FRANKFURT UNIVERSITY OF APPLIED SCIENCES

LE 5 – IBIS – Datenbanken

Überführung ER-Diagramme in relationale Strukturen und referenzielle Integrität

Disclaimer:
Basierend auf Folien der Veranstaltung Datenmanagement der THM Mittelhessen

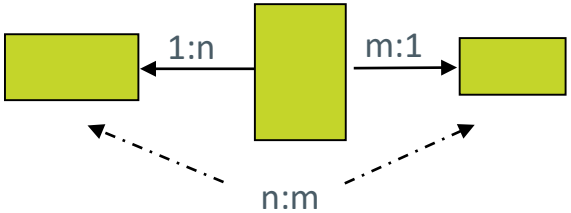
Prof. Dr. Markus Grüne, FB03, Wirtschaftsinformatik

Inhalt

- Vom konzeptuellen ER-Modell zum logischen Relationenmodell
- Referenzielle Integrität
- Typische Muster

ER-Modell zu Relation

- Die ER-Modelle werden in Relationen (Relationenmodell) übersetzt.
- Beziehungen müssen häufig in Tabellen abgelegt werden (n:m-Beziehungen bzw. solche mit weiteren Attributen).
- 1:n Beziehungen können ohne zusätzliche Tabellen abgebildet werden.

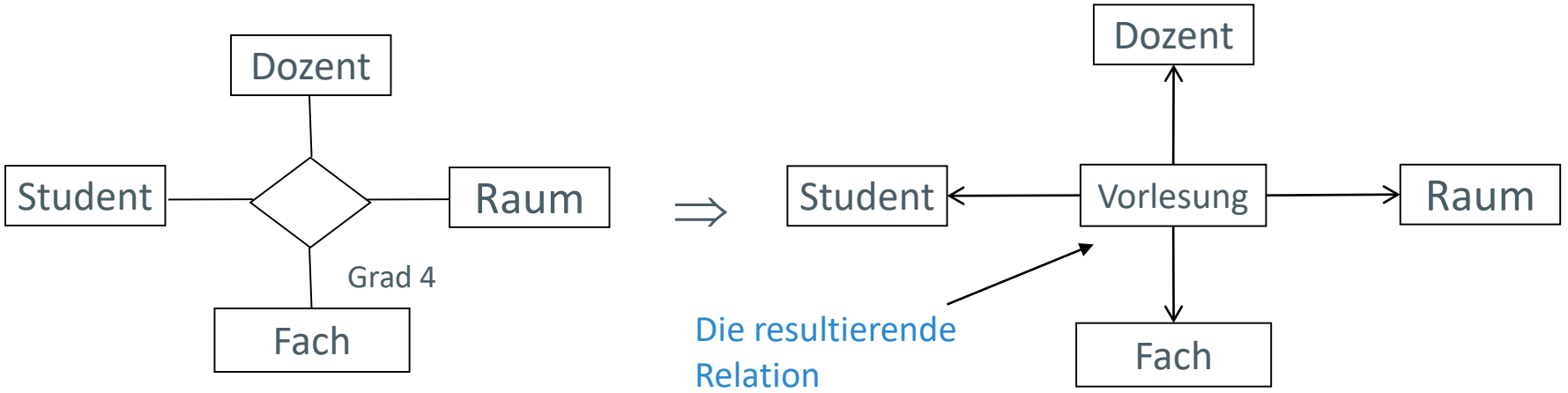


Warum bedarf es bei einer n:m-Beziehung einer weiteren Tabelle?

Beziehungen können Attribute besitzen – diese werden Attribute in der Tabelle, die die Beziehung abbildet.

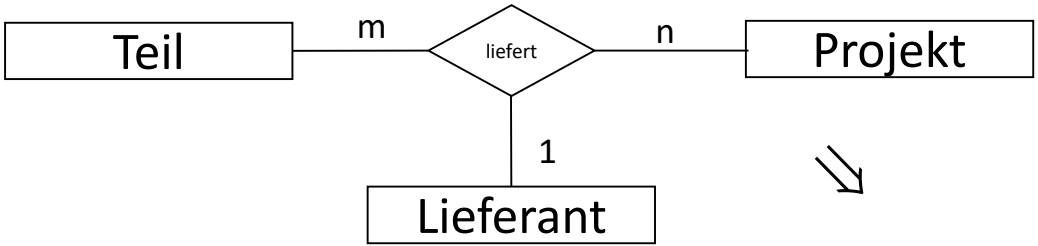
ER-Modell zu Relation

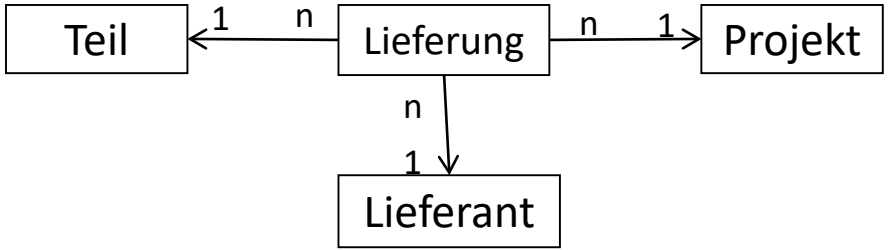
Die ER-Modelle müssen schließlich auf Tabellen in der Datenbank übertragen werden.



Beziehungen von höherem Grad werden durch eine Relation abgebildet!

Beispiel





Die Relation *Lieferung* hat Attribute, die auf *Teil*, *Projekt* und *Lieferant* verweisen.
Primärschlüssel ist die Kombination der Spalten, die *Teil* und *Projekt* referenzieren.
Damit kann es zu einem Teil pro Projekt nur einen Lieferanten geben.

Beispiel mit Daten

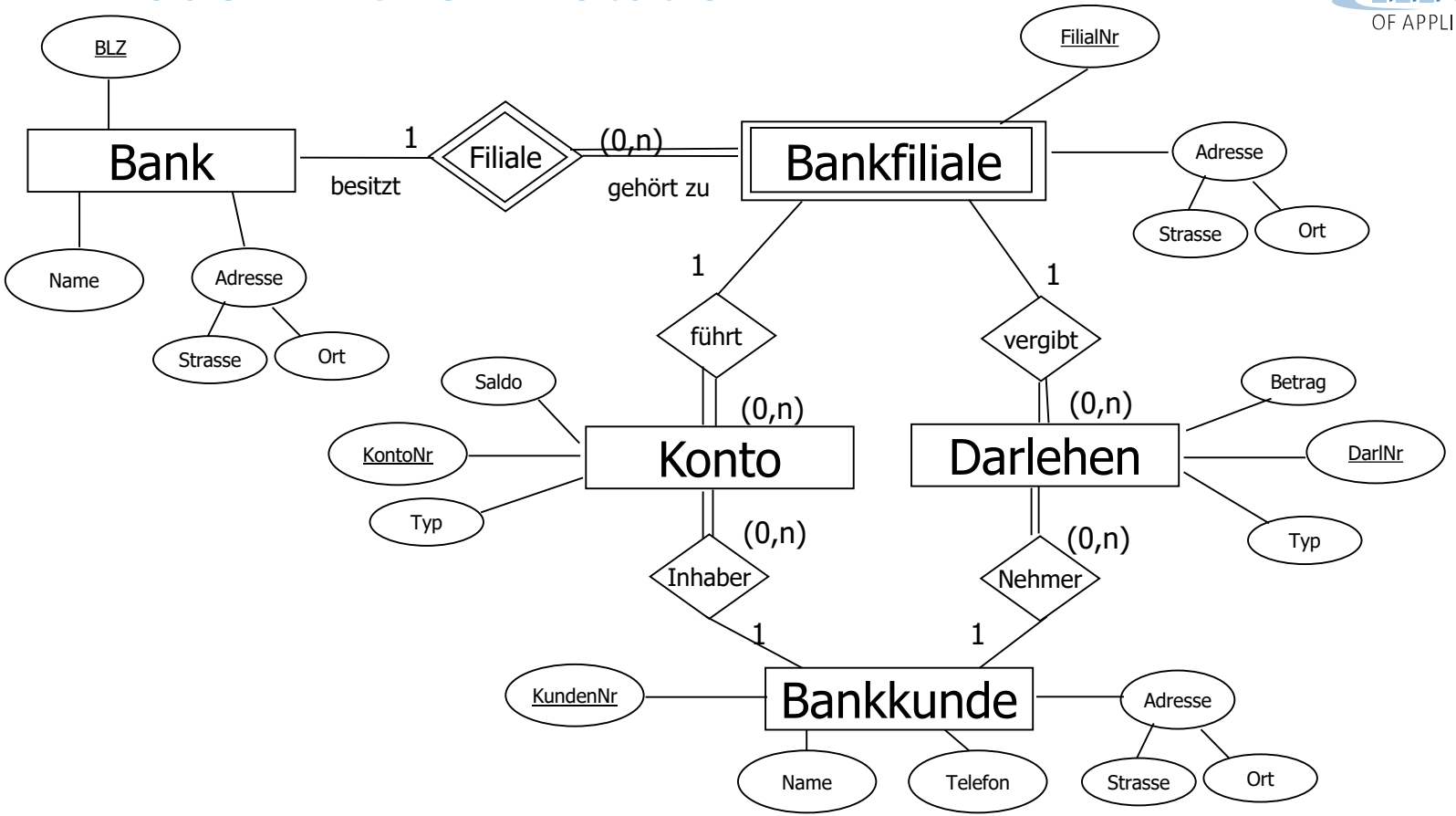
Teil Projekt Lieferant

(Schraube_115, Projekt_1, Müller_AG)
(Schraube_115, Projekt_2, Müller_AG)
(Dübel_115, Projekt_1, Meier_GmbH)

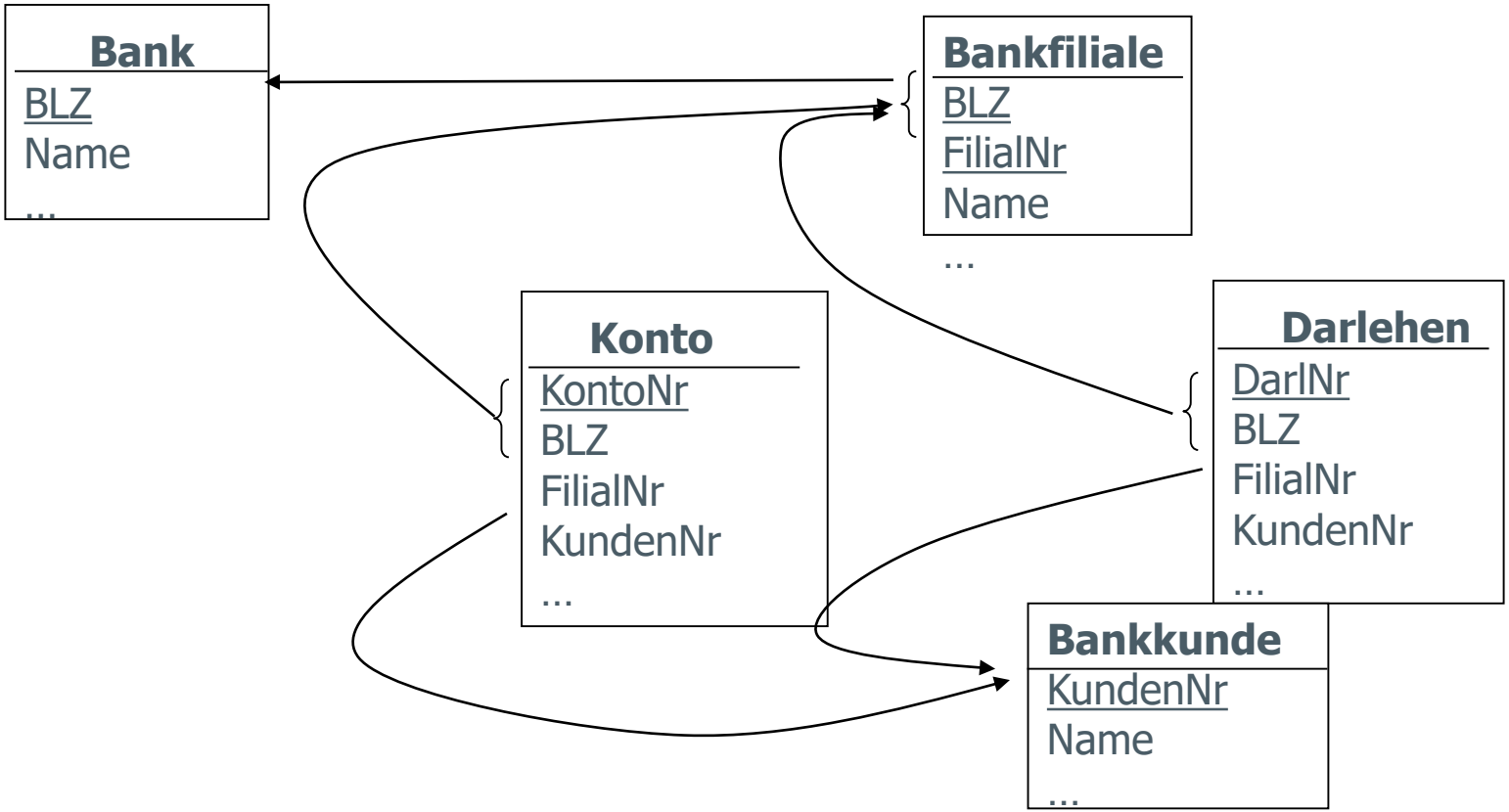
Aber nicht:

(Schraube_115, ~~Projekt_1~~, Meier_GmbH)

ER-Modell in Chen-Notation



Relationenmodell



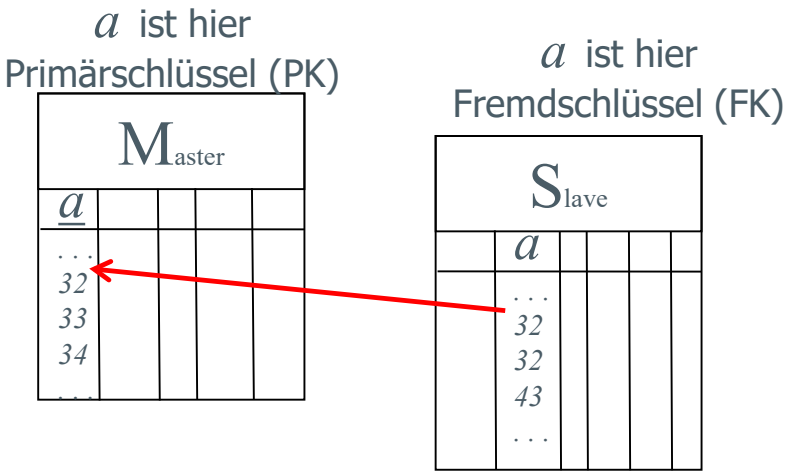
Referenzielle Integrität

Aufgrund der Forderung nach Normalisierung enthalten Tabellen Verweise auf Einträge anderer Tabellen.
Ein elementare Voraussetzung ist dabei, dass diese Einträge immer vorhanden sein müssen.

Datenbanken unterstützen die Referenzielle Integrität.

Bei Änderungen an der Datenbank, die Verweise ändern, wird geprüft, ob die Bezüge gemäß der Regeln intakt bleiben.

Referenzielle Integrität



Regeln beim Löschen in Master:

RESTRICT: Nicht möglich, erst muss Referenz entfernt werden.

CASCADE: Auch in Slave wird gelöscht.

SET NULL: Setze Attribut in Slave auf NULL.

Datensätze in S zeigen über den Fremdschlüssel *a* auf die Tabelle M.
Zu einem Eintrag in S mit Schlüsselwert a (Fremdschlüssel) muss in M ein eindeutiger Datensatz mit Schlüsselwert a (Primärschlüssel) existieren.
Umgekehrt gilt das NICHT!!!

Die Spalte *a* in S kann (in eher seltenen Fällen) den Wert NULL erhalten.

RI kann mit Hilfe von SQL-Statements (DDL) definiert werden!

Typische Muster

Typische Konstellationen wiederholen sich in verschiedenen Datenmodellen strukturell in immer wieder gleicher Weise.

Besitzt man eine Vorlage, so kann man diese leicht auf die neue Situation übertragen.

Im Folgenden werden einige dieser typischen Muster vorgestellt.

1:n identifizierend

- Eine Entität e_1 des Entitätstyps E_1 „besteht“ aus einer vorher unbekannten Anzahl Exemplaren des Entitätstyps E_2.
- Die umfassende Entität wirkt identifizierend.

Beispiel:
Ein *Auftrag* hat n *Auftragspositionen*
Eine *Rechnung* hat n *Rechnungspositionen*

Bemerkung:
E_1 stellt einen Teil des Schlüssels von E_2. Eine Entität aus E_2 kann ohne ein zugehöriges E_1 nicht existieren.
Änderung kann kaskadieren.

Identifying Relationship

An **identifying relationship** is when the existence of a row in a child table depends on a row in a parent table. This may be confusing because it's common practice these days to create a pseudokey for a child table, but *not* make the foreign key to the parent part of the child's primary key. Formally, the "right" way to do this is to make the foreign key part of the child's primary key. But the logical relationship is that the child cannot exist without the parent.

https://stackoverflow.com/questions/762937/whats-the-difference-between-identifying-and-non-identifying-relationships

Identifying Relationship

Example: A Person has one or more phone numbers. If they had just one phone number, we could simply store it in a column of Person. Since we want to support multiple phone numbers, we make a second table PhoneNumbers, whose primary key includes the person_id referencing the Person table.

We may think of the phone number(s) as belonging to a person, even though they are modeled as attributes of a separate table. This is a strong clue that this is an identifying relationship (even if we don't literally include person_id in the primary key of PhoneNumbers).

https://stackoverflow.com/questions/762937/whats-the-difference-between-identifying-and-non-identifying-relationships

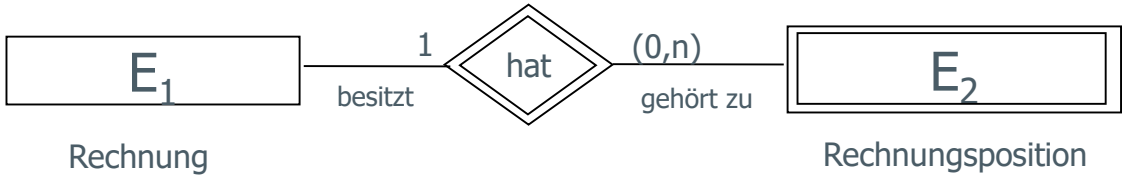
Non-identifying relationship

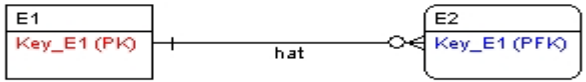
A **non-identifying relationship** is when the primary key attributes of the parent *must not* become primary key attributes of the child. A good example of this is a lookup table, such as a foreign key on Person.state referencing the primary key of States.state. Person is a child table with respect to States. But a row in Person is not identified by its state attribute. I.e. state is not part of the primary key of Person.

A non-identifying relationship can be **optional** or **mandatory**, which means the foreign key column allows NULL or disallows NULL, respectively.

https://stackoverflow.com/questions/762937/whats-the-difference-between-identifying-and-non-identifying-relationships

1:n identifizierend






```
Create table E1 (
   Key_E1 Varchar2(20) NOT NULL.
   ….
);
Create table E2 (
  Key_E1 Varchar2(20) NOT NULL ,
  Key_E2 Number NOT NULL,
  ….
);
Alter table E1 add primary key (Key_E1) ;
Alter table E2 add primary key (Key_E1, keyE2) ;
Alter table E2 add  foreign key (Key_E1) references E1 (Key_E1)  on delete cascade;
```

Achtung: MySQL Code!

1:n nicht identifizierend

- Eine Entität e_1 des Entitätstyps E_1 wird in einer vorher unbekannten Anzahl von Exemplaren des Entitätstyps E_2 „genutzt“.
- Die Entität e_1 wirkt nicht-identifizierend.

Beispiel:
Ein *Kunde* tätigt n *Aufträge*
Ein *Artikel* tritt in n *Rechnungspositionen* auf
Ein *Inhaber* hat n *Konten*

Bemerkung:
Die Schlüssel von E_1 und E_2 sind unabhängig. Die referentielle Integrität muss in der Regel ein Löschen in E_1 verhindern. Die zu referenzierende Entität in E_1 muss bei Anlage eines E_2 existieren.

Aufgabe

Erstellen Sie eine 1:n-Beziehung, die nicht identifizierend ist, in der MySQL-Workbench.

Erstellen Sie anschließend eine 1:n-Beziehung, die identifizierend ist.

Inwiefern unterscheiden sich diese hinsichtlich der DDL?

Lernziel / Fragen

Übergang vom ER-modell (konzeptioneller Entwurf) zum Relationenmodell (logischer Entwurf)

- Was sind Relationen?
- Wie wird ein ER-Modell in ein Relationenmodell übersetzt?
- Welche Regeln gibt es?
- Was ist ein „gutes“ Datenmodell?

Elemente, die bereits in Kapitel 1 vorgestellt wurden, wurden hier präzisiert.