden zu Entity-Typen (vergleichbar Klassen) bzw. Relationships zu Relationship-Typen zusammengefasst.

Beispiele: Entity-Typ: "Bibliotheksbenutzer", "Buch", und Relationship-Typ: "entleiht".

Attribute bei Entities und Relationships, z.B. "Name" bei Entity "Bibliotheksbenutzer", "Entleihdatum" bei Relationship "leiht aus".

Entity-Typ

Relationship-Typ

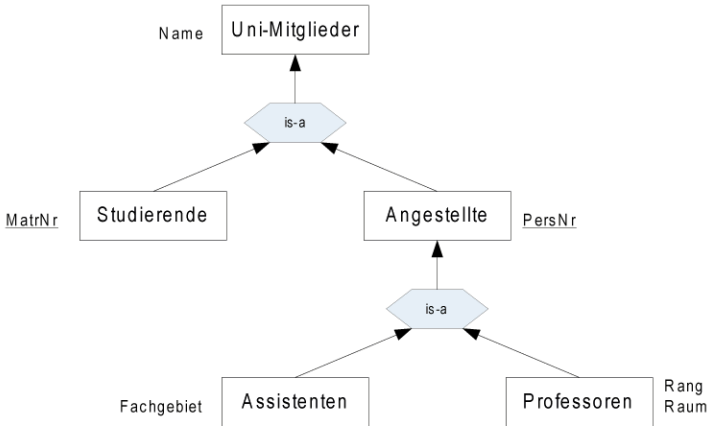
Erweiterung auf n-stellige Beziehungen

Modellierung einer Datenbank findet auf der Ebene von Entity-Typen, Relationship-Typen und Attributen statt.

Generalisierung

Generalisierung

Generalisierung dient beim Konzeptentwurf zur übersichtlicheren Strukturierung der Entity-Typen



Relationale Datenbanksysteme

Entity-Relationship-Diagramm: logisches Modell einer Datenbank. Für Implementierung in DBS: physikalisches Modell nötig.

Beispiel: relationales Datenbankmodell.

Sichten

Verwendung von Sichten zur Auswahl einer Teilmenge der Tabellendaten.

Beispiel für eine Sicht: Vorname und Nachname von Kunden, die in München wohnen.

Views sind damit nichts anderes als benannte Such-Abfragen (z.B. SQL-Anfragen in MS ACCESS mit Namen).

Beispielsysteme

mySQL (Open Source), Microsoft Access, Microsoft SQL Server, Oracle, DB2 (IBM), Sybase, Informix

Relationales Modell

Schema : legt die Struktur der in der Datenbank gespeicherten Daten fest. Darstellung mit Hilfe von Relationen

Relation R ⊆ Wertebereich (Attribut 1) * ... * Wertebereich (Attribut n)

- Beispiel der Relation Telefonbuch

  Telefonbuch ⊆ Text * Text * Zahl

  Darstellung als Schema

  Telefonbuch: {[Name: Text, Adresse: Text, ***Telefonnr: Zahl]*** }

- Relationale Darstellung von Entity-Typen:

  Kunde: {[***Kundennr: Zahl*** , Vorname: Text, Nachname: Text, Ort: Text]}

  Kundennr ist der Primärschlüssel zur Identifizierung der einzelnen Kunden.

- Relationale Darstellung von Relationship-Typen:

  entleiht: {[***Kundennr: Zahl*** , ***Inventarnr: Zahl*** , Datum: Datum]}

  Kundennr und Inventarnr dient zur Identifizierung eines Entleihvorgangs.