

Einführung in SQL

- ▶ **CREATE TABLE**
Erzeugen von Relationen
- ▶ **ALTER TABLE**
Ändern von Relationen
- ▶ **DROP TABLE**
Löschen von Relationen

CREATE TABLE: Übersicht

SQL
Statement

Name der
Tabelle

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL,  
    geschlecht CHAR(1),  
    plz       CHAR(5),  
    ort       CHAR(30)   NOT NULL,  
    arb_zeit  CHAR(1)  
);
```

Attribut

SQL-Datentyp

Nullwert unzulässig
(Standard: Null zulässig)

CREATE TABLE: Festlegung des Primärschlüssels

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL,  
    ...,  
    PRIMARY KEY (p_m_nr)  
);
```



Festlegung des
Primärschlüssels

CREATE TABLE: Default-Werte

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL    DEFAULT 'Mustermann',  
    ...,  
    PRIMARY KEY (p_m_nr)  
);
```

Wird ein Datensatz erzeugt,
ohne einen Wert für das
Attribut *nname* erhält es den
Default-Wert

CREATE TABLE: Default-Werte

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL,  
    ...,  
    PRIMARY KEY (p_m_nr)  
);
```

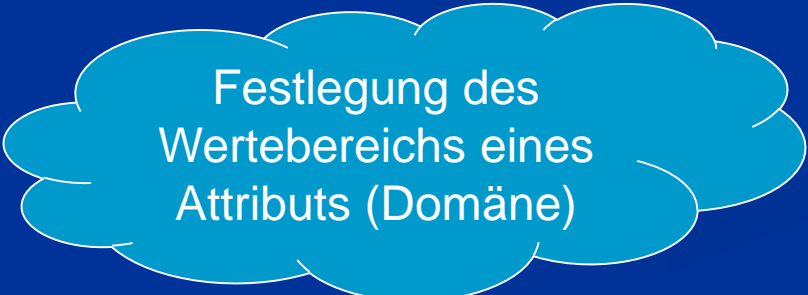
Anmerkung: Es besteht die Möglichkeit, einen Attribut einen Defaultwert zu geben, nachdem man die Tabelle angelegt hat mit ALTER TABLE

```
ALTER TABLE T_Mitarbeiter  
ALTER nname SET DEFAULT 'Mustermann';
```

Wird ein Datensatz erzeugt, ohne einen Wert für das Attribut *nname* erhält es den Default-Wert

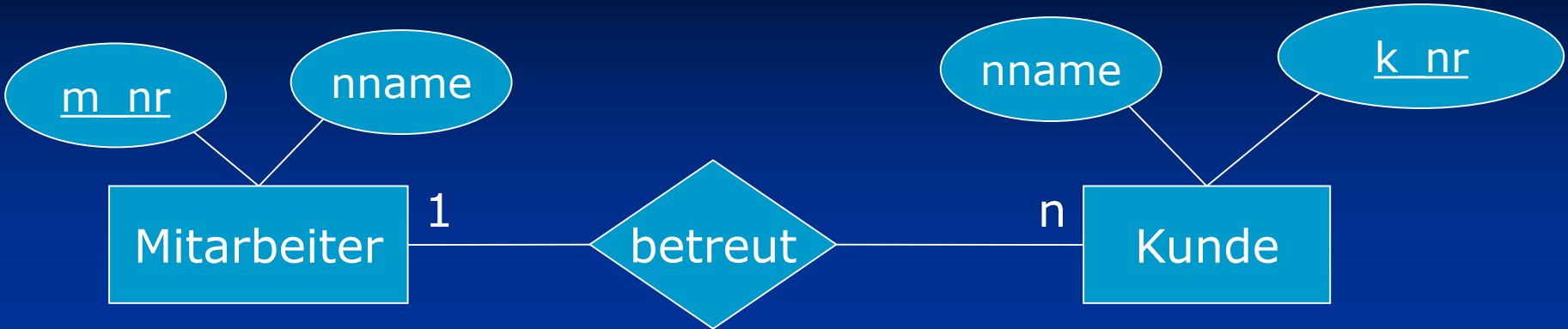
CREATE TABLE: Einschränkung des Wertebereichs mit CHECK

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL,  
    geschlecht CHAR(1),  
    CHECK ( geschlecht in ( 'M' , 'W' ) ),  
    ...  
);
```



Festlegung des
Wertebereichs eines
Attributs (Domäne)

CREATE TABLE: Fremdschlüssel und referenzielle Integrität



T_Mitarbeiter	
<u>p_m_nr</u>	nname
27	Meier
11	Müller

T_Kunden		
<u>p_k_nr</u>	nname	f_m_nr
12	Schmidt	27
10	Werner	14 ?
14	Beyer	11
27	Andreotti	27

Die referenzielle Integrität soll die Gültigkeit von Fremdschlüsselwerten sicherstellen.

CREATE TABLE: Fremdschlüssel und referenzielle Integrität

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr      INTEGER  
    nname       CHAR(30)  
    ...,  
    PRIMARY KEY (p_m_nr)  
);
```

Mitarbeiter-Datensätze, deren Primärschlüsselwert in der Tabelle T_Kunde als Fremdschlüssel steht, können weder geändert (nur Schlüsselwert), noch gelöscht werden!

```
CREATE TABLE T_Kunden  
(  
    p_k_nr      INTEGER      NOT NULL,  
    f_m_nr      INTEGER      NOT NULL,  
    nname       CHAR(30)     NOT NULL,  
    ...  
    PRIMARY KEY (p_k_nr),  
    FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (p_m_nr)  
);
```


FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter
ON UPDATE ... ON DELETE ...



Was passiert mit dem
Fremdschlüsselwert, wenn
sich der
Primärschlüsselwert
ändert?



Was passiert mit dem
Fremdschlüsselwert, wenn
der Datensatz mit dem
Primärschlüsselwert
gelöscht wird?

FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter
ON UPDATE ... ON DELETE ...

- **NO ACTION**

Änderung von referenzierten Primärschlüsseln nicht möglich (default)

- **CASCADE**

Lösch- bzw. Änderungsweitergabe

- **SET NULL**

Fremdschlüsselwert wird auf Null gesetzt

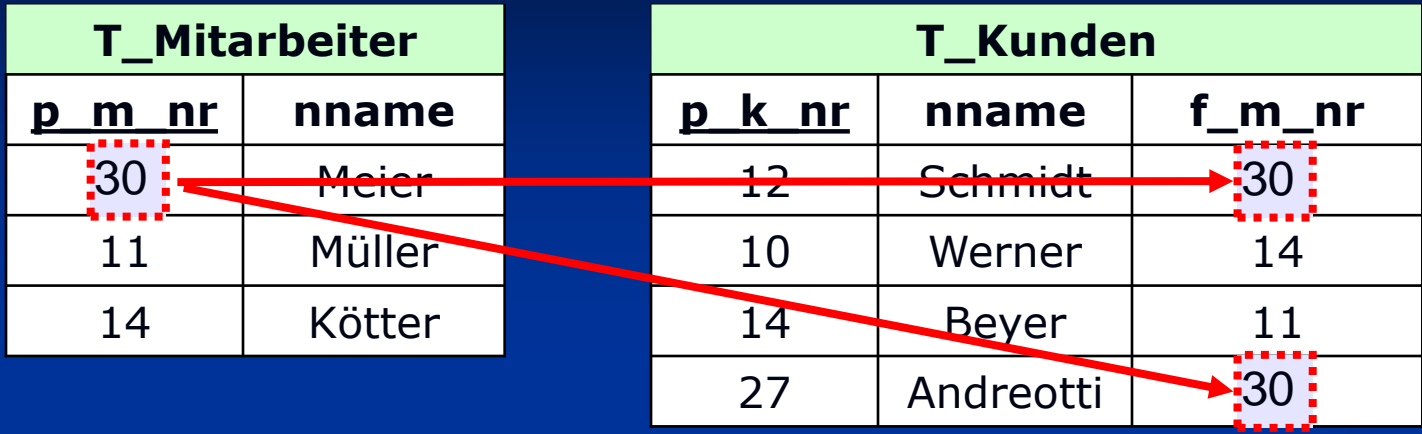
- **RESTRICT**

Änderung von referenzierten Primärschlüsseln nicht möglich. NO ACTION und RESTRICT sind dasselbe.

- **SET DEFAULT (nicht bei MySQL)**

Fremdschlüsselwert wird auf Defaultwert gesetzt

Beispiele:



FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (m_nr)
ON UPDATE CASCADE ON DELETE CASCADE

CREATE TABLE: Fremdschlüssel und referenzielle Integrität

Beispiele:

T_Mitarbeiter			T_Kunden		
<u>p_m_nr</u>	nname		<u>p_k_nr</u>	nname	f_m_nr
27	Meier	→	12	Schmidt	27
11	Müller		10	Werner	14
14	Kötter		14	Beyer	11
		↘	27	Andreotti	27

FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (m_nr)
ON UPDATE CASCADE ON **DELETE CASCADE**

**Problematik: Ist die Löschung der Kunden
erwünscht bzw. sinnvoll?**

CREATE TABLE: Fremdschlüssel und referenzielle Integrität

Beispiele:

T_Mitarbeiter	
<u>p_m_nr</u>	nname
27	Meier
11	Müller
14	Kötter

T_Kunden		
<u>p_k_nr</u>	nname	f_m_nr
12	Schmidt	NULL
10	Werner	14
14	Beyer	11
27	Andreotti	NULL

FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (m_nr)
ON UPDATE CASCADE ON DELETE SET NULL

CREATE TABLE: Fremdschlüssel und referenzielle Integrität

Beispiele:

T_Mitarbeiter		T_Kunden		
<u>p_m_nr</u>	nname	<u>p_k_nr</u>	nname	f_m_nr
30	Meier	12	Schmidt	NULL
11	Müller	10	Werner	14
14	Kötter	14	Beyer	11
		27	Andreotti	NULL

FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (m_nr)
ON UPDATE SET NULL ON DELETE CASCADE

*Ist das Setzen des Fremdschlüssels
auf NULL sinnvoll?*

CREATE TABLE: Verwendung zusammengesetzter Schlüssel (split keys)

Beispiel: Verwendung eines zusammengesetzten Schlüssels (split key)

```
CREATE TABLE T_ABC
```

```
(  
    p_a_nr          INTEGER          NOT NULL,  
    p_b_nr          INTEGER          NOT NULL,  
    ...  
    PRIMARY KEY (p_a_nr, p_b_nr)  
);
```

```
CREATE TABLE T_XYZ
```

```
(  
    x_nr            INTEGER          NOT NULL,  
    f_a_nr          INTEGER          NOT NULL,  
    f_b_nr          INTEGER          NOT NULL,  
    ...  
    PRIMARY KEY (x_nr),  
    FOREIGN KEY (f_a_nr, f_b_nr) REFERENCES T_ABC (p_a_nr, p_b_nr)  
);
```

CREATE TABLE: Verwendung zusammengesetzter Schlüssel (split keys)

Beispiel: Verwendung eines zusammengesetzten Schlüssels bei einer Zwischentabelle (split key)

```
CREATE TABLE T_Projekte
```

```
(  
    projekt_nr      INTEGER          NOT NULL,  
    projektname     VARCHAR(10)     NOT NULL,  
    PRIMARY KEY (projekt_nr)  
);
```

```
CREATE TABLE T_Mitarbeiter
```

```
(  
    p_mitarbeiter_nr  INTEGER        NOT NULL,  
    name              VARCHAR(100)   NOT NULL,  
    PRIMARY KEY (p_mitarbeiter_nr)  
);
```

```
CREATE TABLE T_Projekte_Mitarbeiter
```

```
(  
    pf_mitarbeiter_nr INTEGER NOT NULL,  
    pf_projekt_nr     INTEGER NOT NULL,  
    taetigseit        DATE,  
    PRIMARY KEY(pf_mitarbeiter_nr, pf_projekt_nr)  
);
```

```
ALTER TABLE T_Projekte_Mitarbeiter ADD CONSTRAINT FK1_T_Projekte_Mitarbeiter  
FOREIGN KEY (pf_mitarbeiter_nr) REFERENCES T_Mitarbeiter (p_mitarbeiter_nr) ON  
UPDATE CASCADE ON DELETE CASCADE;
```

```
ALTER TABLE T_Projekte_Mitarbeiter ADD CONSTRAINT FK2_T_Projekte_Mitarbeiter  
FOREIGN KEY (pf_projekt_nr) REFERENCES T_Projekte (projekt_nr) ON UPDATE CASCADE  
ON DELETE CASCADE;
```


CREATE TABLE: Bezeichner für CONSTRAINTS (Beschränkungen)

Beispiele:

```
CREATE TABLE T_Kunden
```

```
(
```

```
    p_k_nr          INTEGER          NOT NULL,
```

```
    CONSTRAINT chk_Werte_p_k_nr CHECK (p_k_nr > 0 AND p_k_nr < 1000),
```

```
    f_m_nr          INTEGER          NOT NULL,
```

```
    nname           CHAR(30)         NOT NULL,
```

```
    vname           CHAR(30)         NOT NULL,
```

```
    CONSTRAINT pk_T_Kunden PRIMARY KEY (p_k_nr),
```

```
    CONSTRAINT fk_T_Kunden FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (p_m_nr)
```

```
);
```

Das Benennen von Beschränkungen ermöglicht...

- die Ausgabe aussagekräftiger Fehlermeldungen
- das nachträgliche Ändern der Beschränkungen über ALTER TABLE

ALTER TABLE

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL  
);
```

```
CREATE TABLE T_Kunden  
(  
    p_k_nr    INTEGER    NOT NULL,  
    f_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL  
);
```

Primärschlüssel hinzufügen

Spalte *vname* hinzufügen

```
ALTER TABLE T_Mitarbeiter ADD PRIMARY KEY (p_m_nr);
```

```
ALTER TABLE T_Mitarbeiter ADD vname CHAR(30);
```

```
ALTER TABLE T_Kunden ADD PRIMARY KEY (p_k_nr);
```

```
ALTER TABLE T_Kunden ADD FOREIGN KEY (f_m_nr) REFERENCES T_Mitarbeiter (p_m_nr);
```

Fremdschlüssel hinzufügen
(Reihenfolge CREATE TABLE flexibel)

DROP TABLE

```
CREATE TABLE T_Mitarbeiter  
(  
    p_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL  
);
```

```
CREATE TABLE T_Kunden  
(  
    p_k_nr    INTEGER    NOT NULL,  
    f_m_nr    INTEGER    NOT NULL,  
    nname     CHAR(30)   NOT NULL  
);
```

DROP TABLE T_Kunden;

DROP TABLE T_Mitarbeiter;

Tabelle T_Kunden löschen
(Reihenfolge beachten!)

Beispiel für MySQL-DB

```
CREATE TABLE T_Mitarbeiter  
(  
  p_m_nr INTEGER NOT NULL,  
  nname CHAR(30) NOT NULL  
) ENGINE = InnoDB;
```

```
CREATE TABLE T_Kunden  
(  
  p_k_nr INTEGER NOT NULL,  
  f_m_nr INTEGER NULL,  
  nname CHAR(30) NOT NULL  
) ENGINE = InnoDB;
```

/*primary keys einfuegen*/

```
ALTER TABLE T_Mitarbeiter ADD CONSTRAINT pk_T_Mitarbeiter PRIMARY KEY (p_m_nr);  
ALTER TABLE T_Kunden ADD CONSTRAINT pk_T_Kunden PRIMARY KEY (p_k_nr);
```

/*foreign keys einfuegen*/

```
ALTER TABLE T_Kunden ADD CONSTRAINT fk_T_Kunden FOREIGN KEY (f_m_nr)  
REFERENCES T_Mitarbeiter (p_m_nr) ON UPDATE CASCADE ON DELETE SET NULL;
```

/*Weitere Beschraenkungen*/

```
ALTER TABLE T_Kunden ADD CONSTRAINT chk_Werte_k_nr CHECK (p_k_nr > 0 AND p_k_nr < 1000);
```

Nur die Storage-Engine *InnoDB* unterstützt
Transaktionen und referenzielle Integrität über
Fremdschlüssel