



Datenbanken

Relationale Datenmodell

Prof. Dr. Ludger Martin

Gliederung

- ★ Relationen
- ★ Transformation eines ER-Diagramms in das Relationenmodell
- ★ Datenbank-Definition mit SQL

Relationale Datenmodell

- ★ Seit Mitte der 80er Jahre De-facto-Standard
- ★ „Sprache“ des Modells bestehend aus
 - ★ Relationen
 - ★ Integritätsbedingungen
 - ★ Schemata

Relationen

<i>buch</i>	<u>invnr</u>	autor	titel	verlag	jahr
	027-2408	Jones	Algorithms	PH	2003
	188-2887	Jameson	Web Design	JO	2006



<i>entleihen</i>	<u>invnr</u>	<u>lesernr</u>	datum
	027-2408	428456	2006-09-30



<i>leser</i>	<u>lesernr</u>	name	telefon
	428456	Andrews	07-8446524

- ★ Datenbank besteht aus: Tabellen, Attribute, Zeilen und Abhängigkeiten
- ★ Zeitveränderliche Inhalte

Relationen

- ★ Jede Tabellenzeile entspricht einem Tupel

$$(x_1, x_2, \dots, x_n)$$

Jedem Platz im Tupel ist ein Attribut fest zugeordnet

- ★ Eine **Relation** besteht aus Attributen und Tupeln
- ★ Jede Relation über eine Attributmengende ist als Tabelle darstellbar – in Kopfzeile stehen Attribute

<i>Buch</i>	<i><u>InvNr</u></i>	<i>Autor</i>	<i>Titel</i>	<i>VerlagName</i>	<i>VerlagOrt</i>	<i>Jahr</i>

Relationen

- ★ Entsprechung ER-Modell und Relationenmodell
 - ★ Ein Entity-Set, das nur einwertige Attribute umfasst, entspricht einer Relation
(*Verzicht auf Zeitmarke*)
 - ★ Attributmenge einer Relation heißt *Relationenformat*
- ★ Nicht immer sind in allen Attributen Werte enthalten → Ergänzung um einen *Nullwert*
- ★ Nullwerte sind nicht miteinander vergleichbar!
- ★ Eine Relation heißt **partiell**, wenn Nullwerte erlaubt, ansonsten **total**

Transformation eines ER-Diagramms in das Relationenmodell

★ Vorgehensweise:

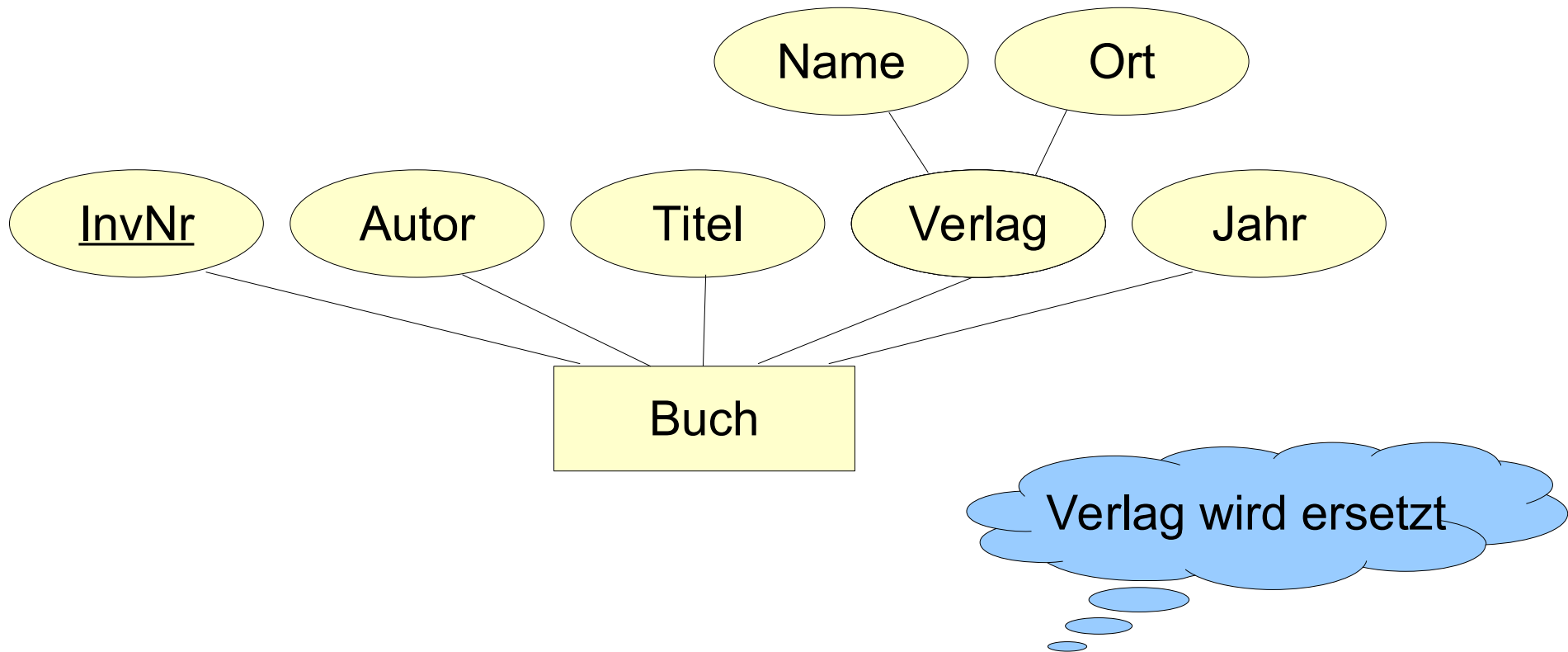
1. Jeder Entity-Typ wird in ein Relationenschema transformiert.
2. Jeder Relationship-Typ wird ebenfalls in ein Relationenschema transformiert, es sei denn, es handelt sich um eine zweistellige 1:1 oder 1:n Beziehung, in diesen Fällen reicht die Hinzunahme von Attributen zu bereits existierenden Relationenschemata.
3. IS-A-Beziehungen werden alleine über Inklusionsabhängigkeiten ausgedrückt.

Transformation eines ER-Diagramms in das Relationenmodell

- ★ Relationenmodell unterstützt keine mehrwertige oder zusammengesetzte Attribute
- ★ Für *zusammengesetzte Attribute* gibt es zwei Möglichkeiten
 - ★ *Flache Darstellung*: Ersetzung durch Komponenten
 - ★ *Zusammensetzung aufgeben*: ohne Komponenten

Transformation eines ER-Diagramms in das Relationenmodell

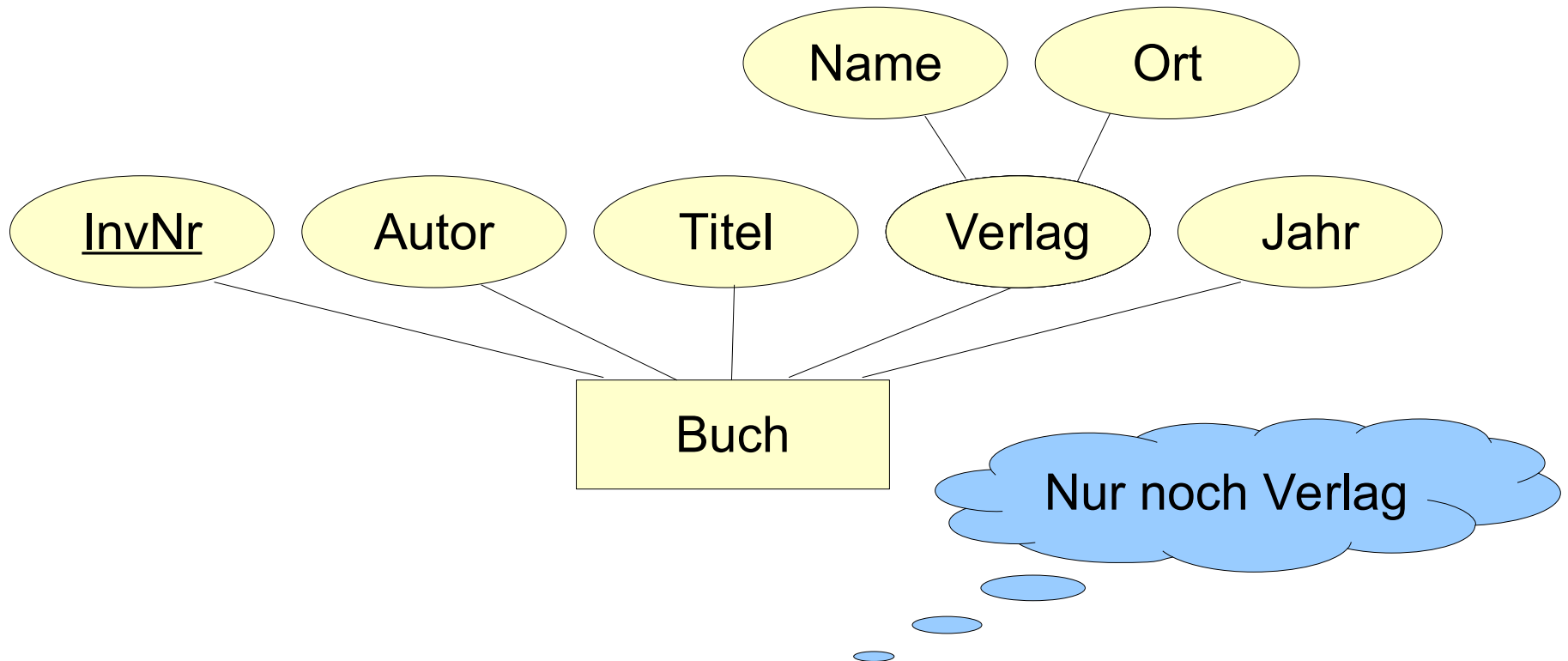
★ Beispiel: flache Darstellung



<i>Buch</i>	<u><i>InvNr</i></u>	<i>Autor</i>	<i>Titel</i>	<i>VerlagName</i>	<i>VerlagOrt</i>	<i>Jahr</i>
-------------	---------------------	--------------	--------------	-------------------	------------------	-------------

Transformation eines ER-Diagramms in das Relationenmodell

★ Beispiel: Zusammensetzung aufgeben



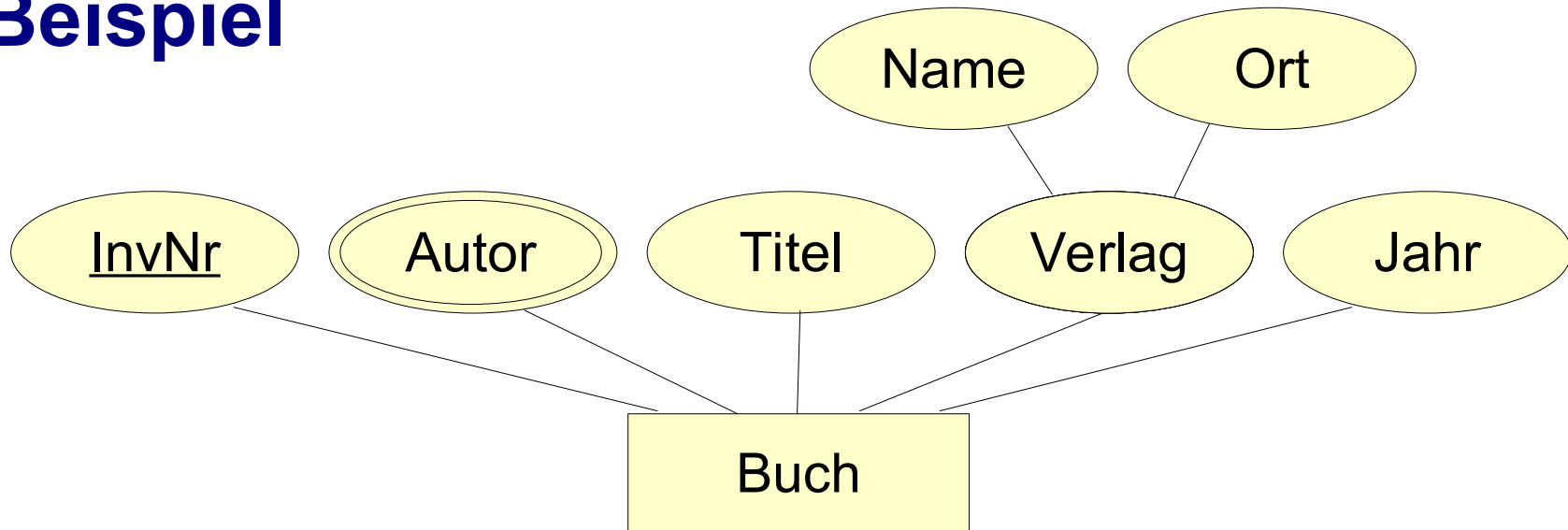
<i>Buch</i>	<u><i>InvNr</i></u>	<i>Autor</i>	<i>Titel</i>	<i>Verlag</i>	<i>Jahr</i>
-------------	---------------------	--------------	--------------	---------------	-------------

Transformation eines ER-Diagramms in das Relationenmodell

- ★ *Mehrwertige Attribute* durch Einführung neuer Relationenschemata mit trivialem Schlüssel
 - ★ Für jedes mehrwertige Attribut wird ein neues Relationenschema mit Attribut und Schlüssel zu dem ursprünglichen Relationenschema hinzugefügt.
 - ★ Logischer Zusammenhang durch Schlüssel dargestellt

Transformation eines ER-Diagramms in das Relationenmodell

★ Beispiel



<i>Buch</i>	<u><i>InvNr</i></u>	<i>Titel</i>	<i>VerlagName</i>	<i>VerlagOrt</i>	<i>Jahr</i>
-------------	---------------------	--------------	-------------------	------------------	-------------

<i>BuchAutoren</i>	<u><i>InvNr</i></u>	<u><i>Autor</i></u>
--------------------	---------------------	---------------------

Neues Relationenschema
mit Schlüsseln

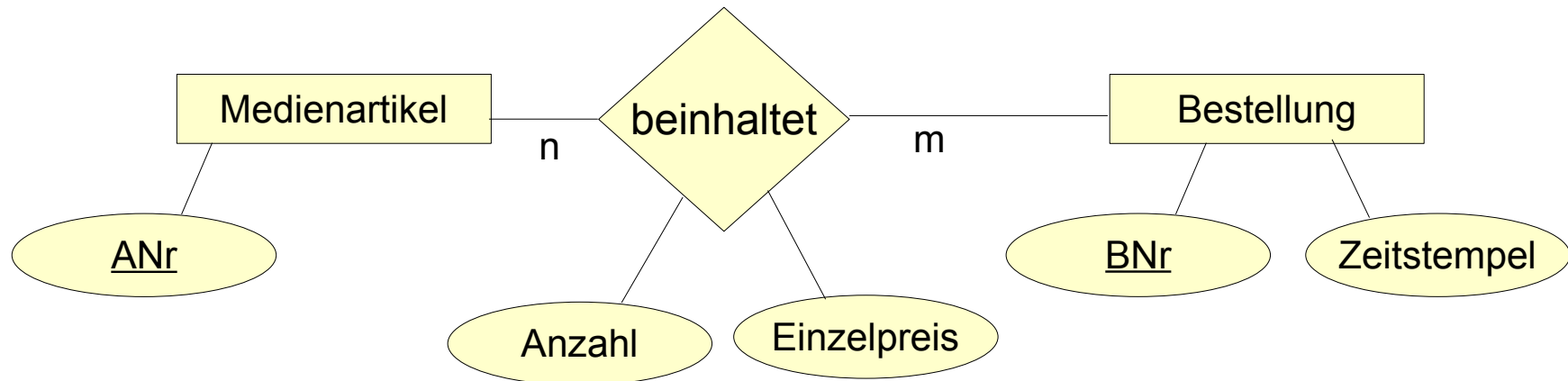
Transformation eines ER-Diagramms in das Relationenmodell

★ *Mehrstellige* sowie *m:n - Relationships*

- ★ Zur Modellierung nur Relationenschemata vorhanden
- ★ Neues Relationenschema wird aus den Schlüsseln der Entities und den Attributen der Relationships gebildet

Transformation eines ER-Diagramms in das Relationenmodell

★ Beispiel



<i>beinhaltet</i>	<u><i>ANr</i></u>	<u><i>BNr</i></u>	<i>Anzahl</i>	<i>Einzelpreis</i>
-------------------	-------------------	-------------------	---------------	--------------------

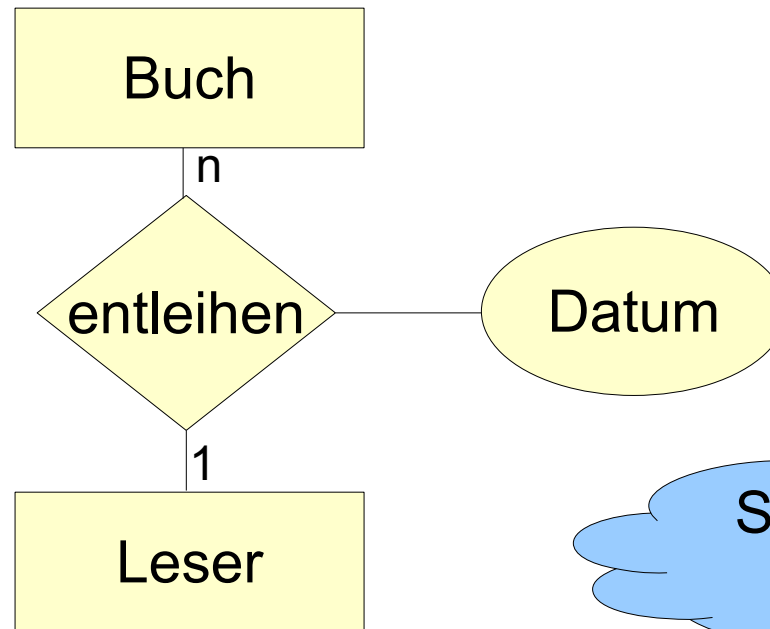
Neues Relationenschema
aus Schlüsseln und Attributen

Transformation eines ER-Diagramms in das Relationenmodell

- ★ Zweistellige 1:1 oder 1:n Relationenschemata
 - ★ Durch Erstellung eines neuen Relationenschemas wie zuvor beschrieben → es geht aber auch ohne neues Relationenschema
 - ★ In eines der beiden Relationenschemata kann Schlüssel auf anderes als Attribut eingefügt werden → benötigt aber *Nullwerte*
 - ★ Aus Performancegründen sollten weniger Relationen angelegt werden

Transformation eines ER-Diagramms in das Relationenmodell

★ Beispiel



Schlüssel und Attribut
hinzugefügt

<i>Buch</i>	<u><i>InvNr</i></u>	<i>Titel</i>	<i>VerlagName</i>	<i>VerlagOrt</i>	<i>Jahr</i>	<i>LeserNr</i>	<i>Datum</i>
-------------	---------------------	--------------	-------------------	------------------	-------------	----------------	--------------

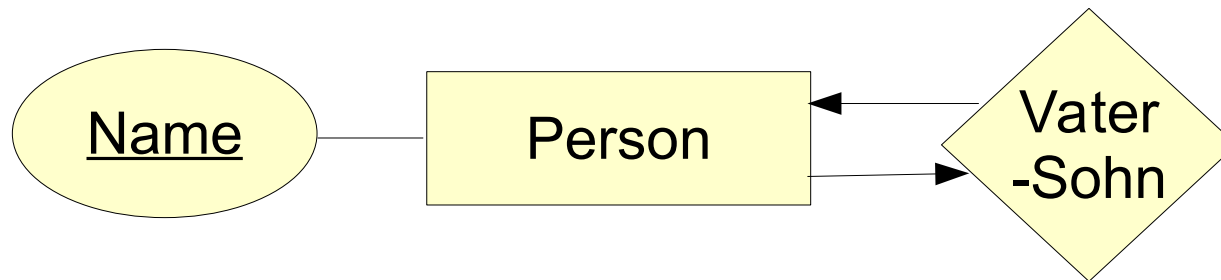
Mit folgender Bedingung:

$$Buch[LeserNr] \subseteq Leser[LeserNr]$$

Transformation eines ER-Diagramms in das Relationenmodell

★ Rekursive Beziehungen

- ★ Schlüssel von Entity wird dupliziert.
- ★ Beziehung wird durch Umbenennung verdeutlicht



<i>VaterSohn</i>	<u><i>VaterName</i></u>	<u><i>SohnName</i></u>
------------------	-------------------------	------------------------

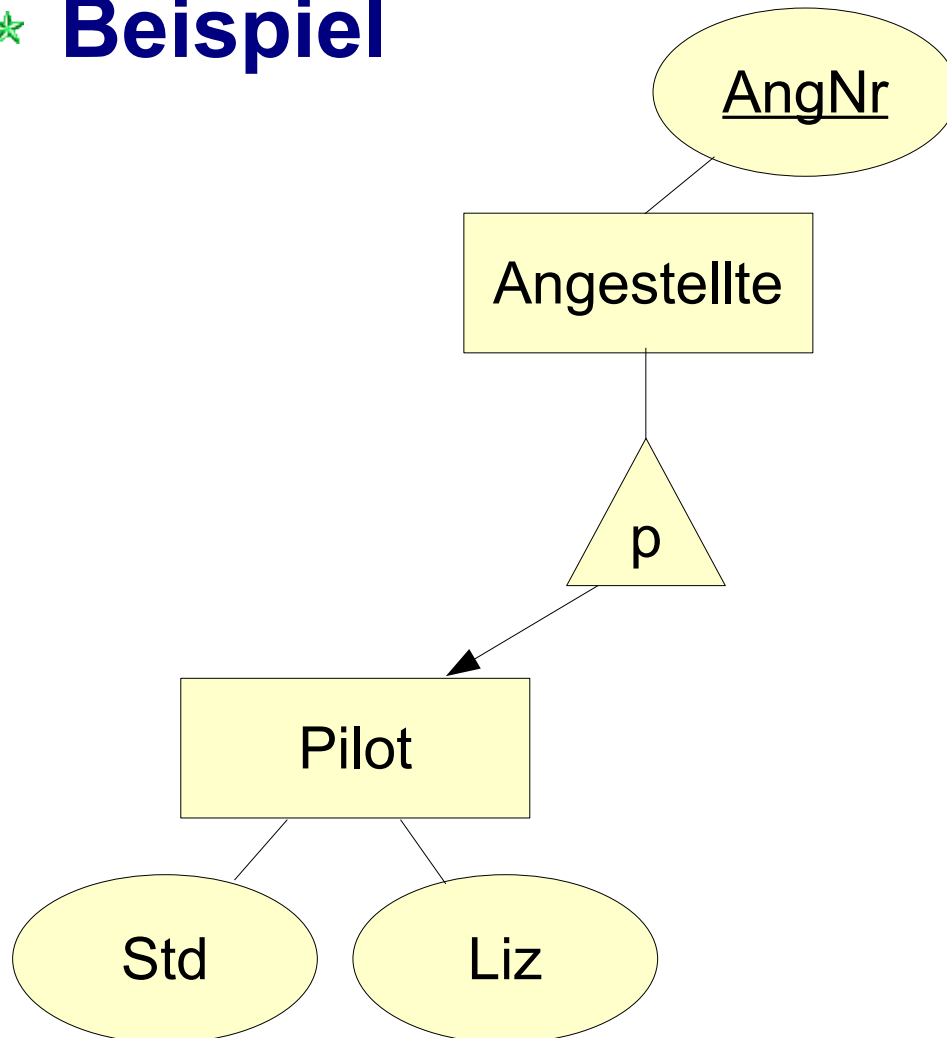
Transformation eines ER-Diagramms in das Relationenmodell

★ IS-A Beziehung

- ★ Nur Schemata für beteiligte Entity-Typen, nicht für Beziehung
- ★ Schlüssel der Verallgemeinerung dient als Schlüssel, ergänzt durch weitere Attribute

Transformation eines ER-Diagramms in das Relationenmodell

★ Beispiel



<i>Pilot</i>	<i><u>AngNr</u></i>	<i>Std</i>	<i>Liz</i>

Transformation eines ER-Diagramms in das Relationenmodell

★ IS-A Beziehung (Fortsetzung)

- ★ Folgende Inklusionsabhängigkeit wird verlangt

$$Pilot[AngNr] \subseteq Angestellte[AngNr]$$

- ★ Bei mehreren Spezialisierungen wird für jede eine Inklusionsabhängigkeit benötigt

Techniker IS-A Angestellte IS-A Person

$$\begin{aligned} Techniker[PersNr] &\subseteq Angestellte[PersNr] \\ &\subseteq Personen[PersNr] \end{aligned}$$

Transformation eines ER-Diagramms in das Relationenmodell

★ IS-A Beziehung (Fortsetzung)

- ★ Für *nicht disjunkt* können Bedingungen angegeben werden

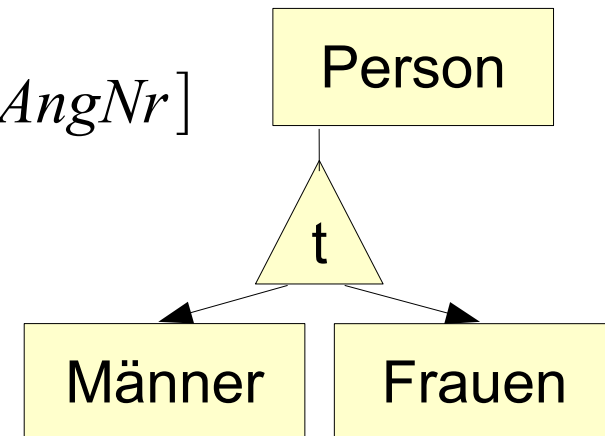
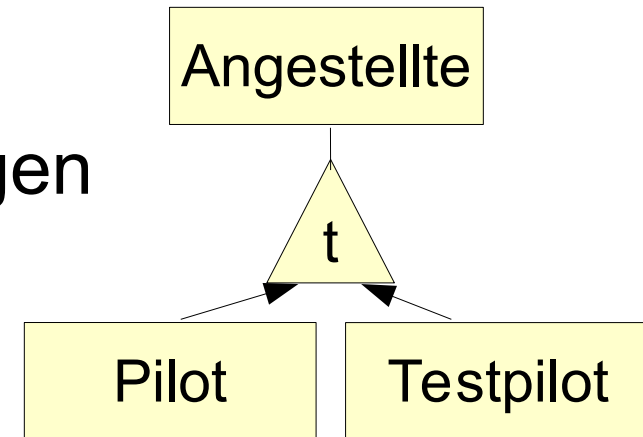
$$\begin{aligned} Pilot[AngNr] &\subseteq Angestellte[AngNr] \\ Testpilot[AngNr] &\subseteq Angestellte[AngNr] \end{aligned}$$

für *total*:

$$Angestellte[AngNr] = Pilot[AngNr] \cup Testpilot[AngNr]$$

für *disjunkt*:

$$Männer[PersNr] \cap Frauen[PersNr] = \emptyset$$



- ★ Bei *partiell* und *nicht disjunkt* gibt es keine weiteren Bedingungen

Datenbank-Definition mit SQL

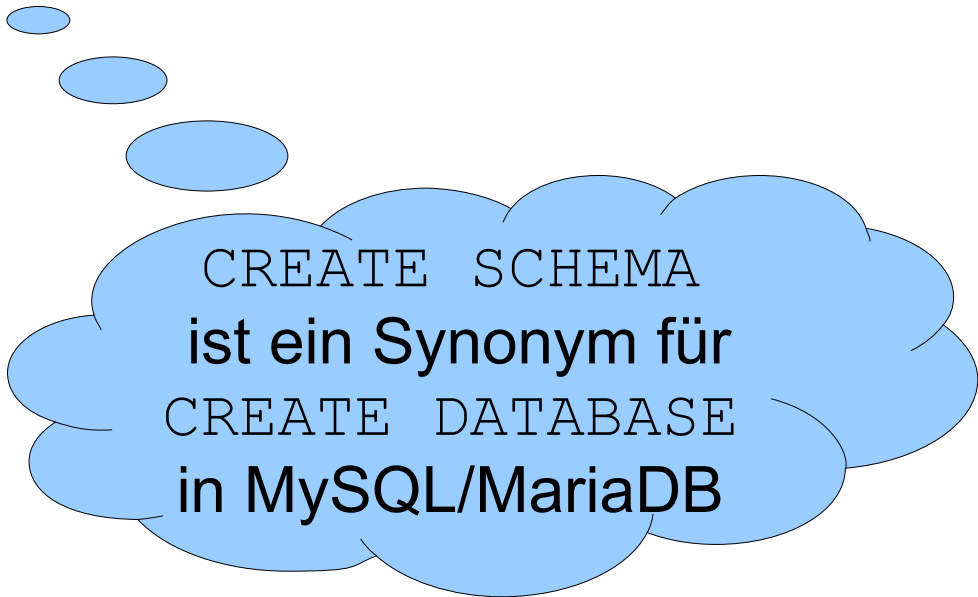
- ★ **Konvention:** Tabellen- und Spaltennamen klein, SQL-Schlüsselworte groß schreiben
- ★ Bezeichnungen
 - ★ *table* – Tabelle
 - ★ *row* – Zeile
 - ★ *column* – Spalte
- ★ SQL-Kommandos zur Daten-Definition
 - ★ CREATE
 - ★ ALTER
 - ★ DELETE

Datenbank-Definition mit SQL

Datenbankschema

- ★ Sammlung von Tabellen
- ★ Eine Schemadefinition hat einen Namen
- ★ Syntax:

```
CREATE SCHEMA db_name
```



CREATE SCHEMA
ist ein Synonym für
CREATE DATABASE
in MySQL/MariaDB

Datenbank-Definition mit SQL

Tabellen

★ Tabelle erstellen:

```
CREATE TABLE tbl_name (  
  col_name type eindeutiger Name und Typ  
  [NOT NULL | NULL] ob Spalte leer oder eindeutig  
  [DEFAULT default_value] Standardwert, falls kein Wert angegeben  
  [AUTO_INCREMENT] , wenn Primärschlüssel, automatisch weiter zählen  
  ... )
```

MySQL/MariaDB spezifisch

[|] und ... gehören nicht zur Syntax!

Datenbank-Definition mit SQL

Tabellen

[] gehören nicht zur Syntax!

★ Numerische Typen (MySQL/MariaDB)

Datentyp	Speicherplatz
TINYINT [(M)]	8 Bit
SMALLINT [(M)]	16 Bit
MEDIUMINT [(M)]	24 Bit
INT [(M)]	32 Bit
BIGINT [(M)]	64 Bit
FLOAT [(M, D)]	32 Bit
DOUBLE [(M, D)]	64 Bit
DECIMAL [(M [, D])]	variabel

★ *Optional* Gesamtlänge in Ziffern M und Ziffern nach Dezimalpunkt D: DECIMAL (5, 3)

★ Andernfalls wird Maximallänge erlaubt

Datenbank-Definition mit SQL

Tabellen

[] gehören nicht zur Syntax!

★ Zeichenreihen-Typen (MySQL/MariaDB)

Datentyp	Maximale Größe
CHAR [(M)] *	255 B
VARCHAR (M) *	255 B
TEXT [(M)] * / BLOB [(M)]	64 kB
MEDIUMTEXT * / MEDIUMBLOB	16 MB
LONGTEXT * / LONGBLOB	4 GB

★ CHAR ist fester Länge, übrige Typen sind variabler Länge

★ TEXT speichert beliebigen Text

★ BLOB speichert binäre Daten

* autom. Encoding-Konvertierung möglich

Datenbank-Definition mit SQL

Tabellen

★ Sonstige Typen (MySQL/MariaDB)

Datentyp	Erklärung
ENUM ('A' , 'B' , ...)	Liste mit maximal 65536 Posten, speichert genau ein Posten
SET ('A' , 'B' , ...)	Liste mit maximal 64 Posten, Speichert alle Posten
DATE	Datum YYYY-MM-DD
TIME	Zeit HH:MM:SS
DATETIME	YYYY-MM-DD HH:MM:SS
TIMESTAMP	YYYY-MM-DD HH:MM:SS in UTC

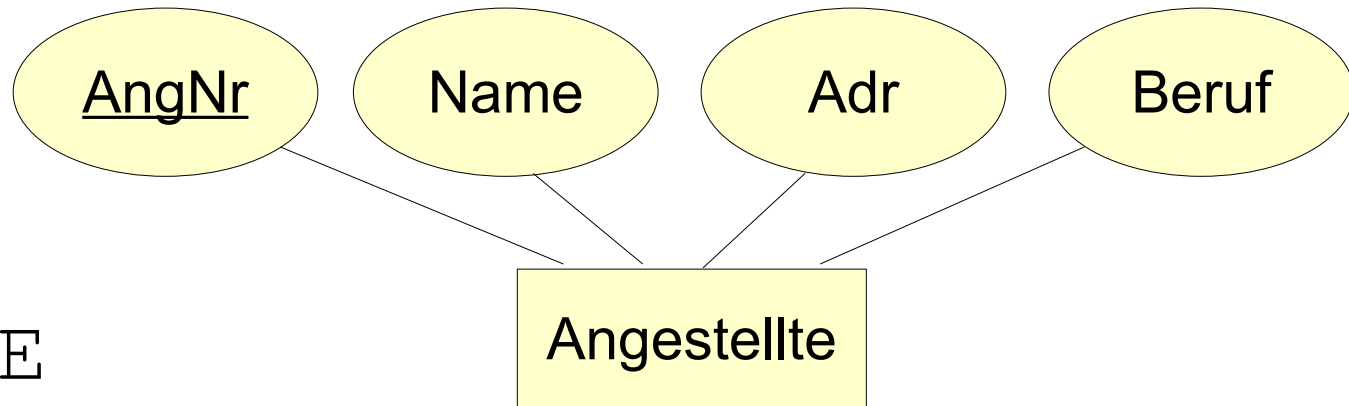
★ Nur eine Auswahl von Datentypen

★ In anderen DBMS kann es andere Typen geben

Datenbank-Definition mit SQL

Tabellen

★ Beispiel



```
CREATE TABLE  
angestellte
```

```
(angnr MEDIUMINT(5) NOT NULL,  
name VARCHAR(50) NOT NULL,  
adr VARCHAR(50),  
beruf VARCHAR(20)  
) ENGINE=InnoDB;
```

Möchten Sie reservierte
Worte als Namen nutzen,
schreiben Sie diese in `
Beispiel: `name`

ENGINE s. nächste Seite!

Datenbank-Definition mit SQL

Storage-Engines

★ Storage-Engines (MySQL/MariaDB)

- ★ Sind für Datenspeicherung zuständig
- ★ Können auch von Dritten entwickelt werden
- ★ Abfragen von Engines
SHOW ENGINES

ENGINE	SUPPORT	COMMENT	TRANSACTIONS	XA *	SAVEPOINTS
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disa...	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign ...	YES	YES	YES
ARCHIVE	YES	Archive storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary ...	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	<null>	<null>	<null>

* Distributed Transaction Processing: The XA Specification

Datenbank-Definition mit SQL

Storage-Engines

★ InnoDB

- ★ Default Storage Engine
- ★ Voll-transaktional
- ★ ACID-konform
 - ★ Referentielle Integrität (Fremdschlüssel)
 - ★ Multiversion Concurrency Control
- ★ Maximale Größe von Tabellen: 64TB
- ★ Maximale Zeilengröße (ohne `varchar`): ~8.000 Bytes
- ★ Index auf BLOB und TEXT: nur erste 767 Byte

Datenbank-Definition mit SQL

Storage-Engines

★ MyISAM

- ★ Bis Version 5.5 Default Storage Engine
- ★ Keine Transaktionen und ACID
- ★ Schnelle Storage Engine
- ★ Maximale Größe von Tabellen: 256TB
- ★ Maximale Zeilenzahl: 2^{64}
- ★ Index auf BLOB und TEXT: vollständig

Datenbank-Definition mit SQL

Storage-Engines

★ MEMORY

- ★ Daten nur im Hauptspeicher
- ★ Daten gehen bei einem Neustart verloren
- ★ Schema wird auf Festplatte gespeichert und wieder hergestellt.
- ★ Hohe Geschwindigkeit bei Zugriff auf Daten
- ★ Maximale Größe von Tabellen: je nach Hauptspeicher

Datenbank-Definition mit SQL

Storage-Engines

★ ARCHIVE

- ★ Wie MyISAM, nur können keine Daten gelöscht werden

★ CSV

- ★ Daten werden als *Comma-Separated-Values* gespeichert

★ BLACKHOLE

- ★ Es werden keine Zeilen gespeichert
- ★ Für Spezialszenario der Replikation (Hochverfügbarkeit)

Datenbank-Definition mit SQL

Integritätsbedingungen

- ☆ Wertebereichs-Bedingung (z.B. `INT`) ✓
- ☆ Attribut-Bedingung (z.B. `NOT NULL`) ✓
- ☆ Primärschlüssel
- ☆ Fremdschlüssel

Datenbank-Definition mit SQL

Integritätsbedingungen

- ★ Mehrere Schlüssel erlaubt
- ★ Nur einer als Primärschlüssel

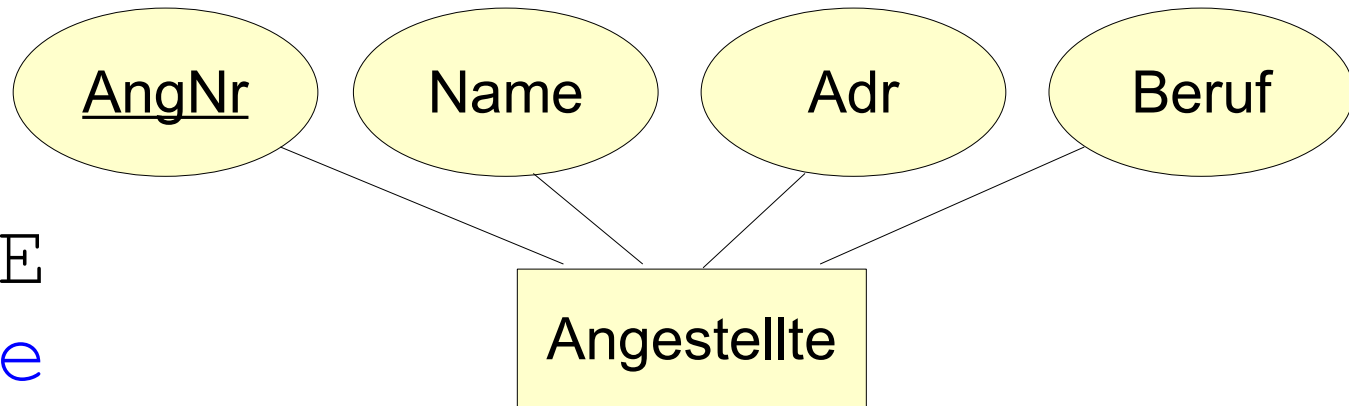
★ CREATE TABLE `tbl_name` (...
 [CONSTRAINT `cstr_name`]
 PRIMARY KEY
 (`list_of_column_names`),
 [CONSTRAINT `cstr_name`]
 UNIQUE [KEY] [`index_name`]
 (`list_of_column_names`),
 ...)

{ } | [] gehören
nicht zur Syntax!

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Beispiel



```
CREATE TABLE
  angestellte
  (angnr MEDIUMINT(5) NOT NULL,
   name VARCHAR(50) NOT NULL,
   adr VARCHAR(50),
   beruf VARCHAR(20),
   PRIMARY KEY (angnr)
  ) ENGINE=InnoDB;
```

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Fremdschlüssel

★ Ist ein Attribut (Attributmenge) in einer Relation gleichzeitig Schlüssel einer anderen Relation, ist das Attribut (Attributmenge) ein Fremdschlüssel. Es ist ein Schlüssel einer fremden Relation.

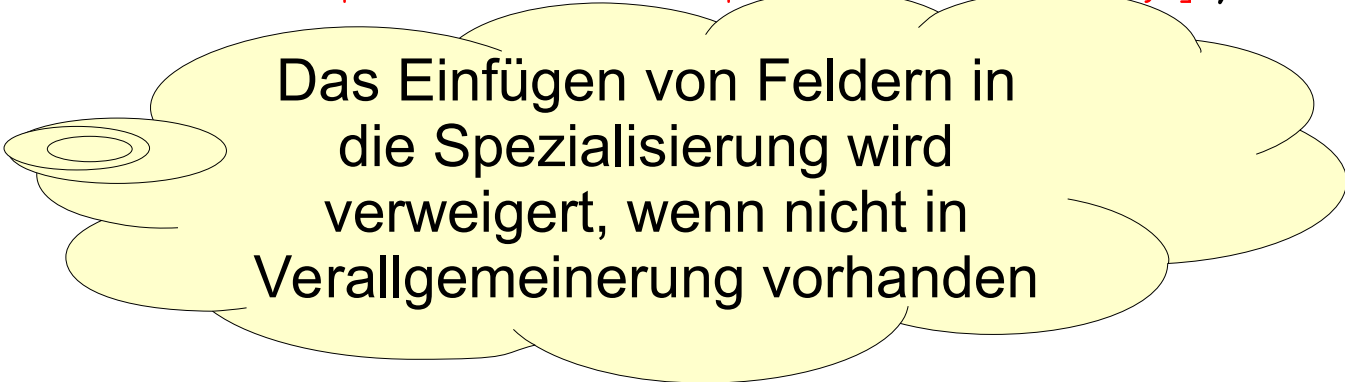
```
★ CREATE TABLE tbl_name ( ...  
    [CONSTRAINT [cstr_name]]  
    FOREIGN KEY (list_of_column_names)  
    REFERENCES table_name (list_of_column_names)  
    [ON DELETE  
        { NO ACTION | CASCADE | SET NULL }]  
    [ON UPDATE  
        { NO ACTION | CASCADE | SET NULL }]  
    ... )
```

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Fremdschlüssel

```
★ CREATE TABLE tbl_name ( ...  
    [CONSTRAINT [cstr_name]]  
    FOREIGN KEY (list_of_column_names)  
    REFERENCES table_name (list_of_column_names)  
    [ON DELETE  
        { NO ACTION | CASCADE | SET NULL }]  
    [ON UPDATE  
        { NO ACTION | CASCADE | SET NULL }],  
    ... )
```



Das Einfügen von Feldern in
die Spezialisierung wird
verweigert, wenn nicht in
Verallgemeinerung vorhanden

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Fremdschlüssel

```
★ CREATE TABLE tbl_name ( ...  
    [CONSTRAINT [cstr_name]]  
    FOREIGN KEY (list_of_column_names)  
    REFERENCES table_name (list_of_column_names)  
    [ON DELETE  
        { NO ACTION | CASCADE | SET NULL }]  
    [ON UPDATE  
        { NO ACTION | CASCADE | SET NULL }],  
    ...
```

Die Aktion, die ausgeführt wird, wenn ein Feld in der Verallgemeinerung gelöscht oder aktualisiert wird, kann mit ON DELETE bzw. ON UPDATE bestimmt werden

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Fremdschlüssel

```
★ CREATE TABLE tbl_name ( ...  
    [CONSTRAINT [cstr_name]]  
    FOREIGN KEY (list_of_column_names)  
    REFERENCES table_name (list_of_column_names)  
    [ON DELETE  
        { NO ACTION | CASCADE | SET NULL }]  
    [ON UPDATE  
        { NO ACTION | CASCADE | SET NULL }],  
    ... )
```


NO ACTION bestimmt, dass das Löschen
oder Aktualisieren verweigert wird

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Fremdschlüssel

```
★ CREATE TABLE tbl_name ( ...  
    [CONSTRAINT [cstr_name]]  
    FOREIGN KEY (list_of_column_names)  
    REFERENCES table_name (list_of_column_names)  
    [ON DELETE  
        { NO ACTION | CASCADE | SET NULL }]  
    [ON UPDATE  
        { NO ACTION | CASCADE | SET NULL }],  
    ... )
```



CASCADE bestimmt, dass die
Spezialisierung ebenfalls gelöscht bzw.
im gleichen Feld aktualisiert wird

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Fremdschlüssel

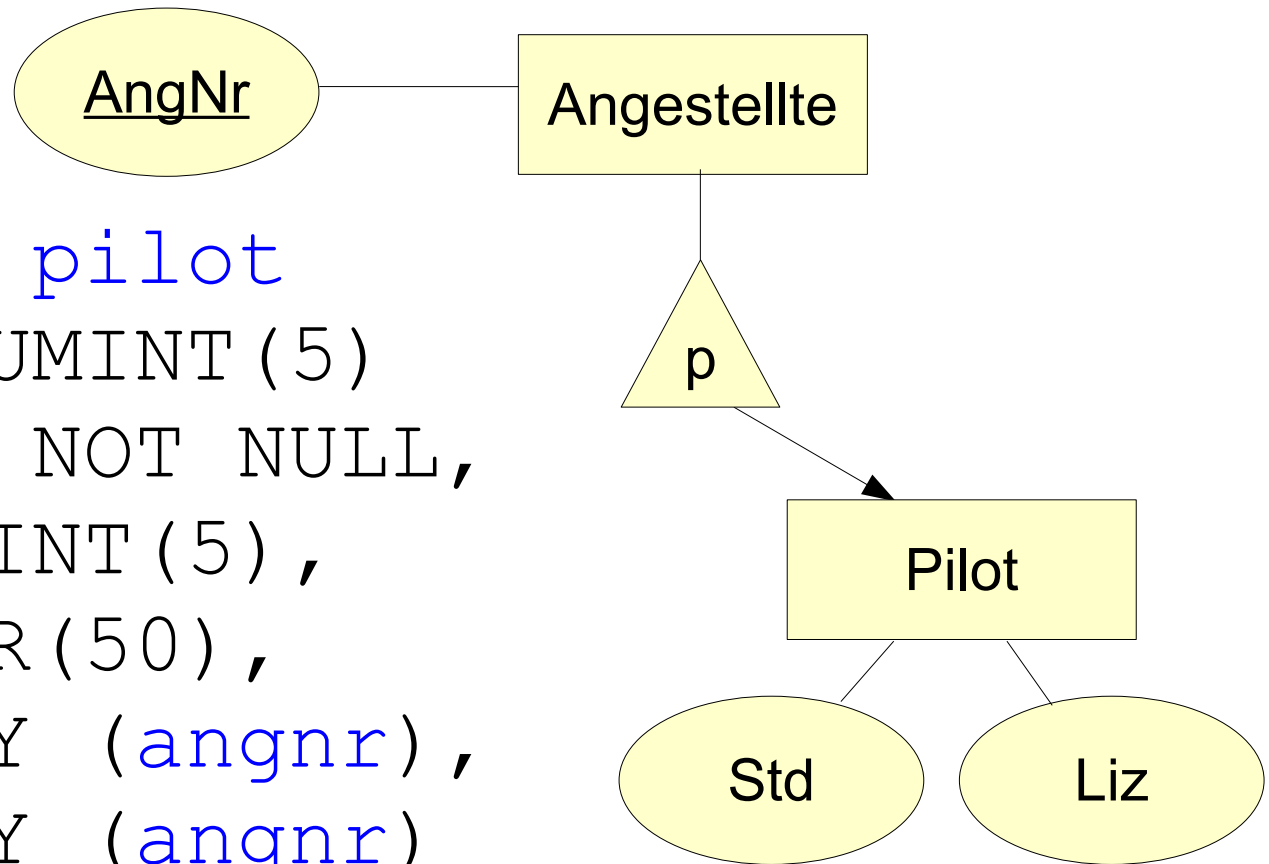
```
★ CREATE TABLE tbl_name ( ...  
    [CONSTRAINT [cstr_name]]  
    FOREIGN KEY (list_of_column_names)  
    REFERENCES table_name (list_of_column_names)  
    [ON DELETE  
        { NO ACTION | CASCADE | SET NULL }]  
    [ON UPDATE  
        { NO ACTION | CASCADE | SET NULL }],  
    ... )
```

SET NULL bestimmt, dass das Feld in der Spezialisierung auf NULL gesetzt wird (Feld darf nicht auf NOT NULL gesetzt sein)

Datenbank-Definition mit SQL

Integritätsbedingungen

★ Beispiel



```
CREATE TABLE pilot
(angnr MEDIUMINT(5)
      NOT NULL,
std MEDIUMINT(5),
liz VARCHAR(50),
PRIMARY KEY (angnr),
FOREIGN KEY (angnr)
REFERENCES angestellte(angnr)
) ENGINE=InnoDB;
```

Datenbank-Definition mit SQL

Zugriffsrechte

{ } | [] gehören
nicht zur Syntax!

★ Zugriffsrechte für Tabellen

```
★ GRANT { SELECT | INSERT | DELETE |  
        UPDATE | ALL }  
ON table_name  
TO user_name [, ...]  
[ WITH GRANT OPTION ]
```

- ★ Mit GRANT OPTION können eigene Privilegien an Dritte weitergegeben werden
- ★ Erzeuger einer Tabelle besitzt automatisch alle Privilegien mit Weitergaberecht

Datenbank-Definition mit SQL

Schemamodifikationen

- ★ Nachträgliche Änderungen zur Laufzeit
- ★ Bestehende Definition modifizieren oder löschen
- ★ **ALTER-Befehl** mit Kombination von `ADD`, `CHANGE` oder `DROP`

Datenbank-Definition mit SQL

Schemamodifikationen

```
★ ALTER TABLE table_name
{ ADD [COLUMN] col_name definition
| CHANGE [COLUMN] old_col_name
    new_col_name definition
| DROP [COLUMN] col_name
| ADD [CONSTRAINT [cstr_name]]
    { PRIMARY KEY (list_of_column_names)
    | UNIQUE (list_of_column_names)
    | FOREIGN KEY (list_of_column_names)
      REFERENCES ...
    }
| DROP PRIMARY KEY
| DROP FOREIGN KEY cstr_name
} , ...
```

Datenbank-Definition mit SQL

Schemamodifikationen

★ Beispiel:

```
ALTER TABLE angestellte  
  CHANGE angnr  
    angnr MEDIUMINT(5)  
    NOT NULL AUTO_INCREMENT
```

Datenbank-Definition mit SQL

Schemamodifikationen

★ Eine Tabelle löschen

```
DROP TABLE tbl_name [, tbl_name]
```

★ Beispiel:

```
DROP TABLE angestellte
```


Literatur

- ★ Pröll, S., Zangerle, E. Und Gassler, W.: MySQL – Das umfassende Handbuch, 2. Auflage, Galileo Press, 2013
- ★ Vossen, Gottfried: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- ★ Schwinn, Hans: Relationale Datenbanksysteme, Hanser, 1992
- ★ Sun Microsystems: MySQL 5.1 Reference Manual, <http://dev.mysql.com/doc/refman/5.1/en/>