


# Wiederholung

## SQL: CREATE

Bezeichner: „Name“  
Typ: varchar(255) (String)



Name	Vorname	Land	Geburtsjahr

```
create table PERSONEN (Name varchar(255), Vorname varchar(255), Land varchar(255), Geburtsjahr int(4) )
```

# Wiederholung

## SQL: Insert & Update

Name	Vorname	Land	Geburtsjahr
Gurion	Ben	Israel	1888
Curie	Marie	Polen	1867



Eingabefehler

insert into PERSONEN (Name, Vorname, Land, Geburtsjahr) values ('Gurion', 'Ben', 'Israel', '1888')

insert into PERSONEN (Name, Vorname, Land, Geburtsjahr) values ('Curie', 'Marie', 'Polen', '1867')

Name	Vorname	Land	Geburtsjahr
Gurion	Ben	Israel	1886
Curie	Marie	Polen	1867



Korrektur

update PERSONEN set Geburtsjahr = '1886' where Name = 'Gurion'

# Wiederholung

## SQL: Delete

Name	Vorname	Land	Geburtsjahr
Gurion	Ben	Israel	1888

```
delete from PERSONEN where Name = 'Curie'
```

# Wiederholung

## SQL: Select

Name	Vorname	Land	Geburtsjahr
Gurion	Ben	Israel	1886
Curie	Marie	Polen	1867
Gagarin	Juri	UdSSR	1934

```
select Name, Land from PERSONEN where Geburtsjahr < 1900
```

### Ergebnis:

Gurion Israel  
Curie Polen

### Hinweise:

- Hinter select kann man auch \* schreiben, dann bekommt man alle Spalten angezeigt.
- In Where-Klauseln kann man auch ein '%' als Wildcard nutzen:  
select ... where Name like 'G%'

# Heute

Primär- und Fremdschlüssel in SQL

→ Verwendung

→ Referenzielle Integrität

Atomarität in SQL

# Schlüssel

## Kleine Wiederholung: Schlüssel ohne DBS

	A	B	C
1	s5000123	Jane	
2	s5000222	Joe	
3	s5000223	Tim	
4			

studends.csv

	A	B	C
1	1	DBS	
2	2	PROGRAMMING	
3	3	GRAPHICS	
4			

lectures.csv

	A	B	C
1	s5000123	3	
2	s5000123	2	
3	s5000223	1	
4	s5000222	3	
5	s5000222	2	
6	s5000222	1	
7			

attendance.csv

csv: Textdatei mit Komma als Trenner:

s5000123, Jane  
s5000222, Joe  
s5000223, Tim

# Schlüssel

## Kleine Wiederholung: Schlüssel ohne DBS

	A	B	C
1	s5000123	Jane	
2	s5000222	Joe	
3	s5000223	Tim	
4			

studends.csv

	A	B	C
1	1	DBS	
2	2	PROGRAMMING	
3	3	GRAPHICS	
4			

lectures.csv

	A	B	C
1	s5000123	3	
2	s5000123	2	
3	s5000223	1	
4	s5000222	3	
5	s5000222	2	
6	s5000222	1	
7			

attendance.csv

Frage: Welche Studierenden (Namen)  
besuchen welche Kurse?

# Schlüssel

## Kleine Wiederholung: Schlüssel ohne DBS

```
file1 = open('students.csv', 'r')
Lines = file1.readlines()
s=[]
for line in Lines:
    line = line.strip()
    s.append( line.split(",") )

file2 = open('lectures.csv', 'r')
Lines = file2.readlines()
l=[]
...

file3 = open('attendance.csv', 'r')
Lines = file3.readlines()
a=[]
...

for aa in a:
    for ss in s:
        if ss[0]==aa[0]:
            for ll in l:
                if aa[0]==ss[0] and aa[1]==ll[0]:
                    print("Student: {} \t Lectures: {}".format(ss[1],ll[1]) )
```

select.py



# Schlüssel

## Kleine Wiederholung: Schlüssel ohne DBS

select.py

```
file1 = open('students.csv', 'r')
Lines = file1.readlines()
s=[]
for line in Lines:
    line = line.strip()
    s.append( line.split(",") )

file2 = open('lectures.csv', 'r')
Lines = file2.readlines()
l=[]
...

file3 = open('attendance.csv', 'r')
Lines = file3.readlines()
a=[]
...

for aa in a:
    for ss in s:
        if ss[0]==aa[0]:
            for ll in l:
                if aa[0]==ss[0] and aa[1]==ll[0]:
                    print("Student: {} \t Lectures: {}".format(ss[1],ll[1]) )
```

Ausgabe

```
Student: Jane  Lectures: GRAPHICS
Student: Jane  Lectures: PROGRAMMING
Student: Tim   Lectures: DBS
Student: Joe   Lectures: GRAPHICS
Student: Joe   Lectures: PROGRAMMING
Student: Joe   Lectures: DBS
```

Funktioniert, aber  
keinerlei Fehlerbehandlung

# Schlüssel: Primärschlüssel

Personen

ID: Primärschlüssel

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Ein Primärschlüssel kennzeichnet eine Zeile eindeutig.  
Er darf also immer nur einmal existieren.

# Schlüssel: Primärschlüssel

Personen

ID: Primärschlüssel

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

```
create table Personen (  
  ID int(10) PRIMARY KEY,  
  Nachname varchar(255),...  
)
```

# Schlüssel: Fremdschlüssel

Personen

ID: Primärschlüssel

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Amtszeiten

ID	gewählt	PersonID
10	1948	1
20	1954	2

PersonID:  
Fremdschlüssel

# Schlüssel: Fremdschlüssel

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Amtszeiten

ID	gewählt	PersonID
10	1948	1
20	1954	2

Ein Fremdschlüssel verweist auf einen Primärschlüssel **einer anderen Tabelle**, der existieren muss.

# Schlüssel: Fremdschlüssel

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Amtszeiten

ID	gewählt	PersonID
10	1948	1
20	1954	2

```
create table Amtzeiten (  
    ID int(10),  
    gewählt int(10),  
    PersonID int(10),  
    FOREIGN KEY(PersonID) references Personen(ID)  
)
```

# Schlüssel:

## Tabellen verbinden

ID	Nachname	Vorname	Land	Geburts jahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

select Nachname, Land, gewählt from  
personen , amtszeiten  
where ID = PersonID

SQL

ID	gewählt	PersonID
10	1948	1
20	1954	2

Nachname	Land	gewählt
Gurion	Israel	1948
Nasser	Ägypten	1954

Ergebnis

# Schlüssel

## Referentielle Integrität

**Personen**

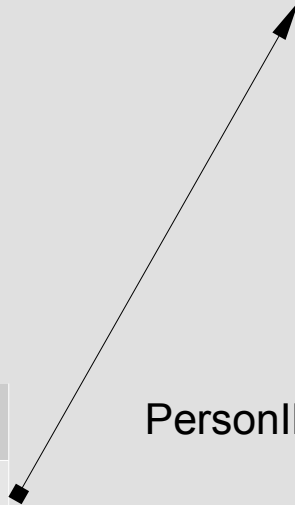
ID: Primärschlüssel

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

**Amtszeiten**

gewählt	PersID
1948	1
1954	2

PersonID: Fremdschlüssel





# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Amtszeiten

gewählt	PersID
1948	1
1954	2

```
create table Amtzeiten (  
    ID int(10),  
    gewählt int(4),  
    PersID int(10),  
    foreign key(PersID) references Personen(ID)  
);
```

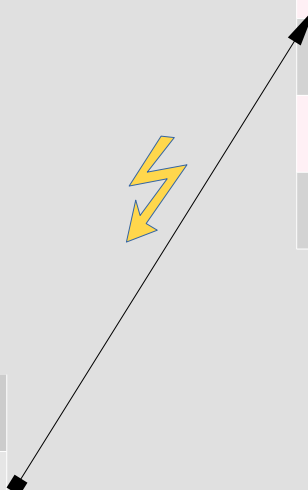
# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Amtszeiten

gewählt	PersID
1948	1
1954	2



delete from Personen where ID=1

# Referentielle Integrität

Amtszeiten

gewählt	PersID
1948	1
1954	2



Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

delete from Personen where ID=1

Object Browser

SCHEMAS

Search objects

db\_e5000160

- Tables
  - amtszeit
  - personen
- Views
- Routines

Query 1

delete from personen where ID=1

SQL Additions

My Snippets

Snippets

Output

Action Output

	Time	Action	Message	Duration / Fetch
1	07:46:53	delete from personen where ID=1	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('db_e5000160`.`amtszeit`, CONSTRAINT `...`	0.140 sec

# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

Amtszeiten

gewählt	PersID
1948	1
1954	2

```
create table Amtzeiten (  
    ID int(10),  
    gewählt int(4),  
    PersID int(10),  
    foreign key(PersID) references Personen(ID) ON DELETE ...  
);
```

# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
4	Gurion	Ben	Israel	1886
2	Nasser	Gamal Abdel	Ägypten	1918

Amtszeiten

gewählt	PersID
1948	1
1954	2

```
create table Amtzeiten (  
    ...  
    foreign key(PersID) references Personen(ID) on delete  
        NO ACTION oder  
        CASCADE oder  
        SET NULL oder  
        SET DEFAULT  
)
```

# Referentielle Integrität

Person

ID	Nachname	Vorname	Land	Geburtsjahr
4	Gurion	Ben	Israel	1886
2	Nasser	Gamal Abdel	Ägypten	1918

Amtszeiten

gewählt	PersID
1948	1
1954	2

„Cascade“ heißt:

Mache mit dem referenzierenden Datensatz  
das gleiche wie mit den referenzierten Datensatz.

Hier: Wenn ich einen referenzierten Datensatz in der  
Tabelle Personen lösche (Gurion), dann entferne  
auch den referenzierenden Datensatz aus Amtszeiten  
(den mit PersID=1)

# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
1	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

ok

Amtszeiten

gewählt	PersID
1948	1
1954	2

# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
10	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934



Amtszeiten

gewählt	PersID
1948	1
1954	2

update Personen set ID=10 where ID=1



# Referentielle Integrität

Amtszeiten

gewählt	PersID
1948	1
1954	2



Personen

ID	Nachname	Vorname	Land	Geburtsjahr
10	Gurion	Ben	Israel	1886
2	Nasser	Gamal, Abdel	Ägypten	1918
3	Curie	Marie	Polen	1867
4	Gagarin	Juri	UdSSR	1934

update Personen set ID=10 where ID=1

The screenshot shows a SQL IDE interface with the following components:

- Object Browser:** Displays the database structure for 'db\_e5000160', including tables 'amtszeit' and 'personen'.
- Query 1:** Contains the SQL statement: `update personen set ID=10 where ID=1`.
- Output:** Shows the execution results, including an error message: `Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('db_e5000160', 'amtszeit', CONSTRAINT '...' 0.156 sec`.
- SQL Additions:** Shows a snippet titled 'My Snippets'.

# Referentielle Integrität

Personen

ID	Nachname	Vorname	Land	Geburtsjahr
10	Gurion	Ben	Israel	1886
2	Nasser	Gamal Abdel	Ägypten	1918

Amtszeiten

gewählt	PersID
1948	1
1954	2

```
create table Amtszeiten (  
    ...  
    foreign key(PersID) references Personen(ID) on update  
        NO ACTION oder  
        CASCADE oder  
        SET NULL oder  
        SET DEFAULT  
)
```

# Datenbankzugriffssprachen

## **Atomarität**

Transaktionen müssen atomar („binär“) ausgeführt werden.  
Entweder komplett richtig oder gar nicht.

# Atomarität

a\_name

EAN	Name
4.013162.000012	Limo
4.013162.000013	Cola

a\_menge

EAN	Menge
4.013162.000012	23
4.013162.000013	42

a\_preis

EAN	Value
4.013162.000012	0,99
4.013162.000013	0,99

# Atomarität

a\_name

EAN	Name
4.013162.000012	Limo
4.013162.000013	Cola
4.013162.000014	Soft

a\_menge

EAN	Menge
4.013162.000012	23
4.013162.000013	42
4.013162.000014	100

a\_preis

EAN	Value
4.013162.000012	0,99
4.013162.000013	0,99
4.013162.000014	0,49

# Atomarität

a\_name

EAN	Name
4.013162.000012	Limo
4.013162.000013	Cola
4.013162.000014	Soft

a\_menge

EAN	Menge
4.013162.000012	23
4.013162.000013	42
4.013162.000014	100

a\_preis

EAN	Value
4.013162.000012	0,99
4.013162.000013	0,99
4.013162.000014	0,49

insert into a\_name (EAN, Name) values ('4.013162.000014', 'Soft');

insert into a\_menge (EAN, Menge) values ('4.013162.000014', '100');

insert into a\_preis (EAN, Value) values ('4.013162.000014', '0,49');

# Atomarität

a\_name

EAN	Name
4.013162.000012	Limo
4.013162.000013	Cola
4.013162.000014	Saft

a\_menge

EAN	Menge
4.013162.000012	23
4.013162.000013	42
4.013162.000014	100

a\_preis

EAN	Value
4.013162.000012	0,99
4.013162.000013	0,99



insert into a\_name (EAN, Name) values ('4.013162.000014', 'Saft');

insert into a\_menge (EAN, Menge) values ('4.013162.000014', '100');

~~insert into a\_preis (EAN, Value) values ('4.013162.000014', '0,49');~~

Fehler in letzter SQL-Anweisung. Die Datenbank wäre „kaputt“, weil wir nun Saft ohne Preis hätten.

# Atomarität

a\_name

EAN	Name
4.013162.000012	Limo
4.013162.000013	Cola

a\_menge

EAN	Menge
4.013162.000012	23
4.013162.000013	42

a\_preis

EAN	Value
4.013162.000012	0,99
4.013162.000013	0,99

Daher: **Rollback**

Die Datenbank sieht nun wieder aus wie vorher.



# Atomarität

## Implizites Rollback

**START TRANSACTION;**

```
insert into a_name (EAN, Name) values ('4.013162.000014', 'Saft');
```

```
insert into a_menge (EAN, Menge) values ('4.013162.000014', '100');
```

```
insert into a_preis (EAN, Value) values ('4.013162.000014', '0,49');
```

**COMMIT;**

# Atomarität

## Beispiel ohne Transaktionsbehandlung

	EAN	Name
▶	4.013162.000012	Limo
	4.013162.000013	Cola
	4.013162.000014	Saft

	EAN	Menge
▶	4.013162.000012	23
	4.013162.000013	42
	4.013162.000014	100

	EAN	value
▶	4.013162.000012	1
	4.013162.000013	1

# Atomarität

## Beispiel mit Transaktionsbehandlung

	EAN	Name
▶	4.013162.000012	Limo
	4.013162.000013	Cola

	EAN	Menge
▶	4.013162.000012	23
	4.013162.000013	42

	EAN	value
▶	4.013162.000012	1
	4.013162.000013	1

SET AUTOCOMMIT=OFF;

← Client-Spezifisch

START TRANSACTION;

insert into a\_name (EAN, Name) values ('4.013162.000014', 'Saft');

insert into a\_menge (EAN, Menge) values ('4.013162.000014', '100');

insert into a\_preis (EAN, Value) values ('4.013162.000014', 'xxx');

COMMIT;

ROLLBACK;

← Client-Spezifisch(?)

# Atomarität

## Commit / Rollback in der Praxis

```
$DriverString = "DBI:mysql:db_e5000160:dbserv.ba-nitsch.de:3306";  
$dbh = DBI->connect($DriverString, 'e5000160', 'xxxx', {RaiseError => 0}) or die "Error Connect";  
$dbh->{AutoCommit} = 0; # enable transactions  
$dbh->{RaiseError} = 1; # die( ) if a query has problems
```

```
eval {  
    $Query = $dbh->prepare("insert into a_name (EAN, Name) values ('4.013162.000014', 'Saft')");  
    $Query->execute; $Query->finish();  
  
    $Query = $dbh->prepare("insert into a_menge (EAN, Menge) values ('4.013162.000014', '100')");  
    $Query->execute; $Query->finish();  
  
    $Query = $dbh->prepare("insert into a_preis (EAN, Value) values ('4.013162.000014', 'xxx')");  
    $Query->execute; $Query->finish();  
};  
  
if ( $@ ) {  
    print("Error: $@\n"); $dbh->rollback();  
}  
else {  
    print("OK:\n"); $dbh->commit();  
}  
  
$dbh->disconnect();
```

# Atomarität: Explizites Rollback

## Pseudocode

**START TRANSACTION;**

```
try {  
    insert into a_name (EAN, Name) values ('4.013162.000014', 'Saft');  
}  
onError: rollback();
```

```
try {  
    insert into a_menge (EAN, Menge) values ('4.013162.000014', '100');  
}  
onError rollback();
```

```
try {  
    insert into a_preis (EAN, Value) values ('4.013162.000014', '0,49');  
}  
onError: rollback();
```

**COMMIT;**