we focus on students

Datenbanken 1

Was ist SQL?

Fachhochschule Dortmund
University of Applied Sciences

Structured Query Language (SQL)

WAS?	
Tabellen	***Tabelleninhalte***
Erstellen	Wiederfinden
Löschen	
Ändern	
Tabelleninhalte	***Daten schützen***
Einfügen	Datenkonsistenz
Ändern	(Transaktionen)
Löschen	Zugriffsrechte

SQL-Standardisierung

SQL/1 1986, SQL/2 1992, SQL/3 1999 … SQL:2011

SQL/Foundation	
***D**ata **D**efinition **L**anguage:* Create Drop Alter	***D**ata **R**etrieval **L**anguage:* Select
***D**ata **M**anipulation **L**anguage:* Insert Update Delete	***D**ata **C**ontrol **L**anguage:* Commit Rollback Grant/Revoke

Herstellerabhängige SQL-Dialekte

SQL-Foundation beschreibt (u.a.): Was gibt es für SQL-Befehle und deren Syntax

Herstellerabhängige SQL-Erweiterungen

Durch die Standardisierung wird lediglich der Funktionsumfang und –struktur des DBMS beschrieben, nicht jedoch die Umsetzung.
Folge:
- Es gibt herstellerabhängige Syntaxunterschiede
- Der Umfang der Umsetzung des Standards ist DBMS-Spezifisch

Beispiel

Beispiel aus dem Standard SQL:2003 (foundation):

‹alter table statement› ::= ALTER TABLE ‹table name› ‹alter table action›

‹alter table action› ::=

‹add column definition›

| ‹alter column definition›

| ‹drop column definition›

| ‹add table constraint definition›

| ‹drop table constraint definition›

```
ALTER TABLE Kunde

ALTER COLUMN Nachname
VARCHAR(100)
```

Quelle: ISO/IEC 9075-4:2003 (E) 4.10 SQL-Statements

Beispiel: ALTER TABLE

SQL:2003

```
ALTER TABLE Kunde
ADD COLUMN
        Nachname CHAR(50)
```

```
ALTER TABLE Kunde
ALTER COLUMN
        Nachname VARCHAR(100)
```

MySQL

```
ALTER TABLE Kunde
ADD
        Nachname CHAR(50);
```

```
ALTER TABLE Kunde
MODIFY
        Nachname VARCHAR(100);
```

Oracle

```
ALTER TABLE Kunde
ADD (
        Nachname CHAR(50)
);
```

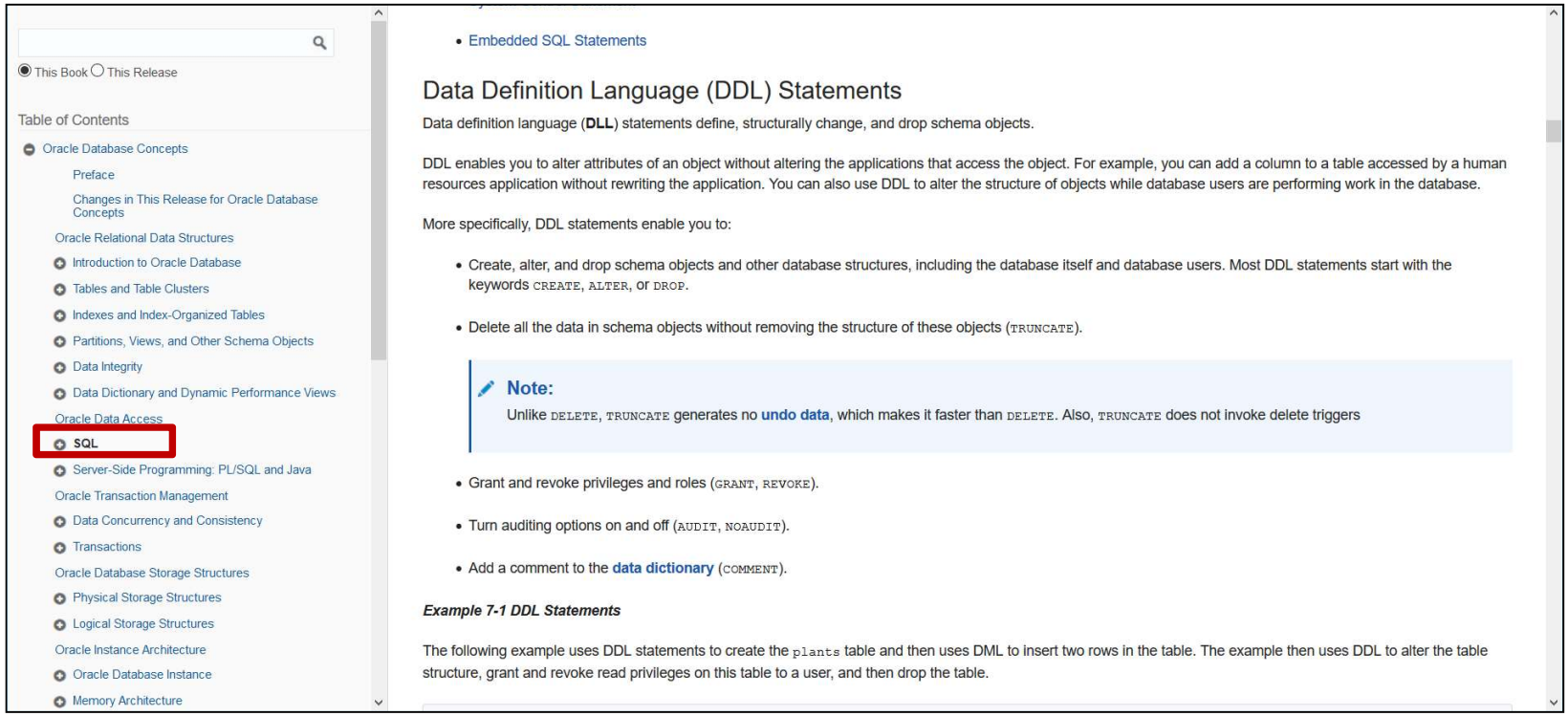
```
ALTER TABLE Kunde
MODIFY(
        Nachname VARCHAR(100)
);
```

Abweichungen vom Standard sind blau markiert.

Dokumentation Oracle 12c

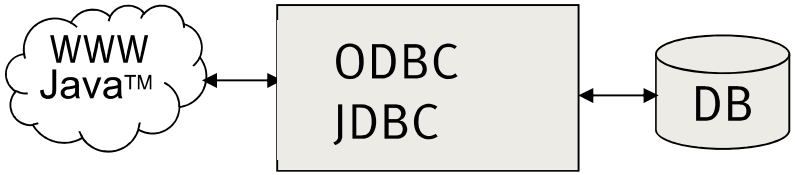
Oracle Database Online Documentation 12c

https://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT1732



Zusammenfassung

SQL/1 1986, SQL/2 1992, SQL/3 1999 ... SQL:2011

SQL/Foundation		**Weitere Teile ...**
***D**ata **D**efinition **L**anguage:* Create Drop Alter	***D**ata **R**etrieval **L**anguage:* Select	DB-Schnittstelle (SQL/CLI) 
***D**ata **M**anipulation **L**anguage:* Insert Update Delete	***D**ata **C**ontrol **L**anguage:* Commit Rollback Grant/Revoke	Datenbankprogrammierung Stored Procedures (SQL/PSM)... und, und, und, ...

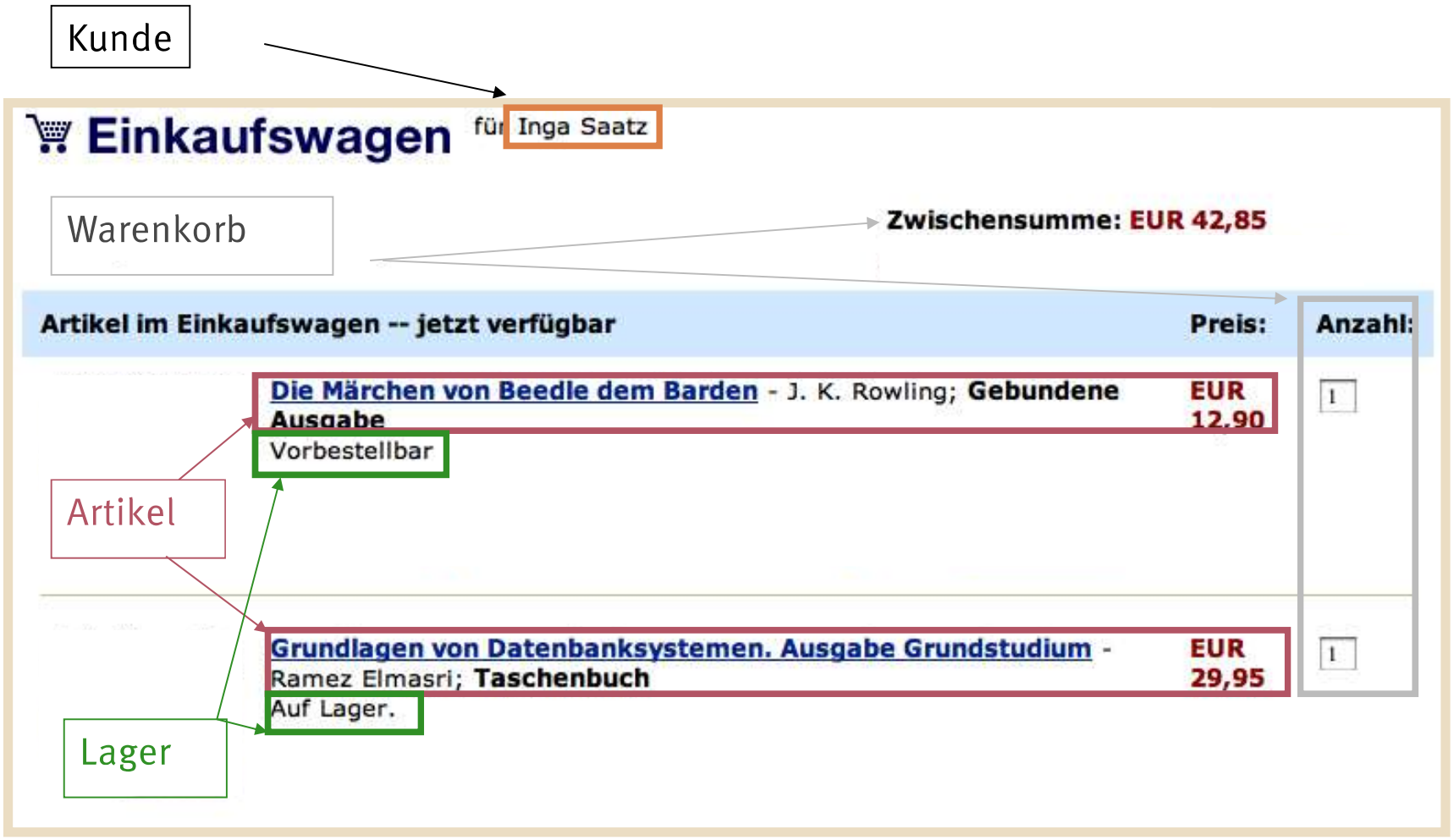
Umgang mit Tabellen (1)

(Basis-)Tabellen anlegen, SQL-Datentypen, NULL-Wert

Prof. Dr. Inga Marina Saatz

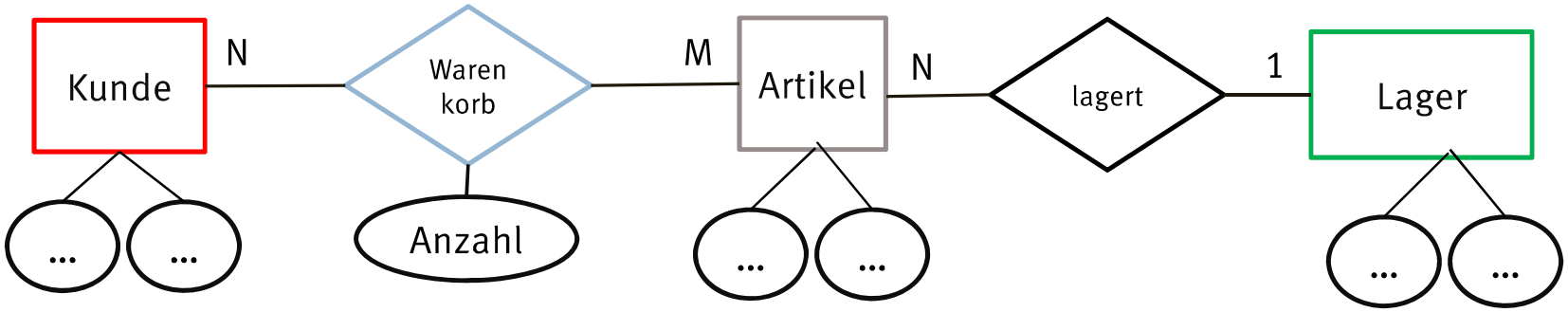
Fachhochschule Dortmund
University of Applied Sciences and Arts

Beispiel


Kunde
Einkaufswagen
für Inga Saatz
Warenkorb
Zwischensumme: EUR 42,85
Artikel im Einkaufswagen -- jetzt verfügbar
Preis:
Anzahl:
Die Märchen von Beedle dem Barden - J. K. Rowling; Gebundene Ausgabe
EUR 12,90
1
Vorbestellbar
Artikel
Grundlagen von Datenbanksystemen. Ausgabe Grundstudium - Ramez Elmasri; Taschenbuch
EUR 29,95
1
Auf Lager.
Lager

Zusammenfassendes Beispiel

- Entity-Relationship Diagramm (ERD) Chen-Notation



- Relationales Modell

Kunde	Kunden-nummer	Anrede	Nach-name	Geburts-datum	...

Warenkorb	Kunden-nummer	Artikel-nummer	Anzahl

Artikel	Artikel-nummer	Artikel-bezeichnung	Preis	...

Lager	Lager-nummer	Standort	ANummer	...

Primär- und Fremdschlüssel

Kunde	**Kunden-nummer**	**Anrede**	**Nach-name**	**Geburts-datum**

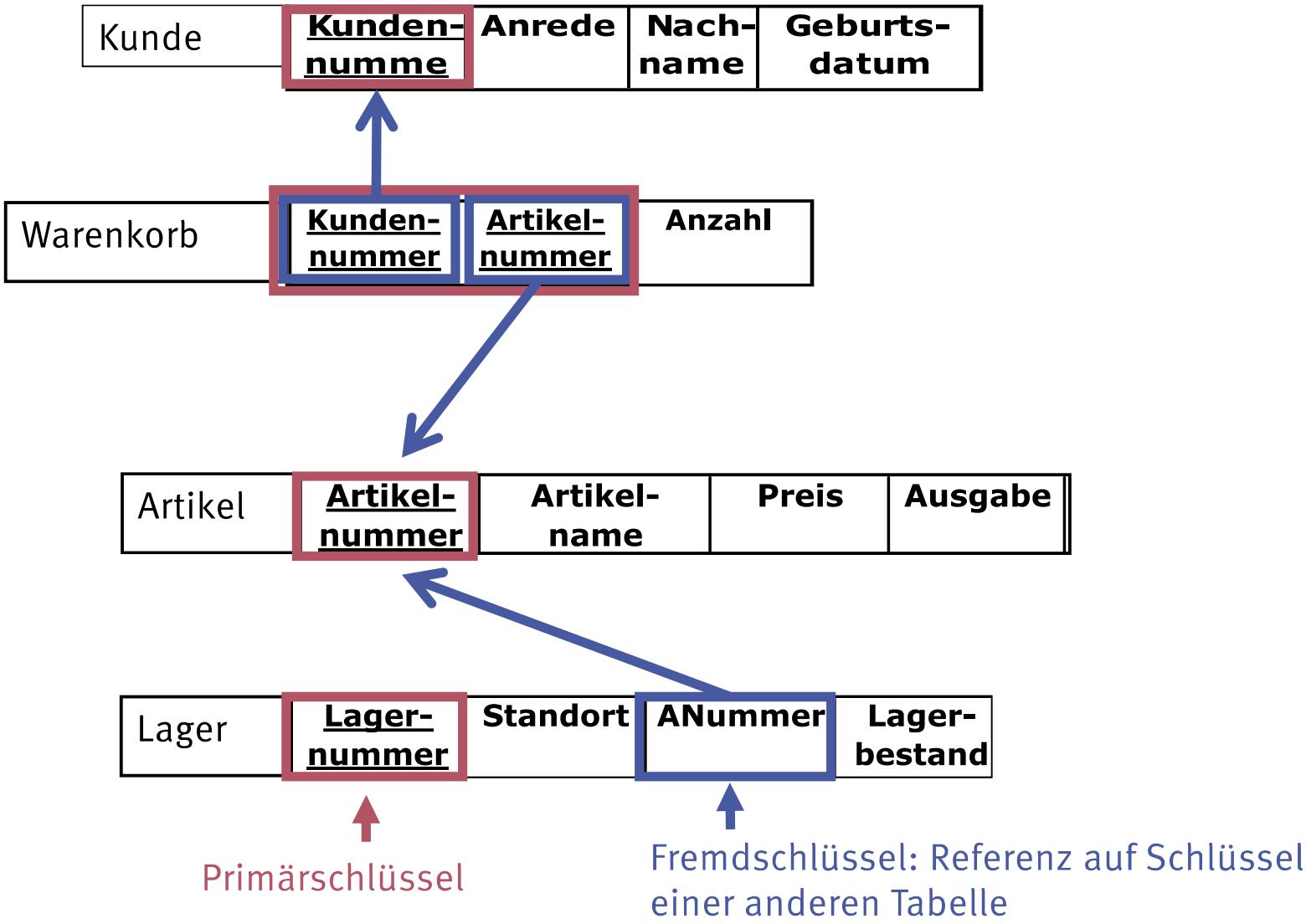
Warenkorb	**Kunden-nummer**	**Artikel-nummer**	**Anzahl**
	2310	4812	1
	2310	4813	1
	1003	4813	5

Artikel	**Artikel-nummer**	**Artikel-name**	**Preis**	**Ausgabe**

Lager	**Lager-nummer**	**Standort**	**ANummer**	**Lager-bestand**

Primärschlüssel

Primär- und Fremdschlüssel

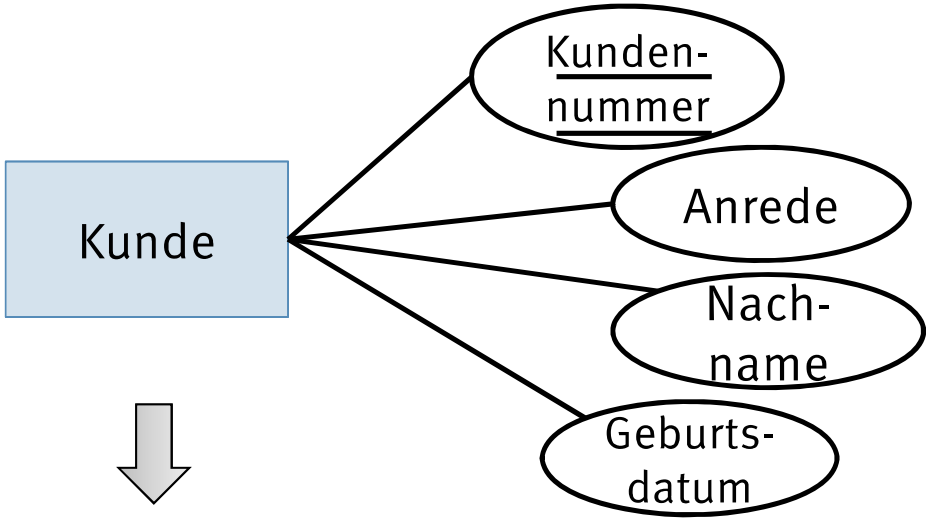


Tabellen anlegen

Der `CREATE TABLE`-Befehl und die Basistabelle

Modellierungsschritte

ER-Diagramm:

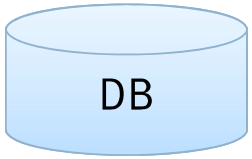


Relationales Modell:

Kunde	Kunden-nummer	Anrede	Nach-name	Geburts-datum

SQL Datenbankschema definieren

Physisches Datenmodell



Tabellendefinition

Festgelegt werden ...

- Tabellenbezeichnung
- die Spaltennamen
- Reihenfolge der Spalten
- Datentypen der Spalten
- Default-Werte
- Integritätsbedingungen

Kunde

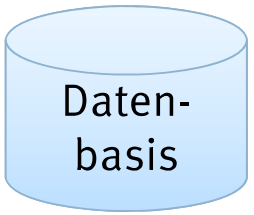
Kunden-nummer	**Anrede**	**Name**	**Geburts-datum**

```
CREATE TABLE Kunde(
Kundennummer
                INTEGER
                UNIQUE NOT NULL,
Anrede          CHARACTER(4),
Nachname        CHARACTER(30)
                DEFAULT 'N.N.',
Geburtsdatum
                DATE
)
```

Basistabelle

- Eine Basistabelle wird erzeugt durch eine **CREATE TABLE** Anweisung
- Die Struktur und Extension ist physisch in der Datenbank gespeichert

Kunden-nummer	Anrede	Nachname	Geburts-datum
2310	Frau	Meitner	17.11.1878
7562	Herr	Einstein	14.03.1879
8365	Frau	Curie	07.11.1867



Unterschiede zum Relationalen Modell:

- ✓ Identische Tupel sind erlaubt. Dies ist allerdings *nicht* möglich bei der Verwendung eines Primärschlüssels
- ✓ Die Reihenfolge der Tupel ist festgelegt
  Beispiel: „select top 10“ ist möglich
- ✓ Die Reihenfolge der Attribute ist definiert.
  Zum Umsortieren der Attribute ist eine neue SQL-Tabelle erforderlich

Tabellen anlegen

Datentypen in SQL

SQL-Datentypen

MySQL-spezifische Erweiterung

Zahlen		
Ganzzahl	*TINYINT*	*1 Byte (-128 ... 127)*
	SMALLINT	2 Byte (-32768 ... 32767)
	INTEGER bzw. INT	4 Byte
	BIGINT	*8 Byte*
Festkommazahl	NUMERIC[(p,[n])]	NUMERIC(12,3): 000432761,432
	DECIMAL[(p,[n])]	NUMERIC(6): 432761
Gleitkommazahl	FLOAT[(p)], REAL	FLOAT(12) 00432761,4320

Datum	DATE	YYYY-MM-DD
Uhrzeit	TIME	HH:MM:SS
Zeitstempel	TIMESTAMP	YYYY-MM-DD HH:MM:SS
Zeitintervall	TIMESTAMP INTERVAL	

Wahrheitswert	BOOLEAN	true, false

p: Genauigkeit n: Anzahl der (Nachkomma-)Stellen

SQL-Datentypen

Zeichenketten	CHARACTER[(n)] VARCHAR[(n)]	CHAR(12): ‘432761,432 _ _ ‘ VARCHAR(12): ‘432761,432‘
Bitfolgen	BIT(n) BIT VARYING(n)	BIT(9): ‘1001001_ _‘ BIT VARYING(9): ‘1001001‘

Beispiele:

Wert	CHAR(4)	Erforderlicher Speicherplatz	VARCHAR(4)	Erforderlicher Speicherplatz
''	' '	4 Byte	''	1 Byte
'ab'	'ab '	4 Byte	'ab '	3 Byte
'abcd'	'abcd'	4 Byte	'abcd'	5 Byte
'abcdefgh'	'abcd'	4 Byte	'abcd'	5 Byte

Tabellen anlegen

Der NULL-Wert

Der NULL-Wert

Kunden-nummer	Anrede	Name	Geburts-datum
2310	Frau	Meitner	07.11.1878
7562	**NULL**	FH Dortmund Dekanat Informatik	**NULL**

NULL : Kein Wert zugeordnet oder zutreffend

Ternäre Logik:
- WAHR (W)
- FALSCH (F)
- NULL

- NULL-Werte können in Tabellen **eingefügt**, **ausgelesen** und **gelöscht** werden.
- Mit NULL-Wert können Tabellen mit sinnvollen, wenn auch teilweise mit unbekannten Daten gefüllt werden.
- Problematik:
  Beim Zählen der Tabelleneinträge werden Null-Werte nicht mitgezählt

  Beispiel:
  Select Count(Kundennummer) FROM Kunden → 2
  Select Count(Geburtsdatum) FROM Kunden → 1

Wahrheitstabellen

Wahrheitstabelle UND

UND	W	F	NULL
W	W	F	NULL
F	F	F	F
NULL	NULL	F	NULL

Ternäre Logik:

- WAHR (W)
- FALSCH (F)
- NULL

Wahrheitstabelle ODER

ODER	W	F	NULL
W	W	W	W
F	W	F	NULL
NULL	W	NULL	NULL

Wahrheitstabelle NICHT

	NICHT
W	F
F	W
NULL	NULL

Zusammenfassung

Zusammenfassung

Tabellen werden angelegt mit dem CREATE TABLE Befehl:

```
CREATE TABLE Kunden (
       Kundennummer INT UNIQUE NOT NULL,
       Anrede       CHARACTER(4),
       Name         VARCHAR(30) DEFAULT 'N.N.' ,
       Geburtsdatum DATE
       )
```

Dabei werden definiert:

- **Tabellenbezeichnung** — Kunde
- **Spaltennamen** — Kundenummer, Anrede, Name, Geburtsdatum
- **Reihenfolge der Spalten** — Durch Reihenfolge der Aufzählung im Befehl
- **Datentypen der Spalten** — INT, CHARACTER, etc.
- **Default-Werte** — optionale Start-Werte
- **Integritätsbedingungen** — UNIQUE, NOT NULL, CHECK... (siehe Teil 2)

Unterschiede der Basistabelle zum Relationenmodell:

» Identische Tupel sind erlaubt
(nur solange kein Primärschlüssel vorhanden ist)

» Festgelegte Tupel- und Attribut-Reihenfolge

SQL-Datentypen (Auswahl):

INT	DATE	VARCHAR(n)	BOOLEAN
NUMERIC (p, n)	TIME	CHARACTER(n)	
DECIMAL (p, n)	TIMESTAMP		BIT(n)
FLOAT (p)			BITVARYING(n)

Der NULL-Wert

» Ist ein Wert !

» Bedeutung: Es ist kein passender Wert vorhanden oder bekannt

» Kann eingefügt, ausgelesen, gelöscht oder explizit gezählt werden

» Wird beim Zählen von Tabelleneinträgen nicht mitgezählt

» Die ternäre Logik des NULL-Wertes beachten

Umgang mit Tabellen (2)

Semantische Integrität, Primär- und Fremdschlüssel definieren, „Sauberes“ Löschen von Werten und Tabellen (Referentielle Integrität), Installationsskript

---

Prof. Dr. Inga Marina Saatz

Fachhochschule Dortmund
University of Applied Sciences and Arts

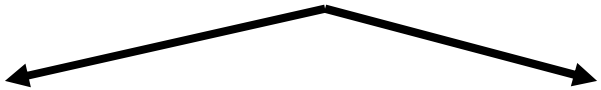
Tabellen anlegen

Die semantische Integrität gewährleisten mit

- `CHECK`
- `CONSTRAINT`
- `UNIQUE`
- `NOT NULL`

Semantische Integrität

Syntax: `[CONSTRAINT` <Bezeichner>`]`
`CHECK (`<Integritätsbedingung>`)`



Spaltenbedingung

```
CREATE TABLE Kunde (
   ...
Anrede CHARACTER(4)
   CHECK (
      Anrede
         IN ('Herr', 'Frau') ),
   ...
)
```

Tabellenbedingung

```
CREATE TABLE Kunde (
   ...
Anrede CHARACTER(4), ...
CONSTRAINT pruefeAnrede
   CHECK (
         Anrede
            IN ('Herr', 'Frau')),
)
```

Weiteres Beispiel für eine Bedingung: PLZ BETWEEN 0 AND 99999

Anmerkung: MySQL *akzeptiert* Check-Klauseln zwar, *prüft* sie jedoch nicht.

Tabellendefinition

Festzulegen:

- Tabellenbezeichnung
- die Spaltennamen
- Reihenfolge der Spalten
- Datentypen der Spalten
- Default-Werte
- Integritätsbedingungen

Kunde

Kunden-nummer	Anrede	Name	Geburts-datum

```
CREATE TABLE Kunde(
Kundennummer
                    INTEGER
                    UNIQUE NOT NULL,
Anrede              CHARACTER(4)
                    CHECK(…),
Nachname            CHARACTER(30)
                    DEFAULT ‘N.N.‘,
Geburtsdatum
                    DATE
)
```

Tabellen anlegen

Primär- und Fremdschlüssel definieren (Einfach und zusammengesetzt)

Tabellendefinition

Festzulegen:

- Tabellenbezeichnung
- die Spaltennamen
- Reihenfolge der Spalten
- Datentypen der Spalten
- Default-Werte
- Integritätsbedingungen
  - **`UNIQUE, NOT NULL`**
  - **`CHECK`**-Klausel*
  - Primär- und Fremdschlüssel

* Wird von MySQL nicht geprüft
→ Trigger erforderlich)

Kunde

Kunden-nummer	**Anrede**	**Name**	**Geburts-datum**

```
CREATE TABLE Kunde(
Kundennummer
                    INTEGER
                    UNIQUE NOT NULL,
Anrede              CHARACTER(4)
                    CHECK(…),
Nachname            CHARACTER(30)
                    DEFAULT ‘N.N.‘,
Geburtsdatum
                    DATE
)
```

Tabellendefinition

Schlüsselkandidat

minimale Attributmenge, die eine Instanz eindeutig identifiziert (**UNIQUE**)

Primärschlüssel

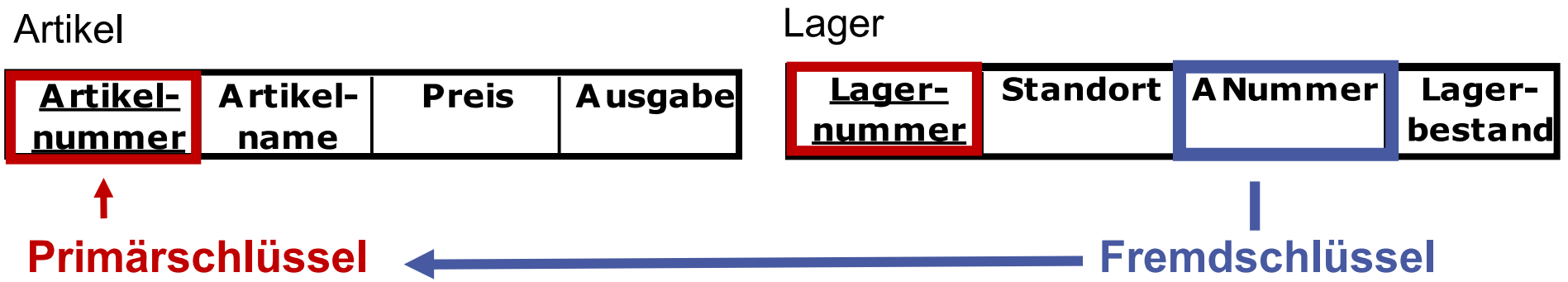
- kann aus mehreren Spalten bestehen
- eindeutig (**UNIQUE**)
- **NOT NULL**
- max. 1x pro Tabelle definierbar

Kunde

Kunden-nummer	**Anrede**	**Name**	**Geburts-datum**

```
CREATE TABLE Kunde(
Kundennummer
                INTEGER
                ~~UNIQUE NOT NULL~~,
Anrede          CHARACTER(4)
                CHECK(…),
Nachname        CHARACTER(30)
                DEFAULT 'N.N.',
Geburtsdatum
                DATE,
PRIMARY KEY (Kundennummer)
)
```

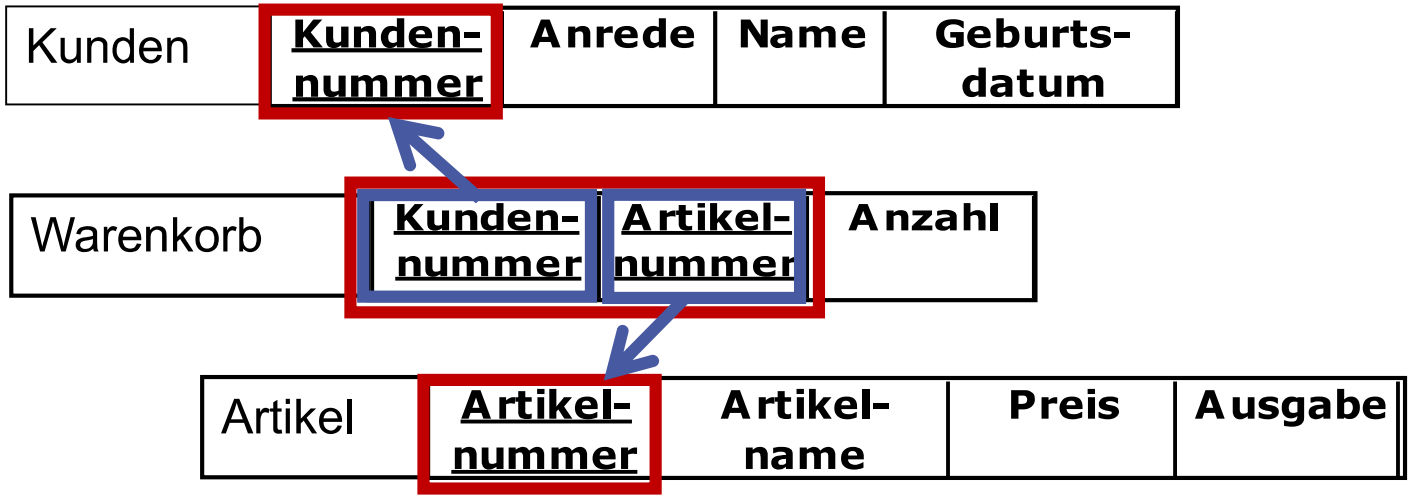
Fremdschlüssel definieren




```
CREATE TABLE Artikel(
Artikelnummer  INTEGER,
...
PRIMARY KEY Artikelnummer
)
```

```
CREATE TABLE Lager(
Lagernummer    INTEGER,
...
ANummer        INTEGER,
...
PRIMARY KEY Lagernummer,
FOREIGN KEY(ANummer)
  REFERENCES Artikel(Artikelnummer))
```

Zusammengesetzter Schlüssel




```
CREATE TABLE Warenkorb(
Kundennummer     INTEGER,
Artikelnummer    INTEGER,
Anzahl           INTEGER CHECK( Anzahl >=0)
PRIMARY KEY
FOREIGN KEY
   REFERENCES
FOREIGN KEY
   REFERENCES
)
```

Tabellen anlegen

Referentielle Integrität: Datenschutz und Kontrolle beim Löschen

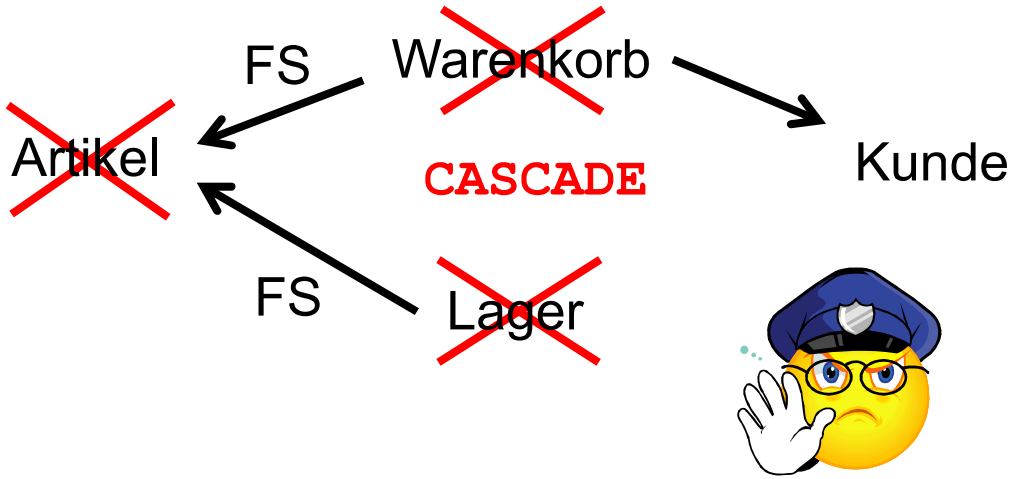
Tabellen löschen

DROP TABLE <tabellenname>
[**RESTRICT** | **CASCADE**]

RESTRICT: Löschen zurückweisen

CASCADE: Löschen abhängender Datenbankobjekte

DROP TABLE Artikel
CASCADE



Ein Installationsskript anlegen

Installationsskript


```
CREATE SCHEMA IF NOT EXISTS Buchhandlung;
USE Buchhandlung;
--  Löschen der vorhandenen Tabellen
DROP TABLE IF EXISTS Warenkorb;
--  Erstellen der Tabellen (Reihenfolge ist wichtig!)
CREATE TABLE Kunden(
Kundennummer  INTEGER PRIMARY KEY,
Anrede        CHARACTER(4)   CHECK (Anrede IN ('Herr','Frau',Null)),
Name          CHARACTER(30)  DEFAULT 'N.N.',
Vorname       CHARACTER(50),
Geburtsdatum  DATE
)ENGINE=InnoDB;    ⟵ Erzwingt referentielle Integrität

-- Einfügen der Kunden
INSERT INTO Kunden VALUES (2310, 'Frau', 'Meitner','Lise','1878-11-17','Berlin');
```

Was passiert, wenn …

… die Kundennummer fehlt?

… bereits vorhanden ist?

Zusammenfassung

Zusammenfassung: Tabellen anlegen

Die semantische Integrität wird beim Anlegen von Tabellen gewährleistet durch Verwendung von

» **CHECK als Spaltenbedingung:**

```
CREATE TABLE Kunden ( …
    Anrede CHARACTER(4)
           CHECK (Anrede IN ('HERR','FRAU',NULL) ),
    … )
```

» **in Verbindung mit CONSTRAINT als Tabellenbedingung:**

```
CREATE TABLE Kunden ( …
    Anrede CHARACTER(4),
    …
    CONSTRAINT pruefeAnrede
           CHECK (Anrede IN ('HERR','FRAU',NULL) ),
    … )
```

Zusammenfassung: Tabellen anlegen

Weitere semantische Integritätsbedingungen:

» `UNIQUE`

» `NOT NULL`

Zusammenfassung: Tabellen anlegen

Primärschlüssel:

» Identifiziert die Instanzen/Tupel der Tabelle eindeutig
→ Maximal 1x pro Tabelle definierbar!

» Kann aus mehreren Spalten/Attributen bestehen (Zusammengesetzter Primärschlüssel)

» Umfasst die Eigenschaften `NOT NULL` und `UNIQUE`

Fremdschlüssel:

» Verweis auf Primärschlüssel anderer Tabellen

Zusammenfassung: Tabellen anlegen

Beispiel mit zusammengesetztem Primärschlüssel, bestehend aus zwei Fremdschlüsseln:

```
CREATE TABLE Warenkorb( …
   Kundennummer   INTEGER,
   Artikelnummer  INTEGER,
   Anzahl         INTEGER CHECK(Anzahl>=0),

   PRIMARY KEY (Kundennummer, Artikelnummer),

   FOREIGN KEY (Artikelnummer)
               REFERENCES Artikel(Artikelnummer),

   FOREIGN KEY (Kundennummer)
               REFERENCES Kunden(Kundennummer),
   … )
```

Zusammenfassung: Tabellen anlegen

Wahrung der referentiellen Integrität:

» **`ON DELETE` für Kontrolle über die Auswirkungen von Löschaufträgen auf referenzierte Tabellen:**

```
CREATE TABLE Lager ( …
  FOREIGN KEY (ANummer) REFERENCES Artikel(Artikelnummer)
                        ON DELETE SET NULL
… )
```

» **`ON DELETE` Optionen:**

» `RESTRICT`	Löschen verhindern, wenn Referenzen vorhanden (Standard, auch ohne Angabe)
» `SET NULL`/ `SET DEFAULT`	Referenzen durch NULL oder einen vorgegebenen Wert ersetzen
» `CASCADE`	Alle referenzierenden Datensätze löschen

Wahrung der referentiellen Integrität

» **`DROP TABLE` Ergänzung für Kontrolle über die Auswirkung von Löschaufträgen auf referenzierendeTabellen:**

```
DROP TABLE Lager RESTRICT
```

» **`DROP TABLE` Optionen:**

- » `RESTRICT` Löschen verhindern, wenn es abhängige Objekte gibt (Standard, auch ohne Angabe)
- » `CASCADE` Alle referenzierenden Tabellen ebenfalls löschen

we
focus
on
students

DML-Operationen

Tupel einfügen, ändern und löschen

Fachhochschule
Dortmund
University of Applied Sciences

Fragestellung

Kunde	Kunden-nummer	Vorname	Nach-Name
	8365	Marie	Curie

Wie können Tupel eingefügt, geändert und gelöscht werden?

Insert Update Delete

Tupel einfügen

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Marie	Sklodowska

```
INSERT INTO  ‹tabellenname›  [(spalte1, ..., spalteN)]
{
  VALUES (wert1, ..., wertN)
}
```

Tupel einfügen

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Marie	Sklodowska

```
INSERT INTO  ‹tabellenname›  [(spalte1, ..., spalteN)]
{
  VALUES (wert1, ..., wertN)
}
```

Syntaxbeschreibung

Beispiel

```
INSERT INTO Kunde
VALUES (8365, 'Marie', 'Sklodowska')
```

Tupel einfügen

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Marie	Sklodowska
	8366	NULL	Meitner

```
INSERT INTO  ‹tabellenname›  [(spalte1, ..., spalteN)]
{
  VALUES (wert1, ..., wertN)
}
```

Syntaxbeschreibung

```
INSERT INTO Kunde (Kundennummer, Nachname)
VALUES (8366, 'Meitner')
```

Tupel ändern

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Marie	Sklodowska
	8366	NULL	Meitner

```
UPDATE ‹tabellenname›
SET  ‹attribut1 = wert1› , ...
       ‹attributN = wertN›
[WHERE  ‹Bedingung›]
```

⇩

```
UPDATE  Kunde
SET  Vorname = 'Lise'
```

Tupel ändern

Kunde

Kunden-nummer	Vorname	Nach-name
8365	Marie	Sklodowska
8366	~~NULL~~ Lise	Meitner

Tupel ändern

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Lise	Sklodowska
	8366	Lise	Meitner

```
UPDATE ‹tabellenname›
SET  ‹attribut1 = wert1› , ...
       ‹attributN = wertN›
[WHERE  ‹Bedingung›]
```

⇩

```
UPDATE  Kunde
SET  Vorname = 'Lise'
```

WHERE fehlt!

Tupel ändern

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Marie	Curie
	8366	Lise	Meitner

```
UPDATE ‹tabellenname›
SET ‹attribut1 = wert1› , ...
      ‹attributN = wertN›
[WHERE ‹Bedingung›]
```

⇩

```
UPDATE  Kunde
SET  Vorname = 'Marie', Nachname= 'Curie'
WHERE  Kundennummer=8365
```

Tupel löschen

Kunde	Kunden-nummer	Vorname	Nach-name
	8365	Marie	Curie
	~~8366~~	~~Lise~~	~~Meitner~~

DELETE FROM ‹tabellenname›
[**WHERE** ‹Bedingung›]

⇩

```
DELETE FROM Kunde
WHERE Kundennummer=8366
```

Zusammenfassung

Wie können Tupel eingefügt, geändert und gelöscht werden?

Einfügen

```
INSERT INTO  ‹tabellenname› [(spalte1, ..., spalteN)]
{
  VALUES (wert1, ..., wertN)
}
```

Ändern

```
UPDATE ‹tabellenname›
SET  ‹attribut1 = wert1› , ...
     ‹attributN = wertN›
[WHERE ‹Bedingung›]
```

Löschen

```
DELETE FROM ‹tabellenname›
[WHERE ‹Bedingung›]
```

Zusammenfassung

Wie können Tupel eingefügt, geändert und gelöscht werden?

Einfügen

```
INSERT INTO  ‹tabellenname› [(spalte1, ..., spalteN)]
{
  VALUES (wert1, ..., wertN)
}
```

Ändern

```
UPDATE ‹tabellenname›
SET  ‹attribut1 = wert1› , ...
     ‹attributN = wertN›
[WHERE ‹Bedingung›]
```

Löschen

```
DELETE FROM ‹tabellenname›
[WHERE ‹Bedingung›]
```

Syntaxbeschreibung

we focus on students

Datenbankadministration

Benutzerverwaltung

Fachhochschule Dortmund
University of Applied Sciences