we focus on students

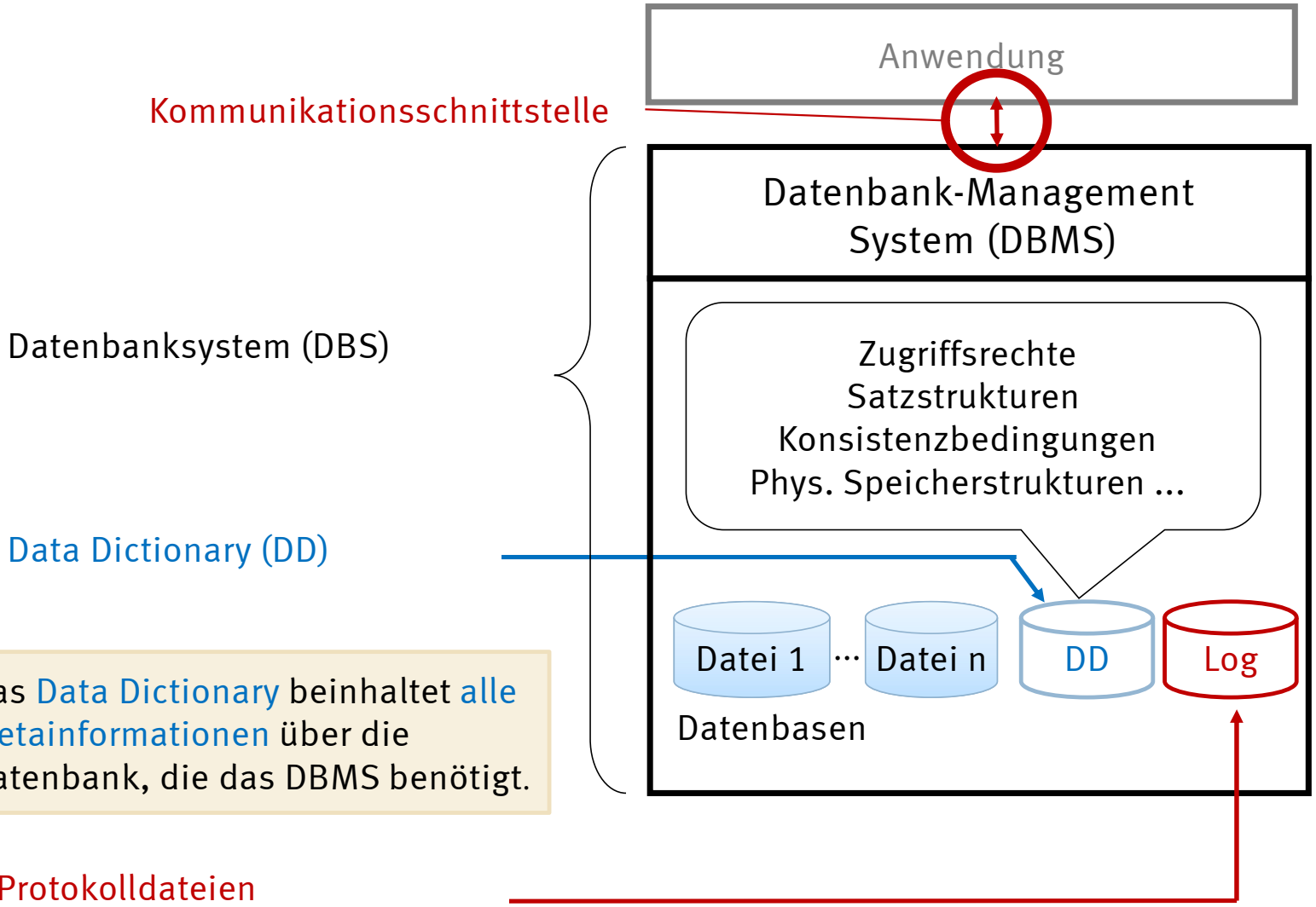
Datenbanken 1

Erste Schritte mit SQL

Fachhochschule Dortmund
University of Applied Sciences

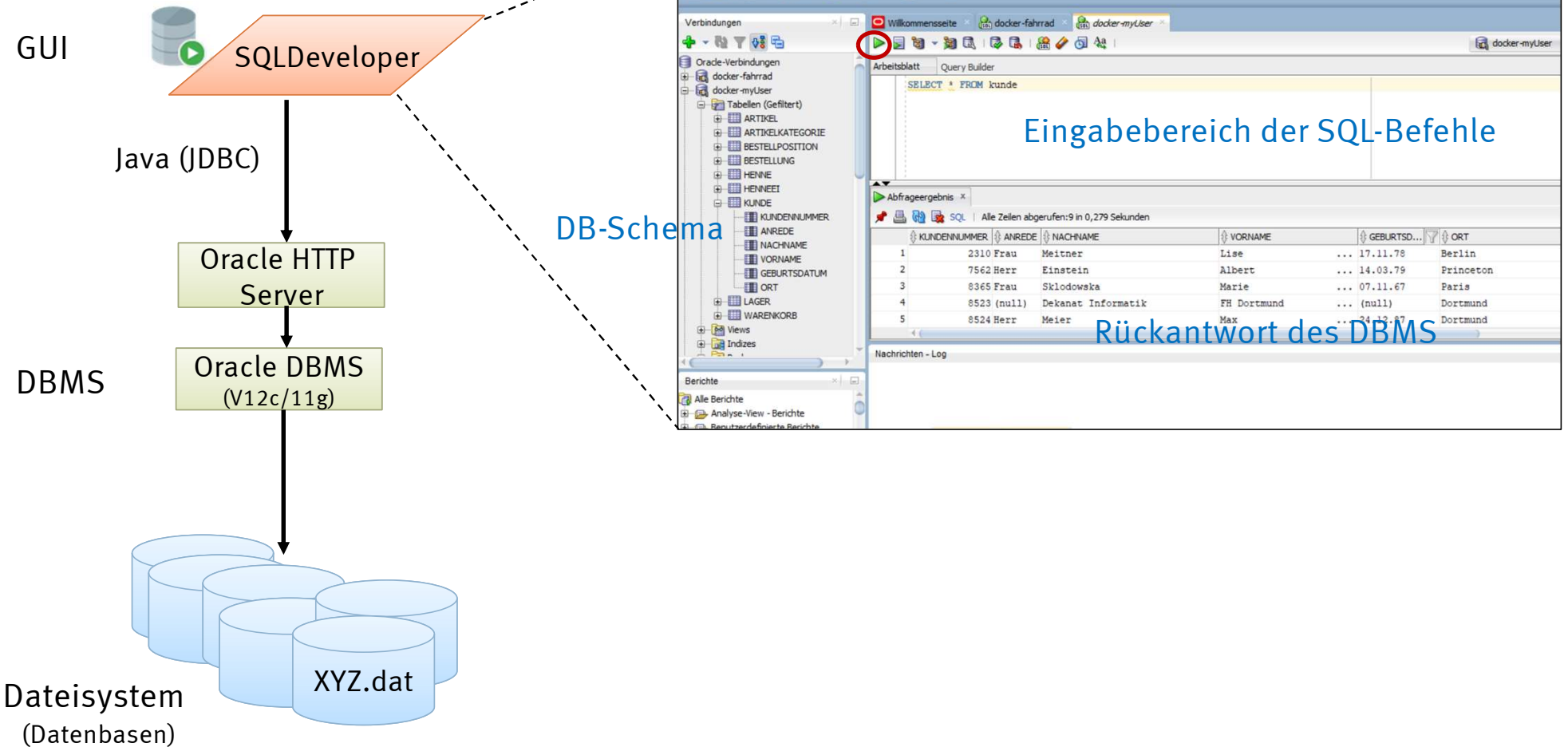
Inhaltsübersicht

Begriffe



Das Data Dictionary beinhaltet alle Metainformationen über die Datenbank, die das DBMS benötigt.

Kommunikation mit dem DBMS



Structured Query Language (SQL)

Normierte, mächtige Anfragesprache (SQL)

- ✓ Standards zur Datendefinition und -abfrage
- ✓ deskriptive Problemformulierung
- ✓ einfache Verknüpfung mehrerer Satztypen
- ✓ hohe Auswahlmächtigkeit

Kundennummer	Artikelnummer	Anzahl	Preis	PreisGesamt
1234	1111	1	5,95 €	5,95 €
1234	1112	2	24,95 €	49,90 €

Feld:	Kundennummer	Artikelnummer	Anzahl	Preis	PreisGesamt: [Warenkorb].Anzahl*
Tabelle:	Warenkorb	Warenkorb	Warenkorb	Artikel	
Funktion:	Gruppierung	Gruppierung	Gruppierung	Gruppierung	Gruppierung
Sortierung:					
Anzeigen:	☑	☑			

```
SELECT Kundennummer, Artikelnummer, ...
FROM Artikel ...
WHERE ...
```

Deklarative Anfragesprache

SQL (**S**tructured **Q**uery **L**anguage)

SQL/1 1986, SQL/2 1992, SQL/3 1999 ... SQL:2016

SQL/Foundation

Data Definition Language:
- Create
- Drop
- Alter

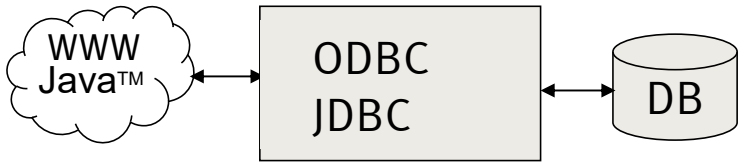
Data Manipulation Language:
- Insert
- Update
- Delete

Data Retrieval Language:
- Select

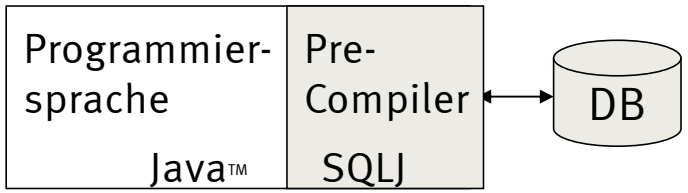
Data Control Language:
- Commit
- Rollback
- Grant/Revoke

Weitere Teile ...

DB-Schnittstelle (SQL/CLI)



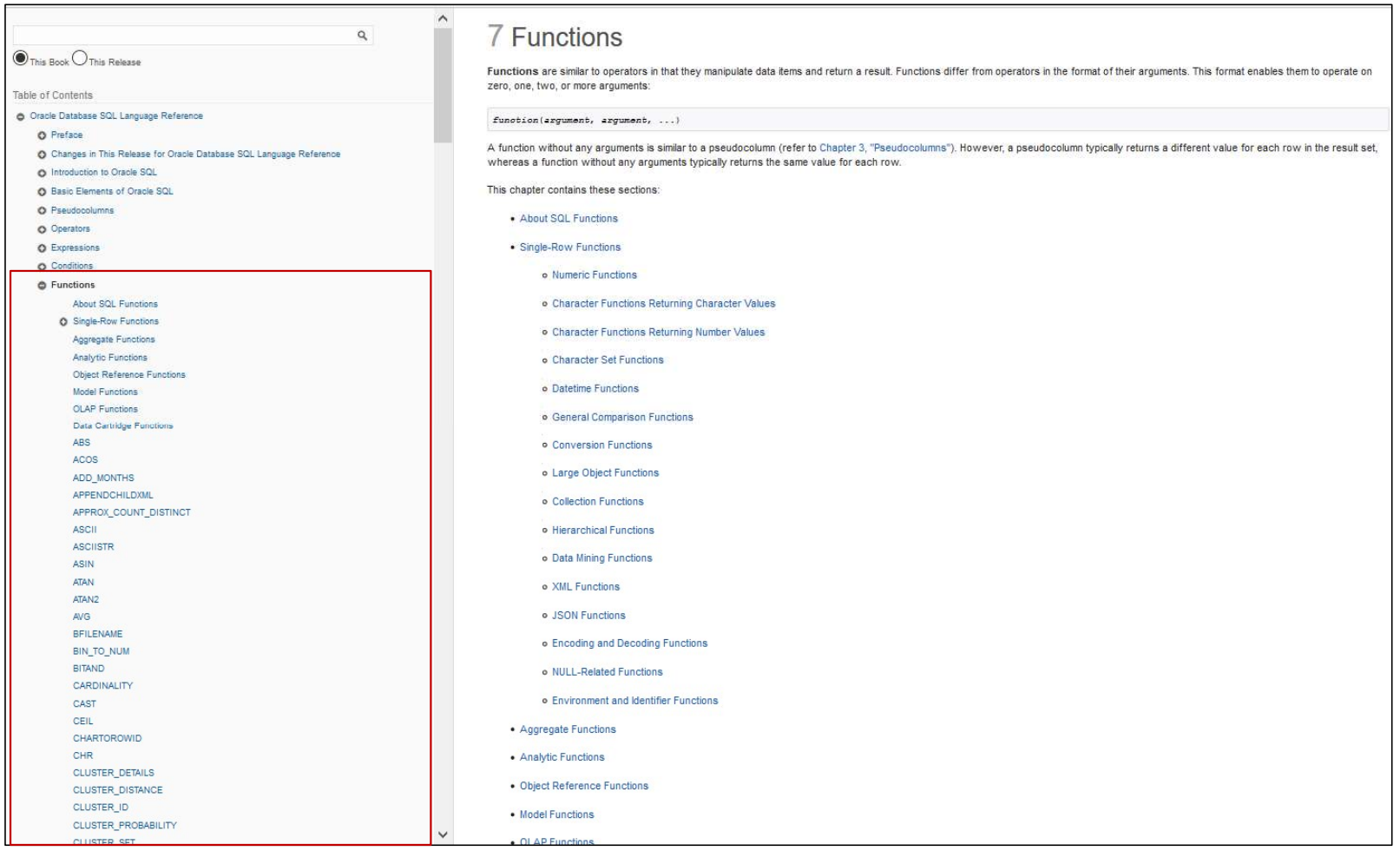
Embedded SQL (SQL/OLB)



XML und SQL (SQL/XML)

Stored Procedures (SQL/PSM)...

Oracle Dokumentation



https://docs.oracle.com/database/121/SQLRF/functions.htm#SQLRF006

Inhaltsübersicht

Structured Query Language (SQL)

Abfrage des Tabellenschemas:

DESCRIBE Kunde;

Schema

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

Abfrage der Tabellenextension:

SELECT * FROM Kunde;

Der NULL-Wert

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

Kunden-nummer	Anrede	Nachname	Geburts-datum
4812	Frau	Meitner	07.11.1878
7562	NULL	FH Dortmund Dekanat Informatik	NULL

Kein Wert zugeordnet oder zutreffend

- **NULL** ist nicht die Abwesenheit von Werten, sondern besitzt eine besondere Semantik.
- Mit **NULL**-Wert können Tabellen mit sinnvollen, wenn auch teilweise unbekannten Daten gefüllt werden.
- **NULL**-Werte können in Tabellen eingefügt, ausgelesen und gelöscht werden.
- Mit **NULL**-Werten kann gerechnet werden:
  NULL + 11 = Null

- Problematik:
  Beim Zählen der Tabelleneinträge werden **NULL**-Werte nicht mitgezählt
- Beispiel:
  - Zählen der Kundennummern liefert: 2
  - Zählen der Geburtsdaten liefert: 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

Inhaltsübersicht

Tabellendefinition

Kunde

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

Tabelle erstellen mit

- Tabellenbezeichnung
- die Spaltennamen
- Reihenfolge der Spalten
- Datentypen der Spalten
- statische Integritätsbedingungen
  (hier: Primärschlüsseldefinition)

```
CREATE TABLE Kunde(
Kundennummer
                    INTEGER,
Anrede              CHARACTER(4),
Nachname            CHARACTER(30),
Geburtsdatum        DATE
Primary Key (Kundennummer)
)
```

Tabelle löschen

```
DROP TABLE Kunde
```

Tabelle umbenennen

```
RENAME TABLE Kunde TO KundeXY
```

Vergleich von Zeichenketten

- Das Ergebnis einer Vergleichsoperationen kann von der Wahl des Datentyps abhängen.

Beispiel Oracle:

```
declare
s1 char(5) := '12';   -- ohne nachfolgende Leerzeichen
s2 char(5) := '12 ';  -- mit nachfolgenden Leerzeichen
```

=› s1 = s2 ist TRUE

```
declare
s1 varchar2(5) := '12';   -- ohne Leerzeichen
s2 varchar2(5) := '12 ';  -- mit Leerzeichen
```

=› s1 = s2 ist FALSE

Ursache:

- Bei Zuweisungen werden CHAR-Variablen mit Leerzeichen aufgefüllt.
- Bei Zuweisungen werden VARCHAR-Variablen nicht mit Leerzeichen aufgefüllt.

Vorsicht Falle!

SQL:2003

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

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

MySQL

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

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

Oracle

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

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

Abweichungen vom Standard sind blau markiert.

Inhaltsübersicht

Einfügen von Daten

Kunde

Kunden-nummer	Anrede	Nach-name	Geburts-datum
2310	Frau	Meitner	17.11.1878

Syntax

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

Kopieren von Daten

Beispiel

```
INSERT INTO Kunde (Kundennummer, Anrede, Nachname, Geburtsdatum)
VALUES (2311,'Frau','Meitner', TO_DATE('17.11.1878'))
```

Wirkung:
- Integritätsbedingungen (z.B. Eindeutigkeit des Primärschlüsselwerts) werden beim Einfügen überprüft
- Fehlende Werte werden mit NULL belegt (falls dies erlaubt ist)

Ändern von Daten

Kunde

Kunden-nummer	Anrede	Nach-name	Geburts-datum
2310	Frau	Meitner	17.11.1878
8365	Frau	~~Sklodowska~~ Curie	07.11.1867

Syntax

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

Beispiel

```
UPDATE Kunde
SET Nachname ='Curie‘
WHERE Kundennummer =8365
```

Merke:
Beziehen sich Änderungen nur auf bestimmte Tupel, so muss eine eindeutige Selektionsbedingung (WHERE-Klause) vorhanden sein.

Löschen von Daten

Kunde

Kunden-nummer	Anrede	Nach-name	Geburts-datum
~~2310~~	~~Frau~~	~~Meitner~~	~~17.11.1878~~
8365	Frau	Curie	07.11.1867

Syntax

DELETE FROM ‹tabellenname›
[WHERE ‹Bedingung›]

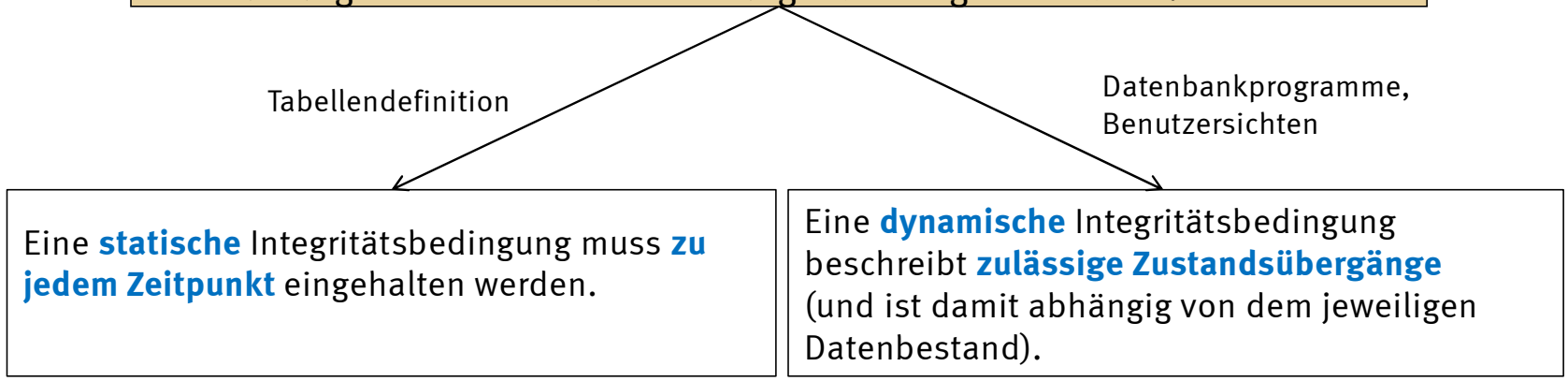
Beispiel

Inhaltsübersicht

Integritätsbedingungen

Datenintegrität:

Die Daten in der Datenbank sollen zu jedem Zeitpunkt die in der Realität geltenden Zusammenhänge und Regeln erfüllen.



Beispiele:

- Die Anrede eines Kunden darf nur die Werte Frau, Herr oder NULL annehmen
- Ein Kunde kann max. 10 gleiche Artikel im den Warenkorb speichern.

Beispiele:

- Ein Artikel kann nur gelöscht werden, wenn kein Lagerbestand vorhanden ist
- Ein Kunde kann max. 10 unterschiedliche Artikel im Warenkorb speichern

Tabellen löschen

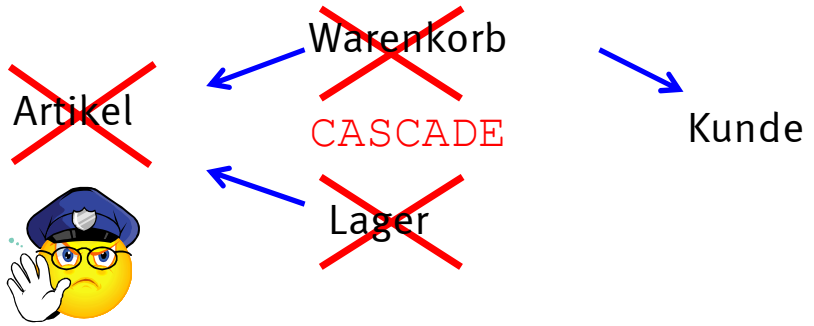
DROP TABLE ‹tabellenname›

[**RESTRICT** | **CASCADE**]

Löschen zurückweisen

Löschen abhängender Datenbankobjekte

```
DROP TABLE Artikel
        CASCADE
```



PostgreSQL	CASCADE löscht alle abhängigen Datenbankobjekte
OracleDB	CASCADE CONSTRAINTS löscht abhängige Integritätsbedingungen
MySQL (Vers. 5)	RESTRICT und CASCADE werden ignoriert

Vorsicht Falle!

Bei Oracle führt die Verwendung des Befehls DROP TABLE ... CASCADE CONSTRAINTS Zu einer Verletzung der referentiellen Integrität.

```
DROP TABLE Artikel
        CASCADE CONSTRAINTS
```

Nur in einem Installationsskript verwenden!

Artikel

Artikel-nummer	Artikel-name	Preis	Ausgabe
4812	Basiswissen	29,95 €	gebunden
4813	Das Ende der	12,90 €	broschiert

Lager

Lager-nummer	Standort	ANummer	Lager-bestand
27135	R235	4812	18
27423	R371	4813	0

???

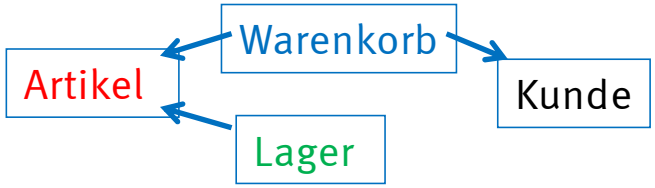
```
Table ARTIKEL gelöscht.

SELECT Lagernummer, ANummer FROM Lager

                                LAGERNUMMER    ANUMMER
---------------------------------------- ----------
                                   27135        4811
                                   27136        4812
```

Aufbau eines Installationsskripts

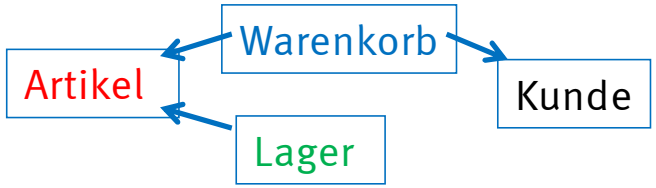
Zugehörende Integritätsbedingungen löschen

```
--
-- Löschen der vorhandenen Tabellen
--
DROP TABLE Warenkorb CASCADE CONSTRAINTS;
DROP TABLE …
--
-- Erstellen der Tabellen (Reihenfolge ist wichtig!)
--
CREATE TABLE Kunde(
Kundennummer INTEGER PRIMARY KEY,
Anrede     CHARACTER(4)
  CHECK (ANREDE IN('Herr', 'Frau')),
Nachname  CHARACTER(30)DEFAULT 'N.N.',
Vorname    CHARACTER(50),
Geburtsdatum DATE
);
--
CREATE TABLE Artikel(
…
```

Reihenfolge beim Löschen:



Reihenfolge beim Erstellen:



Tabellendefinition - statische Integritätsbedingungen

Bei der Tabellendefinition sind festzulegen:

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

Kunde

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

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

Schlüsselattribute

Schlüsselkandidat

min. Attributmenge, die eine Instanz eindeutig identifiziert

(**UNIQUE**)

Primärschlüssel

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

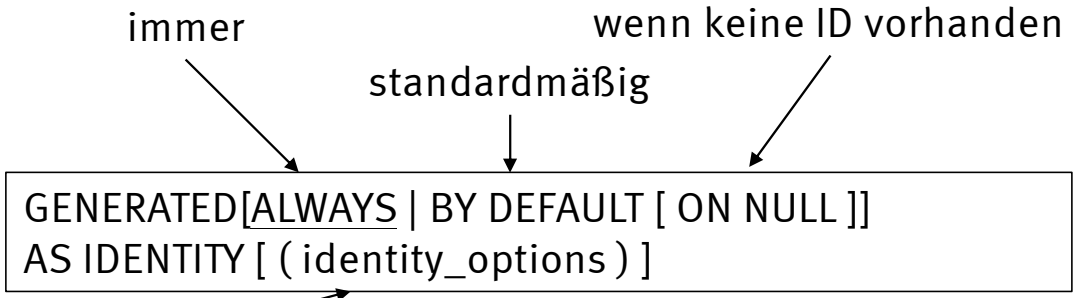
Kunde

Kunden-nummer	**Anrede**	**Nach-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)
)
```

Inkrementelle Schlüsselwerte (ab Oracle 12c)

Ab Oracle 12c gibt es eine Klausel zur Generierung inkrementeller Schlüsselwerte, die intern als Sequence-Objekt gespeichert wird. Es darf max. ein generierter Schlüssel pro Tabelle angelegt werden. Intern wird eine Sequence angelegt.



identity options (für numerische Datentypen):

```
{ START WITH ( integer | LIMIT VALUE )
| INCREMENT BY integer
| ( MAXVALUE integer | NOMAXVALUE )
| ( MINVALUE integer | NOMINVALUE )
| ( CYCLE | NOCYCLE )
| ( CACHE integer | NOCACHE )
| ( ORDER | NOORDER ) }...
```

Beispiel Oracle 12c

```
CREATE TABLE Kunde(
  Kundennummer INT
     GENERATED AS IDENTITY
          INCREMENT BY 2,
  Nachname    VARCHAR(50)
)
```

```
INSERT INTO Kunde VALUES (DEFAULT, 'Curie')
```

Inkrementelle Schlüssel - Oracle

Mit dem Datenbankobjekt Sequence können fortlaufender Nummern (→ Schlüsselwerte) erzeugt werden.

```
CREATE SEQUENCE Sequenzname
        [INCREMENT BY Integer]
        [START WITH Integer]
        [MAXVALUE Integer | NOMAXVALUE]
        [MINVALUE  Integer  | NOMINVALUE]
        [CYCLE | NOCYCLE]
        [CACHE | NOCACHE]
        [ORDER | NOORDER]
```

Schrittweite (Default 1)

Oracle

Inkrementelle Schlüsselwerte 1

Die durch eine Sequenz erzeugten Nummern können als Schlüsselwerte in DML-Befehlen genutzt werden.

Definition

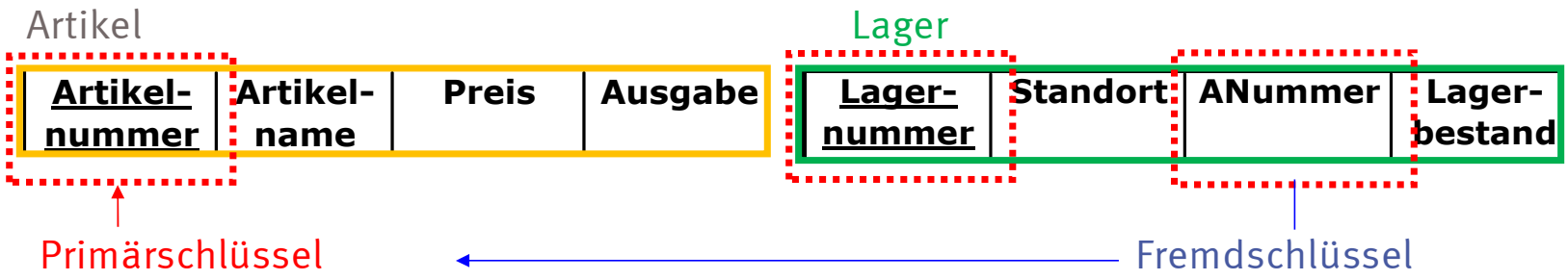
```
CREATE SEQUENCE Kun_seq
        INCREMENT BY 2
        START WITH  1
        NOMAXVALUE
        NOCYCLE
        CACHE 10;
```

Verwendung

```
INSERT
INTO Kunde(Kundennummer, Nachname)
VALUES (Kun_seq.NEXTVAL, 'Meitner')
```

NEXTVAL: nächste verfügbare Nummer
CURRVAL: aktuell verwendete Nummer

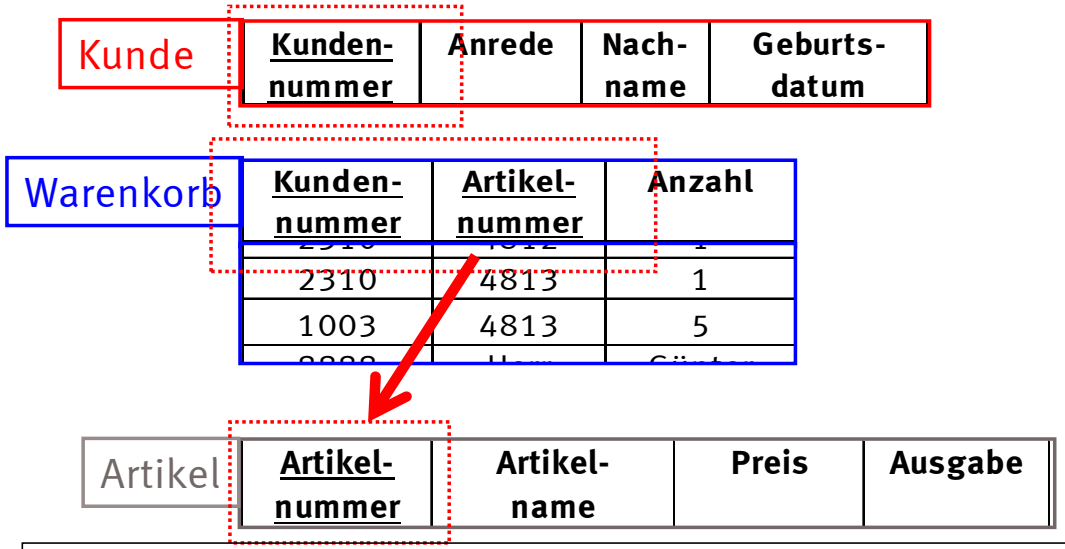
Fremdschlüssel definieren




```
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       (Kundennummer, Artikelnummer),
FOREIGN KEY       (Artikelnummer)
                  REFERENCES Artikel (Artikelnummer),
FOREIGN KEY       (Kundennummer)
                  REFERENCES Kunde (Kundennummer))
```

Delete-Option 1

```
DELETE
FROM Artikel
WHERE Artikelnummer=4812
```

Artikel

Artikel-nummer	Artikel-name	Preis	Ausgabe
4812	Datenbanke	29,95 €	gebunden
~~4813~~	~~Märchen vo~~	~~12,90 €~~	~~broschiert~~

Lager

Lager-nummer	Standort	ANummer	Lager-bestand
27135	R235	4812	18
27423	R371	4813	0

???

Inkonsistenter Zustand

Gewünschte Konsistenzsicherung durch das DBMS:

```
SQL-Fehler: ORA-02292: integrity constraint (SAATZBH.SYS_C009165) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

Delete-Option 2

Artikel

Artikel-nummer	Artikel-name	Preis	Ausgabe
4812	Basiswissen	29,95 €	gebunden
~~4813~~	~~Das Ende der~~	~~12,90 €~~	~~broschiert~~

Lager

Lager-nummer	Standort	ANummer	Lager-bestand
27135	R235	~~4812~~	18
27423	R371	4813	0

???

Lösch-Option:

[**ON DELETE**
{**CASCADE** | **SET NULL** | **RESTRICT**}]
SET DEFAULT

Datensatz löschen

Referenzwert setzen

Änderung verbieten (Standard)

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

Was bewirkt diese Klausel?

Delete-Option

Artikel

Artikel-nummer	Artikel-name	Preis	Ausgabe
4812	Datenbanke	29,95 €	gebunden
~~4813~~	~~Märchen vo~~	~~12,90 €~~	~~broschiert~~

Lager

Lager-nummer	Standort	ANummer	Lager-bestand
27135	R235	4812	18
27423	R371	~~4813~~	0

On DELETE SET NULL → **NULL**

Was bedeutet diese Deklaration?

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

Vorsicht Falle:
Was passiert, wenn der Lagerbestand › 0 ist?

Dynamische Integritätsbedingung:
„Ein Artikel darf nur gelöscht werden, wenn kein Lagerbestand vorhanden ist.“
Umsetzung:
→ Implementierung eines DB-Prüfprogramms (Trigger) ist erforderlich

Eigenschaften von Constraints (Oracle)

Mit dem Check-Constraint sind unter Oracle eingeschränkte Suchbedingungen formulierbar, wie der Vergleiche mit Konstanten oder anderen Attributen der zugehörigen Tabelle.

- Beispiel:

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

- Eigenschaften:
  - Alle Constraints haben einen Namen
  - Constraints können gelöscht, deaktiviert und reaktiviert werden

```
ALTER TABLE Kunde DISABLE  CONSTRAINT pruefeAnrede;
ALTER TABLE Kunde ENABLE   CONSTRAINT pruefeAnrede;
ALTER TABLE Kunde DROP     CONSTRAINT pruefeAnrede;
```

  - Abweichungen vom Standard-SQL bei Oracle:
    - Keine SELECT-Anfragen auf andere Tabellen
    - Keine Nutzung der Systemvariablen SYSDATE im CHECK-Constraint

Definition von Integritätsbedingungen (Oracle)

Attribut-Constraint

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

Bezug auf ein Attribut herstellen

Tabellen-Constraint

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

Struktur erweiterndes Constraint

```
CREATE TABLE Kunde (
...
Anrede CHARACTER(4),
...
);

ALTER TABLE Kunde
ADD CONSTRAINT pruefeAnrede
     CHECK (
              Anrede
                 IN ('Herr', 'Frau'))
);
```

Reguläre Ausdrücke

Reguläre Ausdrücke können in der Check-Klausel verwendet werden.
Der Beginn des regulären Ausdrucks wird durch das Zeichen **^** gekennzeichnet.
Der reguläre Ausdruck wird mit dem Zeichen $ abgeschlossen.

- Beispiele
  - Das Passwort des Kunden besteht nur aus Zahlen:

```
CHECK (REGEXP_LIKE(Passwort, '^[[:digit:]]+$'))
```

  - Das Passwort des Kunden besteht nur aus Buchstaben:

```
CHECK (REGEXP_LIKE(Passwort, '^[A-Za-z]+$'))
```

Bei Oracle ist das aktuelle Datum (SYSDATE) in der CHECK-Klausel nicht verwendbar. Weshalb?

Inhaltsübersicht

Organisation

- Diese Woche
  - Inhalte
    - Lernmodul DDL und DML
    - Wochentest
    - Praktikum
    - Projektaufgabe

Vielen Dank
für Ihre Aufmerksamkeit