Datenbanken

6. Vorlesung: Einführung Entity-Relationship-Modellierung

Prof. Dr. Andreas de Vries

Fachhochschule Südwestfalen, Standort Hagen

Version: 4. November 2019

Inhaltsübersicht

Beispiel einer Umsatztabelle

- Ein produzierendes Unternehmen will eine Datenbank nutzen, um seine Umsätze zu speichern.
- Es setzt dafür eine Tabelle mit folgendem Relationentyp ein:

```
CREATE TABLE umsaetze (
  id           int SERIAL,
  datum        date,
  artikel      varchar(50),
  einzelpreis decimal(10,2),
  anzahl       int,
  kunde        varchar(50),
  wohnort      varchar(50),
  umsatz       decimal(10,2)
) DEFAULT CHARSET=UTF8;
```

Beispiel einer Umsatztabelle

- Alles funktioniert wunderbar, das Unternehmen speichert die Umsätze für Oktober und November 2019:

ID	Datum	Artikel	Einzelpreis	Anzahl	Kunde	Wohnort	Umsatz
1	2019-11-06	Hammer	9.90	2	Anna	Hagen	19.80
2	2019-11-02	Säge	4.95	3	Bert	Meschede	14.85
3	2019-10-03	Hammer	9.90	1	Otto	Soest	9.90
4	2019-10-04	Mottek	12.95	1	Anna	Hagen	12.95
5	2019-11-06	Säge	4.95	2	Doro	Iserlohn	9.90
6	2019-10-06	Zange	3.90	4	Anna	Hagen	15.60
7	2019-11-06	Zange	3.90	3	Doro	Iserlohn	11.70
8	2019-10-06	Hammer	9.90	1	Otto	Soest	9.90
9	2019-11-05	Säge	4.95	4	Anna	Hagen	19.80
10	2019-11-06	Mottek	12.95	2	Anna	Hagen	25.90
11	2019-10-04	Mottek	12.95	3	Bert	Meschede	38.85
12	2019-10-06	Zange	3.90	1	Bert	Meschede	3.90

Beispiel einer Umsatztabelle

Die Geschäftsführung ist sehr zufrieden, sie kann alle ihr wichtigen Auswertungen durchführen lassen, zum Beispiel ...

- Welcher Kunde hat welchen Artikel gekauft?

```
SELECT kunde, artikel FROM umsaetze GROUP BY kunde, artikel;
```

- Welcher Gesamtumsatz wurde im November 2019 erzielt?

```
SELECT sum(umsatz) FROM umsaetze WHERE datum BETWEEN '2019-11-01' AND '2019-11-30'
```

- In welchen Ort wurde welcher Artikel wie oft verkauft?

```
SELECT wohnort, artikel, sum(anzahl) FROM umsaetze GROUP BY wohnort, artikel
```

- Welcher Umsatz wurde je Artikel erzielt, sortiert nach Gesamtumsatz?

```
SELECT artikel, sum(umsatz) AS gesamt FROM umsaetze GROUP BY artikel
ORDER BY gesamt DESC;
```

- Welcher Artikel wurde im Oktober am meisten verkauft?

```
SELECT artikel, sum(anzahl) AS gesamt FROM umsaetze
WHERE datum BETWEEN '2019-10-01' AND '2019-10-31'
GROUP BY artikel HAVING gesamt >= ALL (
  SELECT sum(anzahl) FROM umsaetze
  WHERE datum BETWEEN '2019-10-01' AND '2019-10-31'
  GROUP BY artikel
);
```

Probleme mit Daten in einer einzigen Tabelle

Es scheint also: Alles gut! Aber im Lauf der Zeit zeigen sich Probleme:

- Beim Einfügen eines neuen Umsatzes weiß ein Sachbearbeiter nicht den Wohnort des Kunden:

```
INSERT INTO umsaetze (datum, artikel, einzelpreis, anzahl, kunde, umsatz)
VALUES ('2019-10-21', 'Hammer', 9.90, 1, 'Otto', 9.90);
```

  Damit wird das Ergebnis unserer obigen Abfrage „In welchen Ort wurde welcher Artikel wie oft verkauft?“ fehlerhaft, obwohl die Datenbank den Ort ja „kennt“.

  → ***Gefahr inkonsistenter Daten***

- Es wird unnötiger Speicherplatz verbraucht, denn dieselbe Information (Artikel-Einzelpreis, Kunde-Wohnort) wird bei *jedem einzelnen* Umsatz gespeichert.

  → ***ineffiziente Speicherung***

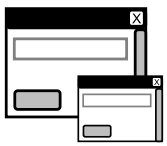
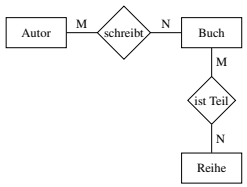
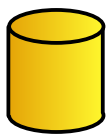
- Was ist die Ursache dieser Probleme?
  → ***redundante Daten!***

- Was ist die Lösung?
  → ***mehrere Tabellen!***

Inhaltsübersicht

Datenmodellierung

- Die Programmierung mehrerer Tabellen ist im realen Kontext sehr komplex.
- Notwendig ist daher eine sorgfältige Strukturierung der Daten und ihres Kontextes, d.h. ein „abstrahiertes“ ***Datenmodell*** zur ...
  - Strukturierung der Anforderungen, um sie programmierbar zu machen;
  - Erleichterung der Kommunikation zwischen Anwendern und Entwicklern
  - Strukturierung der Daten, um sie effizient speichern und manipulieren zu können.
- Datenmodellierung ist die Grundlage für den *logischen Datenbankentwurf*, also den Entwurf der Tabellen mit Attributen und Primärschlüsseln.

Externe Ebene		Benutzeroberflächen, Views
Logische Ebene		ER-Diagramme, Datenmodell, Normalisierung
Interne Ebene		Speicherformate und -verfahren, Datenzugriffssteuerung

- Datenmodelle gehören damit zur logischen Ebene der ANSI-SPARC-Architektur

Entity-Relationship-Diagramme

- Als Methodik zur Datenmodellierung verwendet man üblicherweise ***Entity-Relationship-Diagramme***
  - ... zur besseren Anschauung und
  - ... zur besseren Übersicht
- Zusammen mit einer Beschreibung der darin verwendeten Elemente bilden sie das *Entity-Relationship-Modell (ERM)*.
- Grundlegende Idee: eine ***Entität*** ist das abstrahierte Modell einer Tabelle

- Damit lassen sich aus einem ERM die Tabellen der relationalen Datenbank eindeutig ableiten.
- Ein Entity-Relationship-Diagramm besteht aus drei Elementen:
  - dem ***Entitätstyp***
  - der ***Beziehung***
  - der ***Kardinalität***

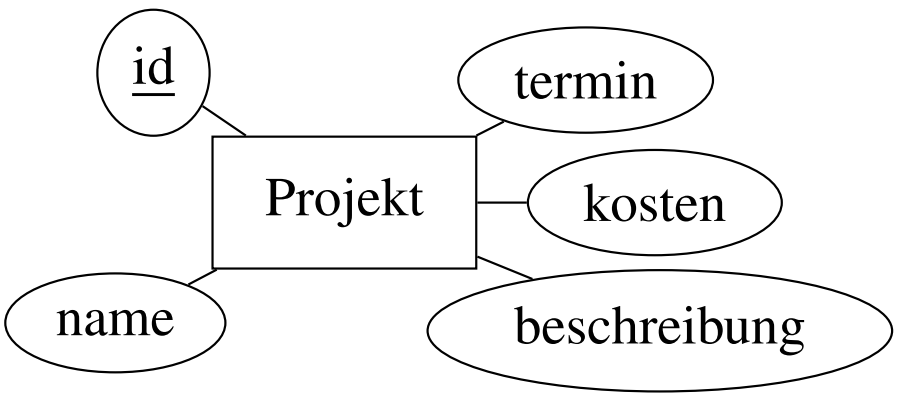
Inhaltsübersicht

Entität und Entitätstyp

- Eine ***Entität (entity)*** ist ein individuelles Objekt der zu modellierenden Welt.
- Das kann ein realer Gegenstand sein, z.B.:
  - ein Exemplar *Asterix der Gallier*,
  - der Angestellte Otto Meier
  - der Hammer mit der Artikelnummer 4711,
- ... oder ein immaterieller Vorgang oder ein Konstrukt, wie z.B.:
  - die Bestellung vom 1. April,
  - der Liefervertrag Nr. 123,
  - das Unternehmen ABC AG.
- Der ***Entitätstyp (entity type)*** ist der Oberbegriff oder die Kategorie einer oder mehrerer gleichartiger Entitäten
  - ... z.B.: *Buch*, *Angestellter*, *Artikel*, *Projekt*, *Bestellung*, *Vertrag*, *Unternehmen* ...
- Ein Entitätstyp wird im ER-Diagramm mit einem Rechteck dargestellt, in dem der Name des Typs steht:

Attribute von Entitäten

- Eine Entität hat in der Regel mehrere ***Attribute***.
- Eines davon ist der ***Primärschlüssel*** der Entität, der sie unter allen anderen Entitäten stets eindeutig identifiziert.
- Häufig werden die Attribute als kleine Ellipsen mit dem Entitätstyp verbunden oder alternativ in dem Rechteck des Entitätstypen aufgelistet:

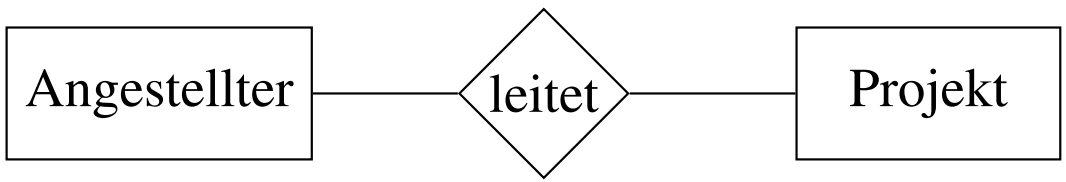


Projekt
- id
- name
- termin
- kosten
- beschreibung

- Entitätstypen modellieren also die Tabellen einer relationalen Datenbank,
- Attribute modellieren der Entitäten die Spalten der Tabelle,
- Eine Entität entspricht einem Datensatz, d.h. einer Zeile der Tabelle.

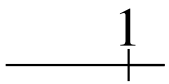
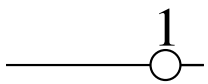
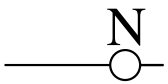
Beziehungen

- Eine ***Beziehung (relationship)*** im ERM ist der Zusammenhang zwischen zwei Entitäten.
- Sie wird in einem ER-Diagramm mit einem Verb in eine Raute zwischen zwei Entitätstypen dargestellt, z.B.:

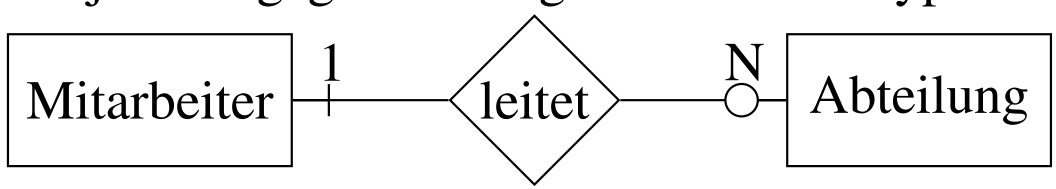


Kardinalitäten

- Eine ***Kardinalität (cardinality)*** drückt aus, wieviel Entitäten des einen Typs mit wieviel Entitäten des anderen eine Beziehung maximal und mindestens haben.
- Für die Maximalzahl wird 1 oder N für „eins“ oder „mehrere“ über der Beziehung geschrieben, und ein Kreis oder ein Strich an der Beziehungslinie für die Mindestzahl. D.h. es gibt 4 mögliche Kardinalitäten:

Symbol				
Bezeichnung	1	C	M	CM

- Im Englischen: Bezeichnungen 1, C (für “choice”), M (für “many”) und CM.
- Ein Strich ⇒ eine *Muss-Beziehung*, ein Kreis ⇒ eine *Kann-Beziehung*.
- In einem ER-Diagramm wird für eine konkrete Entität des einen Entitätstyps die Kardinalität an dem jeweils gegenüberliegenden Entitätstypen notiert:



- Ein Mitarbeiter *kann* mehrere Abteilungen leiten, eine Abteilung *muss* von genau einem Mitarbeiter geleitet werden.
  ⇒ eine 1-CM-Beziehung, oder: eine Beziehung „1 zu N muss-kann“.

Das Beispiel der Umsatztabelle – Entitätstypen

- Wie können wir das Datenmodell unserer Umsatzdatenbank entwerfen?

ID	Datum	Artikel	Einzelpreis	Anzahl	Kunde	Wohnort	Umsatz
1	2019-11-06	Hammer	9.90	2	Anna	Hagen	19.80
2	2019-11-02	Säge	4.95	3	Bert	Meschede	14.85
3	2019-10-03	Hammer	9.90	1	Otto	Soest	9.90
4	2019-10-04	Mottek	12.95	1	Anna	Hagen	12.95
⋮							

- *Leitfrage:* Welche der relevanten Daten gehören zusammen?
  ⇒ Entitätstypen und ihre Attribute

Kunde
- name
- wohnort

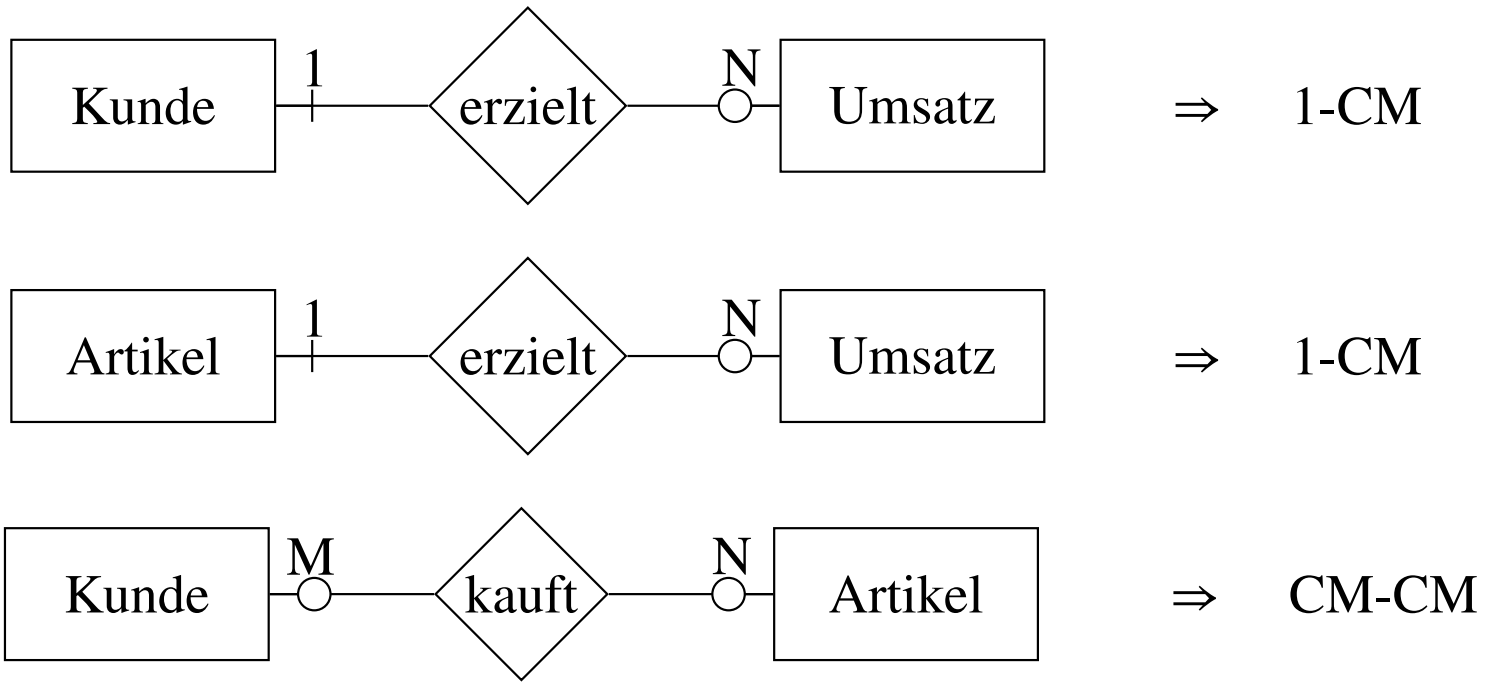
Artikel
- name
- einzelpreis

Umsatz
- datum
- kunde.name
- artikel.name
- anzahl
- umsatz

Das Beispiel der Umsatztabelle – Beziehungen

- Welche Beziehungen haben wir zwischen den Entitäten?
- Dazu: Welche Entitätspaare gibt es?

  Kunde – Umsatz, Artikel – Umsatz, Kunde – Artikel



- Wie aber können wir die *Beziehungen* zwischen diesen Tabellen in SQL programmieren?

Fortsetzung folgt . . .

Inhaltsübersicht

Das war's für heute!

Wir haben heute behandelt:

- Probleme bei der Speicherung komplexer Daten in einer einzigen Tabelle, insbesondere durch Redundanz: inkonsistente Daten und ineffiziente Speicherung
- Beherrschung komplexer Daten durch Modellierung
- Entity-Relationship-Diagramme als Werkzeug zur Datenmodellierung

Haben Sie noch Fragen