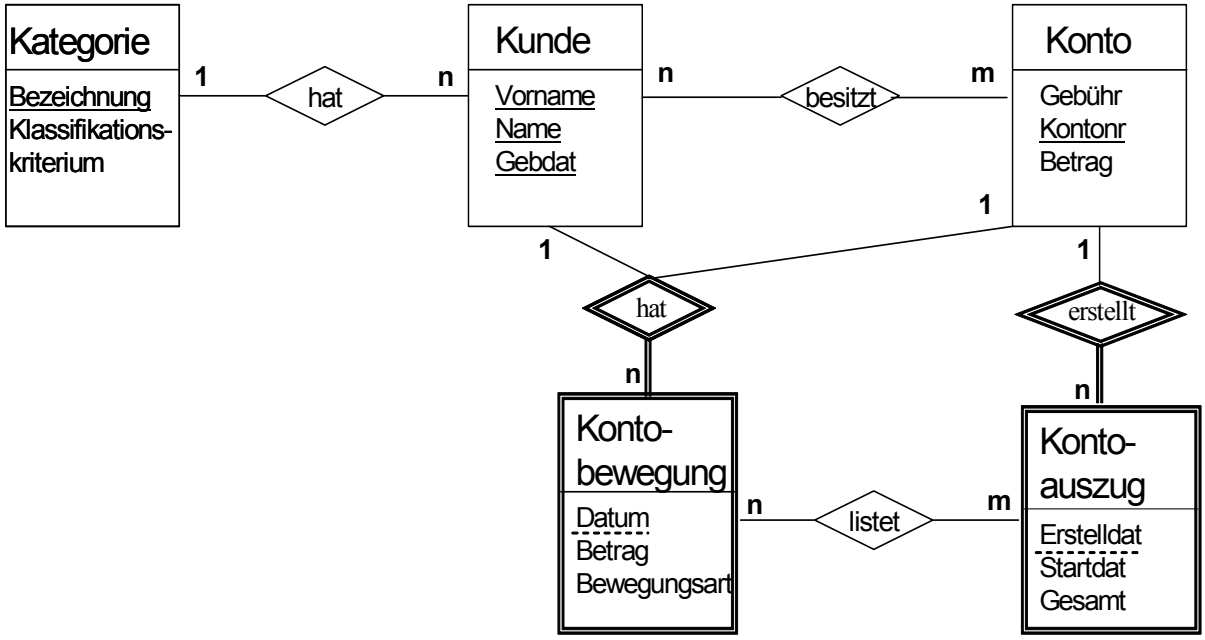
Technische Universität München
Fakultät für Informatik
Prof. R. Bayer, Ph.D.
Dipl.-Inform. Michael Bauer
Dr. Gabi Höfling

WS 2003/04,
Datenbanksysteme I
Lösungsblatt 4

17.11. 2003

Abbildung E/R nach relational - Beispiel: Kontoführung



Aufgabe 1: Umsetzen eines E/R-Diagramms in ein relationales Schema

Setzen Sie das oben angegebene E/R-Schema in ein relationales Modell um. Beachten Sie dabei, dass zu jeder relationalen Tabelle ein Schlüssel und zu jedem Attribut ein Typ gehört. Gehen Sie dabei in folgenden Teilschritten vor:

1. Setzen Sie alle starken Entitäten um.
2. Setzen sie alle schwachen Entities um. Welcher Unterschied ergibt sich zu 1.?
3. Setzen sie alle Beziehungen um.
4. Können in dem so entstandenen Relationenschema Tabellen zusammengefaßt werden?
5. Bestimmen Sie für alle Tabellen die Fremdschlüssel. Auf welche Tabellen verweisen sie?

1.1. Starke Entitäten:

Kategorie: (Bezeichnung:string, Klassifikationskriterium:string)
Kunde: (Vorname:string, Name:string, Gebdat:date)
Konto: (Kontonr: integer, Betrag:numeric, Gebühr:numeric)

1.2. Schwache Entitäten:

Kontobewegung: (Datum:date, Vorname:string, Name:string, Gebdat:date, Kontonr:integer, Betrag: numeric, Bewegungsart: string)
Kontoauszug: (Erstelldat:date, Kontonr:integer, Startdat:date, Gesamt:numeric)

1.3. Beziehungen:

hatKateg: (Vorname:string, Name:string, Gebdat:date, Bezeichnung:string)
besitzt: (Vorname:string, Name:string, Gebdat:date, Kontonr:integer)
listet: (Erstelldat:date, Kontonr:integer, Datum:date, Vorname:string,
Name:string, Gebdat:date)

Für *hatKotobewegung* und *erstellt* werden keine Tabellen angelegt, da sie die jeweils starken mit den zugehörigen schwachen Entitäten verbinden.

1.4. Zusammenfassen

Zusammenfassen kann man Tabellen in der Regel dann, es sich um eine 1:n bzw. 1:1 Beziehung handelt und die Tabellen gleiche Schlüssel aufweisen. In unserem Beispiel ist dies bei den Tabellen *Kunde* und *hatKateg* der Fall:

Kunde: (Vorname:string, Name:string, Gebdat:date)
hatKateg: (Vorname:string, Name:string, Gebdat:date, Bezeichnung:string)
wird zu
Kunde: (Vorname:string, Name:string, Gebdat:date, Bezeichnung:string)

1.5. Fremdschlüssel

Kunde:Bezeichnung verweist auf Kategorie:Bezeichnung
Kontobewegung:Vorname, Name, Gebdat verweist auf Kunde:Vorname, Name, Gebdat
Kontobewegung:Kontonr verweist auf Konto:Kontonr
besitzt:Vorname, Name, Gebdat verweist auf Kunde:Vorname, Name, Gebdat
besitzt:Kontonr verweist auf Konto:Kontonr
listet:Erstelldat, Kontonr verweist auf Kontoauszug:Erstelldat, Kontonr
listet:Datum, Kontonr verweist auf Kontobewegung:Datum, Kontonr
listet:Vorname, Name, Gebdat verweist auf Kunde:Vorname, Name, Gebdat

Aufgabe 2: Alternative Schlüsselumsetzung

Neben der klassischen Darstellung von Primärschlüsseln als Werte aus der Attributmenge gibt es noch die Möglichkeit, den Elementen künstlich erzeugte eindeutige Identifikatoren zuzuweisen, z.B. eine fortlaufende Nummer. Wann dies in einem relationalen Schema sinnvoll sein kann, zeigt diese Aufgabe:

1. Betrachten Sie dazu die Entität Kunde und seine Beziehungen. Erstellen Sie eine Umsetzung, die eine künstlich erzeugte eindeutige Kundennummer verwendet.
2. Schätzen Sie nun ab, wie viel Speicherplatz sowohl in der Modellierung aus Aufgabe 1 als auch in der neuen Modellierung benötigt wird. Es gelten folgende Abschätzungen:

   Anzahl der Kunden: 1.000.000

   Anzahl der Konten: 5.000.000

   Anzahl der Kontobewegungen: 100.000.000 (im Jahr)

   Kontobewegungen pro Kontoauszug: 5

   Für den Kundennamen etc. können sie einen durchschnittlichen Speicherbedarf von 15 Byte annehmen. Zur Darstellung von Zahlentypen sollte immer der nächstgrößere Standardtyp verwendet werden, also 16, 32, 64, 128 bit (Datum als 16bit-Zahl, Betrag als 48bit Zahl (Datentyp Numeric)).

2.1. Umsetzung mit einer künstlich erzeugten eindeutigen Kundennummer:

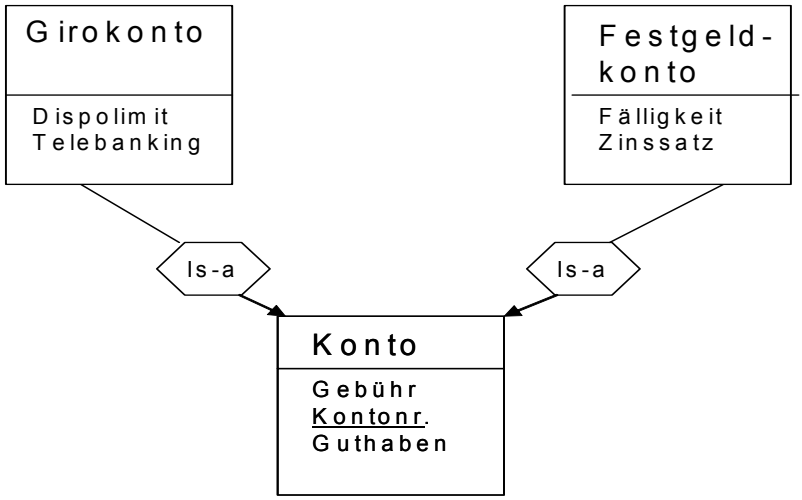
Kunde:	(Kundennr:integer, Vorname:string, Name:string, Gebdat:date, Bezeichnung:string)
Kontobewegung:	(Datum:date, Kundennr:integer, Kontonr:integer, Betrag: numeric, Bewegungsart: string)
besitzt:	(Kundennr:integer, Kontonr:integer)
listet:	(Erstelldat:date, Kontonr:integer, Datum:date, Kundennr:integer)

2.2. Speicherbedarf pro Tupel in Bytes (in gleicher Attributreihenfolge wie in 2.1.).

Kunde klassisch:	15+15+2+15 = 47
Kunde ID:	4+15+15+2+15 = 51
Kontobewegung klassisch:	2+15+15+2+4+6+15 = 59
Kontobewegung ID:	2+4+4+6+15 = 31
besitzt klassisch:	15+15+2+4 = 36
besitzt ID:	4+4 = 8
listet klassisch:	2+4+2+15+15+2 = 40
listet ID:	2+4+2+4 = 12

Tabelle /Platz	Pro Eintrag klassisch	Pro Eintrag ID	Gesamt klassisch	Gesamt ID
Kunde	47	51	$0.47*10^8$	$0.51*10^8$
Kontobewegung	59	31	$59*10^8$	$30*10^8$
besitzt	36	8	$3.6*10^8$	$0.8*10^8$
listet	40	12	$8*10^8$	$2.4*10^8$
Gesamt			7.1 GB	3.3 GB

Aufgabe 3: Modellierungsalternativen für Generalisierungen



Es verschiedene Alternativen zur Darstellung der Generalisierung:

- „Tabelle pro Entitiy“, bei der für jede Entity des Diagramms auch eine eigene Tabelle erstellt wird.
- „Pushdown“, bei der alle Generalisierungsattribute in der Spezialisierungstabelle platziert werden.
- „Pullup“, bei der alle Spezialisierungsattribute in eine einzige Generalisierungstabelle „heraufgezogen“ werden.

Die Vor- und Nachteile der einzelnen Strategien sollen am Beispiel deutlich werden.

1. Bilden Sie für jeden dieser Ansätze das obige E/R-Schema auf ein relationales Schema ab. Welche Daten werden in den Tabelle abgelegt ?

Tabelle pro Entity:

Konto: (Kontonr, Guthaben, Gebühr)

Festgeldkonto: (Kontonr, Fälligkeit, Zinssatz)

Girokonto: (Kontonr, Dispolimit, Telebanking)

Pushdown:

Festgeldkonto: (Kontonr, Guthaben, Gebühr, Fälligkeit, Zinssatz)

Girokonto: (Kontonr, Guthaben, Gebühr, Dispolimit, Telebanking)

Zusatzüberlegung: Gibt es Elemente in der Generalisierung, die nicht Elementen einer Spezialisierung entsprechen, z.B. Konten, die weder Girokonten noch Festgeldkonten sind? Wenn ja, dann muss auch eine Tabelle *Konto* angelegt werden, die diese (und nur diese) aufnimmt.

Konto: (Kontonr, Guthaben, Gebühr)

Pullup:

Konto: (Kontonr, Typ, Guthaben, Gebühr, Fälligkeit, Zinssatz, Dispolimit, Telebanking)

Alle für eine bestimmte Ausprägung nicht notwendigen Attribute werden leer gelassen, bzw. mit NULL belegt (Vorgriff auf SQL). Neben den Attributen aller Klassen muss noch ein Typidentifikator hinzugefügt werden, der es ermöglicht, den Entitytyp für einen Eintrag zu erkennen. Eine Überprüfung an Hand von leeren und belegten Feldern kann diesen nur dann ersetzen, wenn sichergestellt ist, dass die identifizierenden Felder immer belegt sind, z.B. *Dispolimit* bei *Girokonto*. Eine solche Überprüfung ist jedoch relativ schwierig.

2. Überlegen Sie (mit Hilfe relationaler Algebra), welche Schritte notwendig sind, um auf die vollständigen Daten aller Girokonten zuzugreifen.

Tabelle pro Entity:
Konto $\bowtie$ Girokonto

Pushdown:

Girokonto

Pullup:

$\pi_{\text{Kontonr, Guthaben, Gebühr, Dispolimit, Telebanking}}(\sigma_{\text{Typ='Girokonto'}}(\text{Konto}))$

3. Überlegen Sie, welche Schritte notwendig sind, um Kontonummern, Guthaben und Gebühren für alle Konten zu ermitteln.

Tabelle pro Entity:
Konto

Pushdown:

$\pi_{\text{Kontonr,Guthaben,Gebühr}}(\text{Girokonto}) \cup \pi_{\text{Kontonr,Guthaben,Gebühr}}(\text{Festgeldkonto})$

bzw. falls es Konten gibt, die weder Giro- noch Festgeldkonten sind:
$\pi_{\text{Kontonr,Guthaben,Gebühr}}(\text{Girokonto}) \cup \pi_{\text{Kontonr,Guthaben,Gebühr}}(\text{Festgeldkonto})$
$\cup \pi_{\text{Kontonr,Guthaben,Gebühr}}(\text{Konto})$

Pullup:

$\pi_{\text{Kontonr,Guthaben,Gebühr}}(\text{Konto})$

4. Welche Strategie ist empfehlenswert? (Gründe)

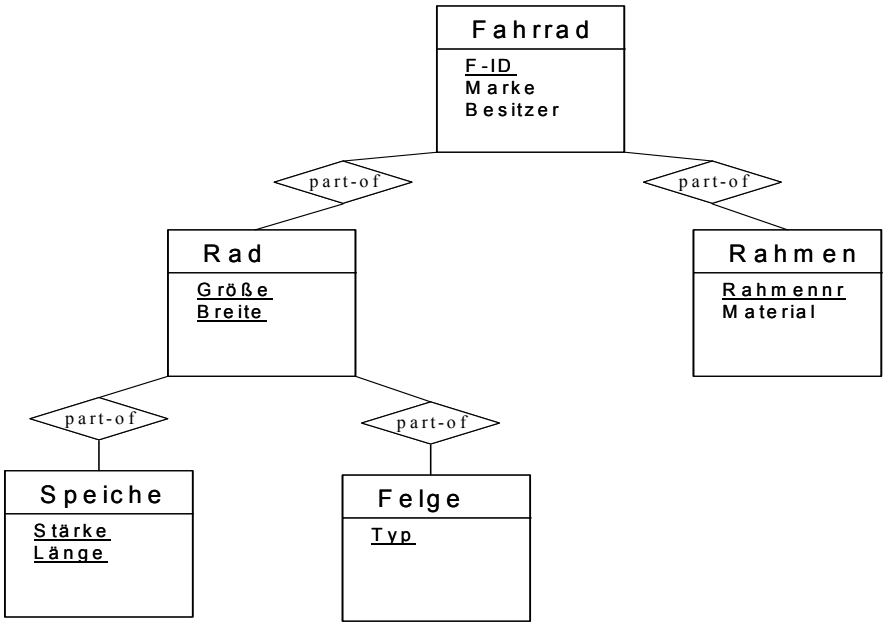
Die Entscheidung für eines der Verfahren fällt aufgrund des Anfragemusters.

- Falls sich viele Anfragen auf Spezialisierungen beziehen (z.B. Girokonten), so ist das Pushdown-Verfahren vorteilhaft, da hier lediglich diese Tabelle gelesen werden muss.
- Im „Tabelle-pro-Entity"-Verfahren muss ein Join zwischen Konto und Girokonto durchgeführt werden, um alle Attribute zu erhalten. Falls sich die Anfragen hauptsächlich auf die Generalisierung beziehen (z.B. Guthaben), ist das Verfahren „Tabelle-pro-Entity" besser, da hier nur eine einzige Tabelle betrachtet werden muss.
- Beim „Pushdown"-Verfahren müssen dagegen alle Spezialisierungstabellen betrachtet werden, und die Ergebnisse dann kombiniert werden.
- Das Pullup-Verfahren hat zwar den Vorteil, dass alle Anfragen mit Zugriffen auf eine einzige Tabelle erfüllt werden können. Diese enthält jedoch eine sehr große Menge an leeren Feldern, wodurch sehr viel Platz verschwendet wird.

Aufgabe 4: Aggregation

Unter Aggregation versteht man die Zusammenfassung von mehreren Einzelteilen zu einem komplexeren Ganzen. Zur Modellierung wird die sogenannte part-of-Beziehung (1:n bzw. 1:1) verwendet. Das Beispiel beschreibt also, aus welchen Teilen ein Fahrrad besteht. Da das Konzept der Aggregation weder voll mit dem E/R- noch mit dem relationalen Modell verträglich ist, muß die Umsetzung E/R nach relational direkt aus der jeweiligen Aufgabenstellung abgeleitet werden.

Erarbeiten Sie ein relationales Schema für folgendes Beispiel und überlegen Sie, ob es Optimierungsmöglichkeiten gibt.



Fahrrad: (F-ID, Marke, Besitzer) Rad: (Größe, Breite) Rahmen: (Rahmennr, Material) Speiche: (Stärke, Länge) Felge: (Typ) PartofRad: (Größe, Breite, F-ID) PartofRahmen: (Rahmennr, F-ID) PartofSpeiche: (Stärke, Länge, Größe, Breite) PartofFelge: (Typ, Größe, Breite)	Zusammengefasst: Fahrrad: (F-ID, Marke, Besitzer) Rad: (Größe, Breite, F-ID) Rahmen: (Rahmennr, Material, F-ID) Speiche: (Stärke, Länge, Größe, Breite) Felge: (Typ, Größe, Breite)

Eine begrenzte Verbesserung dieser Modellierung hinsichtlich der Tupelanzahl würde die Angabe der *Anzahl* in den Relationen *Rad* und *Speiche* mit sich bringen.

Vergibt man für alle Bauteile einen Identifikator, können alle part-of-Beziehungen in einer einzigen (rekursiven) Relation gespeichert werden:

Fahrrad: (F-ID, Marke, Besitzer)
Rad: (R-ID, Größe, Breite)
Rahmen: (Rahmennr, Material)
Speiche: (S-ID, Stärke, Länge)
Felge: (F-ID, Typ)
Partof: (Teil-ID, Einbauteil-ID)