

# DLBDSPBDM01\_D – PROJEKT: DATA-MART-ERSTELLUNG IN SQL

Portfolio

The background of the slide features a stack of several books, with their pages and spines visible. Overlaid on this image is a semi-transparent layer containing CSS code snippets, such as 'display: block; position: absolute', 'left: -5px', 'left: -6px', 'display: inline-block', 'display: block', 'text-align: center', 'text-decoration: underline', and 'text-align: right'.

# BUCHTAUSCH-APP

## EN ANWENDUNGSFALL FÜR BUCHVERMIETUNG



# INHALTSVERZEICHNIS

[PROJEKTZIEL](#)

[TOOLS](#)

[DATENBANKERSTELLUNG](#)

[ANFORDERUNGSANALYSE](#)

[ER MODEL](#)

[IMPLEMENTIERUNGSVERFAHREN](#)

[ADMINS](#)

[USERS](#)

[REGISTRATIONS](#)

[LANGUAGES](#)

[AUTHORS](#)

[PUBLISHINGHOUSES](#)

[ADDRESSES](#)

[ADDRESSES CONSTRAINTS](#)

[CATEGORIES](#)

[SHIPPINGOPTIONS](#)

[BOOKS](#)

[LOANS](#)

[COMMENTS](#)

[VALUATIONS](#)

[RESERVATIONS](#)

[SEARCHES](#)

[TESTFÄLLE](#)

[VALIDIERUNGSREGELN](#)

[WEITERE TESTFÄLLE](#)

[ZUWEISUNG VON TRIGGERN](#)

[KORREKTUR DER TRIGGER-FUNKTION](#)

[VALIDATIONSTESTS](#)

[OPTIMIERUNG FÜR HÄUFIGE ABFRAGEN](#)

[INDEXIERUNG UND MATERIALIZES VIEW](#)

[KOMPLEXE ABFRAGE ÜBER MEHR TABELLEN](#)

[METADATEN](#)





## PROJEKTZIEL

Entwicklung einer funktionalen Datenbank für eine Büchertausch-App, die von den Studierenden entwickelt werden soll

App richtet sich an Nachbarschaft, die Bücher ausleihen und anzubieten möchten



# TOOLS

## DRAW.IO

- Open Source
- Benutzerfreundlich
- Ermöglicht einfache Visualisierung des Datenmodells
- Bietet diverse Vorlagen
- Ermöglicht Echtzeit-Zusammenarbeit mit mehreren Benutzern
- Vielfältige Exportoptionen von Diagrammen
- Integration mit GitHub, Microsoft OneDrive

## POSTGRESQL

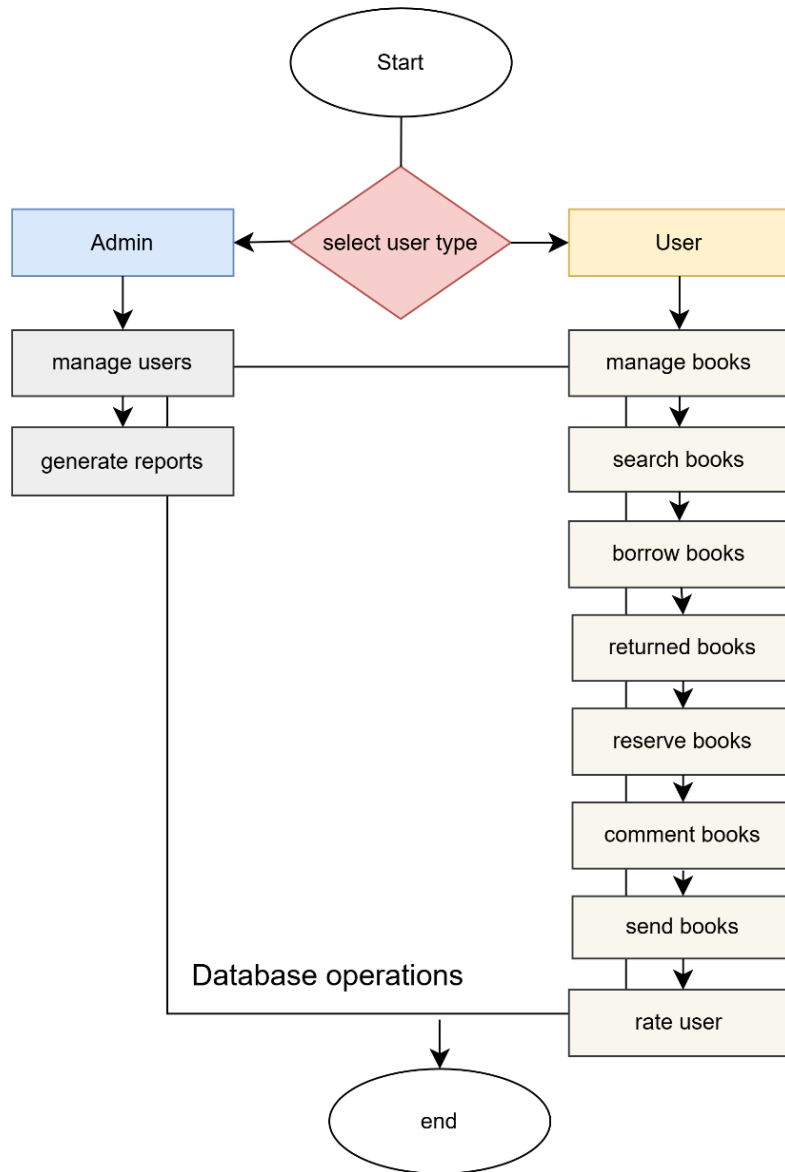
- Open Source
- Quelloffen
- Leistungsstark
- Skalierbar
- Gewährleistet Datenintegrität
- Unterstützt komplexe Abfragen
- Gut für wachsende Anwendungen geeignet
- Bietet umfassende Dokumentation

# DATENBANKERSTELLUNG

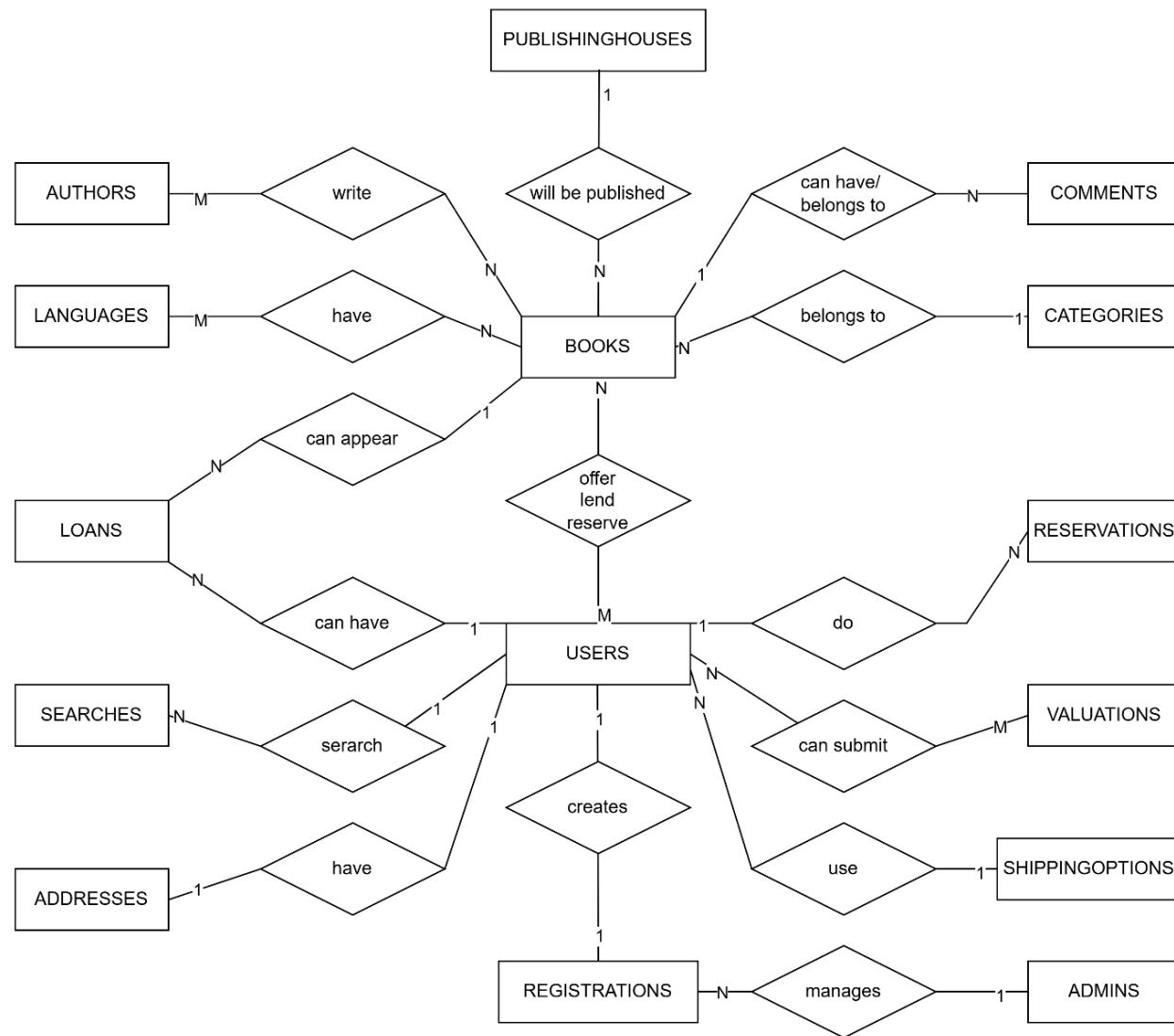
Die Entwicklung einer Datenbank für die Büchertausch-App erfolgt in mehreren strukturierten Schritten

- Anforderungsanalyse – um die Ziele der Datenbank zu definieren
- Datenmodell-Design – Identifizierung der Hauptentitäten, deren Attribute und Beziehungen
- Erstellung eines ER-Modells
- Implementierung der Datenbank

# ANFORDERUNGSANALYSE



- Das Ziel ist es, ein System für die lokale Gemeinschaft zu schaffen, das eine gemeinsame Nutzung von Büchern ermöglicht
- Die User können Bücher auflisten, die Ausleihdauer bestimmen, Abholorte angeben oder eine Versandoption anbieten
- App ermöglicht eine räumlich Suche, um verfügbare Bücher in ihrer Nähe zu sehen
- Eine Kalender-Funktion ermöglicht bereits ausgeliehene Bücher zu reservieren



ER MODELL



# IMPLEMENTIERUNGSVERFAHREN

- Für die Datenbank Implementierung wird zuerst ein Datenbank-Management-System (DBMS) ausgewählt, in diesem Fall PostgreSQL
- Um die Integrität der Daten und die Abfragegeschwindigkeit zu optimieren, werden Indizes und Fremdschlüsselbeziehungen hinzugefügt
- Es werden spezifische Regeln für die Eingabedaten festgelegt, um die Datenvalidierung sicherzustellen (z.B. E-Mail-Adressen, Telefonnummern , oder ISBN-Nummern.
- Definition von Pflichtfeldern
- Um die Funktionalität und Korrektheit der Validierungsregeln zu prüfen, werden Tests durchgeführt
- Testen der Datenbank durch diverses Abfragen, um die Leistung und Korrektheit zu gewährleisten
- Eventuell werden Optimierungen vorgenommen, um die Abfragegeschwindigkeit zu erhöhen

# ADMINS

Query Query History

```
1 --Erstellen der ADMINS-Tabelle
2
3 CREATE TABLE ADMINS (
4     AdminID SERIAL PRIMARY KEY,           --Eindeutige, automatisch inkrementierende Kennung des Admins
5     Firstname VARCHAR(100) NOT NULL,      --Vorname des Admins als Pflichtfeld bis 100 Zeichen
6     Surname VARCHAR(200) NOT NULL,       --Nachname des Admins als Pflichtfeld bis 200 Zeichen
7     EMail VARCHAR(200) UNIQUE NOT NULL   --Einmalige E-Mailadresse des Admins als Pflichtfeld bis 200 Zeichen
8 );
9
10 --Beispieldaten für ADMINS-Tabelle
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 31 msec.

Query Query History

```
10 --Beispieldaten für ADMINS-Tabelle
11
12 INSERT INTO ADMINS (Firstname, Surname, Email)
13     VALUES ('Martina', 'Oezkan', 'moez@iu-study.org'),
14            ('Maxim', 'Mustermann', 'maximilian.mustermann@study.de'),
15            ('Anton', 'Müller', 'am@gmail.de'),
16            ('Paula', 'Schlau', 'pschlau23@yahoo.com'),
17            ('Jule', 'Rich', 'jule.rich@t-online.de'),
18            ('Tilo', 'Sar', 'tilo66@gmail.com'),
19            ('Anne', 'Baumann', 'any4586@aol.de'),
20            ('Boris', 'Baecker', 'bbaecker@gmx.de'),
21            ('Sinem', 'Oezdal', 'Sino@yahoo.com'),
22            ('Murat', 'Kan', 'muki@example.de');
23
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 31 msec.

```
26 --Testfall der Tabelle ADMINS
27
28 SELECT * FROM ADMINS;|
29
30 --Erwartetes Ergebnis - Auflistung aller Studenten,
31 --die der Tabelle ADMINS hinzugefügt wurden und für die App
32 --Verwaltung zuständig sind.
33
```

Data Output Messages Notifications

≡+

📄

▼

📋

▼

🗑️

🔄

📥

📈

SQL

	adminid [PK] integer	firstname character varying (100)	surname character varying (200)	email character varying (200)
1	1	Martina	Oezkan	moez@iu-study.org
2	2	Maxim	Mustermann	maximilian.mustermann@study.de
3	3	Anton	Müller	am@gmail.de
4	4	Paula	Schlau	pschlau23@yahoo.com
5	5	Jule	Rich	jule.rich@t-online.de
6	6	Tilo	Sar	tilo66@gmail.com
7	7	Anne	Baumann	any4586@aol.de
8	8	Boris	Baecker	bbaecker@gmx.de
9	9	Sinem	Oezdal	Sino@yahoo.com
10	10	Murat	Kan	muki@example.de

# USERS

Query Query History

```
1  --Erstellen der USERS-Tabelle
2
3  CREATE TABLE USERS (
4      UserID SERIAL PRIMARY KEY,      --Eindeutige, automatisch inkrementierende Kennung des Users
5      Firstname VARCHAR(100) NOT NULL, --Vorname des Admins als Pflichtfeld bis 100 Zeichen
6      Surname VARCHAR(200) NOT NULL,   --Nachname des Admins als Pflichtfeld bis 200 Zeichen
7      EMail VARCHAR(200) UNIQUE NOT NULL, --Einmalige E-Mailadresse des Admins als Pflichtfeld bis 200 Zeichen
8      UserPassword VARCHAR(255) NOT NULL, --Passwort für die Anmeldung des Users (Zeichenanzahl für mögliche Hashwerte)
9      Phone VARCHAR(20)                --Telefonnummer des Users bis 20 Zeichen
10 );
```

Query Query History

```
1  --Beispieldaten für USERS-Tabelle
2
3  INSERT INTO USERS (Firstname, Surname, Email, UserPassword, Phone)
4      VALUES ('Max', 'Mueller', 'mm@beispiel.com', 'nsjcdi125#5', '0160222569' ),
5              ('Maxim', 'Muster', 'maxim123@gmx.de', '00009', '074268895648'),
6              ('Jana', 'Koncelmann', 'jk@gmail.de', '$Ser%215!', '01744488908'),
7              ('Petra', 'Schlauer', 'peta1@yahoo.com', 'petraS1auR', '015012345678'),
8              ('Julius', 'Keller', '33J33K@t-online.de', 'Kelo5', '0733566513'),
9              ('Ali', 'Sari', 'alis@gmail.com', 'ben&sen2025=biz', '0160987654321'),
10             ('Anna', 'Burger', 'any6@aol.de', 'Toplianska25', '0907863842'),
11             ('Sarah', 'Baecker', 'baecky@gmx.de', 'SB$4444!', '016078451296'),
12             ('Zeynep', 'Akyol', 'akyolzeynep@yahoo.com', 'Zeynep08012004', '0175553428'),
13             ('Davide', 'Weis', 'Davis@example.de', 'whitY#007', '017512312344');
```

```
15
16  --Testfall der Tabelle USERS
17
18  SELECT * FROM USERS;
19
20  --Erwartetes Ergebnis - Auflistung aller Benutzer,
21  --die der Tabelle USERS hinzugefügt wurden
```

Data Output Messages Notifications

	userid [PK] integer	firstname character varying (100)	surname character varying (200)	email character varying (200)	userpassword character varying (255)	phone character varying (20)
1	1	Max	Mueller	mm@beispiel.com	nsjcdi125#5	0160222569
2	2	Maxim	Muster	maxim123@gmx.de	00009	074268895648
3	3	Jana	Koncelmann	jk@gmail.de	\$Ser%215!	01744488908
4	4	Petra	Schlauer	peta1@yahoo.com	petraS1auR	015012345678
5	5	Julius	Keller	33J33K@t-online.de	Kelo5	0733566513
6	6	Ali	Sari	alis@gmail.com	ben&sen2025=biz	0160987654321
7	7	Anna	Burger	any6@aol.de	Toplianska25	0907863842
8	8	Sarah	Baecker	baecky@gmx.de	SB\$4444!	016078451296
9	9	Zeynep	Akyol	akyolzeynep@yahoo.com	Zeynep08012004	0175553428
10	10	Davide	Weis	Davis@example.de	whitY#007	017512312344

# REGISTRATIONS

Query Query History

```
1  --Erstellen der REGISTRATIONS-Tabelle
2
3  CREATE TABLE REGISTRATIONS (
4      RegID SERIAL PRIMARY KEY,           --Eindeutige, automatisch inkrementierende Kennung der Userregistrierung
5      UserID INT REFERENCES USERS(UserID), --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
6      AdminID INT REFERENCES ADMINS(AdminID), --Fremdschlüssel AdminID mit Bezug auf Tabelle ADMINS
7      RegistrationDate TIMESTAMP NOT NULL, --Zeitpunkt der Registrierung
8      Status INT CHECK (Status IN (0,1))   --Registrierungsstatus (aktiv = 1, inaktiv = 0)
9  );
10
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 36 msec.

```
12  --Beispieldaten für REGISTRATIONS-Tabelle
13
14  INSERT INTO REGISTRATIONS (UserID, AdminID, RegistrationDate, Status)
15  VALUES (9,1,'2025-01-15 10:23:00',1),
16          (2,3,'2025-01-02 11:44:00',1),
17          (7,1,'2025-02-01 10:00:00',0),
18          (9,1,'2025-01-15 00:20:00',1),
19          (1,2,'2025-04-25 07:23:01',1),
20          (3,1,'2025-03-15 10:23:00',1),
21          (5,1,'2025-01-30 09:13:50',0),
22          (4,2,'2025-03-15 10:27:00',1),
23          (8,2,'2025-01-15 07:26:00',1),
24          (6,3,'2025-02-28 03:33:00',1);
25
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 35 msec.

```
28  --Testfall der Tabelle REGISTRATIONS
29
30  SELECT * FROM REGISTRATIONS;
31
32  --Erwartetes Ergebnis - Auflistung aller Registrierungen
```

Data Output Messages Notifications

	regid [PK] integer	userid integer	adminid integer	registrationdate timestamp without time zone	status integer
1	1	9	1	2025-01-15 10:23:00	1
2	2	2	3	2025-01-02 11:44:00	1
3	3	7	1	2025-02-01 10:00:00	0
4	4	9	1	2025-01-15 00:20:00	1
5	5	1	2	2025-04-25 07:23:01	1
6	6	3	1	2025-03-15 10:23:00	1
7	7	5	1	2025-01-30 09:13:50	0
8	8	4	2	2025-03-15 10:27:00	1
9	9	8	2	2025-01-15 07:26:00	1
10	10	6	3	2025-02-28 03:33:00	1

# LANGUAGES

Query Query History

```
1 --Erstellen der LANGUAGES-Tabelle
2
3 CREATE TABLE LANGUAGES (
4     LanguageID SERIAL PRIMARY KEY, --Eindeutige, automatisch inkrementierende Kennung des Autors
5     Description VARCHAR(100)       --Bezeichnung der Sprache bis 100 Zeichen
6 );
7
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 46 msec.

```
9 --Beispieldaten für LANGUAGES-Tabelle
10
11 INSERT INTO LANGUAGES (Description)
12 VALUES ('Deutsch'),
13         ('Englisch'),
14         ('Franzoesich'),
15         ('Spanisch'),
16         ('Italienisch'),
17         ('Slowakisch'),
18         ('Türkisch'),
19         ('Arabisch'),
20         ('Portugiesisch'),
21         ('Griechisch');
22
23
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 53 msec.

```
25 --Testfall der Tabelle LANGUAGES
26
27 SELECT * FROM LANGUAGES;
28
29 --Erwartetes Ergebnis - Auflistung aller Sprachen
```

Data Output Messages Notifications

	languageid [PK] integer	description character varying (100)
1	1	Deutsch
2	2	Englisch
3	3	Franzoesich
4	4	Spanisch
5	5	Italienisch
6	6	Slowakisch
7	7	Türkisch
8	8	Arabisch
9	9	Portugiesisch
10	10	Griechisch



# AUTHORS

Query Query History

```
1  --Erstellen der AUTHORS-Tabelle
2
3  CREATE TABLE AUTHORS (
4      AuthorID SERIAL PRIMARY KEY,      --Eindeutige, automatisch inkrementierende Kennung des Autors
5      Firstname VARCHAR(100) NOT NULL,  --Vorname des Autors als Pflichtfeld bis 100 Zeichen
6      Surname VARCHAR(200) NOT NULL    --Nachname des Autors als Pflichtfeld bis 200 Zeichen
7  );
8
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 40 msec.

```
9
10 --Beispieldaten für AUTHORS-Tabelle
11
12 INSERT INTO AUTHORS (Firstname, Surname)
13     VALUES ('Franz','Kafka'),
14             ('Jane','Austen'),
15             ('Leo','Tolstoi'),
16             ('William','Shakespeare'),
17             ('Toni','Morrison'),
18             ('Haruki','Murakami'),
19             ('Sally','Rooney'),
20             ('Margaret','Atwood'),
21             ('Colleen','Hoover'),
22             ('Leila','Slimani');
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 31 msec.

```
26 --Testfall der Tabelle AUTHORS
27
28 SELECT * FROM AUTHORS;
29
30 --Erwartetes Ergebnis - Auflistung aller Autoren
```

Data Output Messages Notifications

	authorid [PK] integer	firstname character varying (100)	surname character varying (200)
1	1	Franz	Kafka
2	2	Jane	Austen
3	3	Leo	Tolstoi
4	4	William	Shakespeare
5	5	Toni	Morrison
6	6	Haruki	Murakami
7	7	Sally	Rooney
8	8	Margaret	Atwood
9	9	Colleen	Hoover
10	10	Leila	Slimani

# PUBLISHINGHOUSES

Query Query History

```
1  --Erstellen der PUBLISHINGHOUSES-Tabelle
2
3  CREATE TABLE PUBLISHINGHOUSES (
4      PubHouseID SERIAL PRIMARY KEY, --Eindeutige, automatisch inkrementierende Kennung des Verlages
5      Name VARCHAR(255) NOT NULL,    --Bezeichnung des Verlages bis 255 Zeichen als Pflichtfeld
6      FoundationYear INT,            --Gründungsjahr des Verlages
7      Phone VARCHAR(20),              --Telefonnummer des Verlages bis 20 Zeichen
8      Email VARCHAR(200),             --Einmalige E-Mailadresse des Verlages bis 200 Zeichen
9      Homepage VARCHAR(255)          --Webseite des Verlages bis 255 Zeichen
10 );
11
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 48 msec.

```
28
29 --Testfall der Tabelle PUBLISHINGHOUSES
30
```

```
31 SELECT * FROM PUBLISHINGHOUSES;
32
```

```
33 --Erwartetes Ergebnis - Auflistung aller Verlagshäuser
```

Data Output Messages Notifications

	pubhouseid [PK] integer	name character varying (255)	foundationyear integer	phone character varying (20)	email character varying (200)	homepage character varying (255)
1	1	Klett Gruppe	1785	03412396-0	info@klett.support	www.klett.de
2	2	Springer Nature	2015	-	info@springer.com	www.springer.com
3	3	Fischer Verlag GmbH	1886	+49 69 6062-0	info@fischerverlage.de	www.fischerverlage.de
4	4	Penguin Random House Verlagsgruppe GmbH	2009	0800 500 33 22	Kundenservice@penguin.de	www.penguin.de
5	5	Suhrkamp Verlag GmbH	1950	+49 30 740744-0	info@suhrkamp.de	www.suhrkamp.de
6	6	Carl Hanser Verlag GmbH & Co.KG	1928	+49 89 99830-0	info@hanser.de	www.hanser.de
7	7	HarperCollins Publishers	1989	+1 212-207-7000	-	www.harpercollins.com
8	8	Europa Editions	0	+39 06 3722829	info@europaeditons.com	www.europaeditons.com
9	9	World Editions	2013	-	-	www.worldeditions.org
10	10	Kehrer Verlag	1995	+49 711 6672-1216	svk@svk.de	www.kehrerverlag.com

```
13 --Beispieldaten für PUBLISHINGHOUSES-Tabelle
14
15 INSERT INTO PUBLISHINGHOUSES (Name, FoundationYear, Phone, Email, Homepage)
16 VALUES ('Klett Gruppe', 1785, '03412396-0', 'info@klett.support', 'www.klett.de'),
17 ('Springer Nature', 2015, '-', 'info@springer.com', 'www.springer.com'),
18 ('Fischer Verlag GmbH', 1886, '+49 69 6062-0', 'info@fischerverlage.de', 'www.fischerverlage.de'),
19 ('Penguin Random House Verlagsgruppe GmbH', 2009, '0800 500 33 22', 'Kundenservice@penguin.de', 'www.penguin.de'),
20 ('Suhrkamp Verlag GmbH', 1950, '+49 30 740744-0', 'info@suhrkamp.de', 'www.suhrkamp.de'),
21 ('Carl Hanser Verlag GmbH & Co.KG', 1928, '+49 89 99830-0', 'info@hanser.de', 'www.hanser.de'),
22 ('HarperCollins Publishers', 1989, '+1 212-207-7000', '-', 'www.harpercollins.com'),
23 ('Europa Editions', 0, '+39 06 3722829', 'info@europaeditons.com', 'www.europaeditons.com'),
24 ('World Editions', 2013, '-', '-', 'www.worldeditions.org'),
25 ('Kehrer Verlag', 1995, '+49 711 6672-1216', 'svk@svk.de', 'www.kehrerverlag.com');
26
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 33 msec.

# ADDRESSES

Query Query History

```
1  --Erstellen der ADDRESSES-Tabelle
2
3  CREATE TABLE ADDRESSES (
4      AddressID SERIAL PRIMARY KEY,          --Eindeutige, automatisch inkrementierende Kennung der Anschrift
5      UserID INT REFERENCES USERS(UserID),    --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
6      PubHouseID INT REFERENCES PUBLISHINGHOUSES(PubHouseID), --Fremdschlüssel PubHouseID mit Bezug auf Tabelle PUBLISHINGHOUSES
7      Street VARCHAR(100) UNIQUE NOT NULL,    --Straße bis 200 Zeichen
8      HouseNo VARCHAR(20) NOT NULL,          --Hausnummer bis 20 Zeichen
9      ZIPCode VARCHAR(6),                   --Postleitzahl bis 6 Zeichen
10     Place VARCHAR(100)                    --Ort bis 100 Zeichen
11 );
12
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 92 msec.

Query Query History

```
11 );
12
13 --Beispieldaten für ADDRESSES-Tabelle
14
15 INSERT INTO ADDRESSES (UserID, PubHouseID, Street, HouseNO, ZIPCode, Place)
16 VALUES (3, NULL, 'Langenweg', '3', '01589', 'Berlin' ),
17         (NULL, 1, 'Rotebuehlstrasse', '77', '70178', 'Stuttgart' ),
18         (NULL, 2, 'Europaplatz', '3', '69115', 'Heidelberg'),
19         (1, NULL, 'Hauptstrasse', '11B', '40213', 'Duesseldorf'),
20         (NULL, 9, 'Cassellastrasse', '30-32', '60386', 'Frankfurt am Main'),
21         (9, NULL, 'Landhausweg', '59', '50672', 'Koeln'),
22         (8, NULL, 'Friedrich-Ebert-Allee', '123', '53113', 'Bonn'),
23         (10, NULL, 'Isernhägener Strasse', '16', '30938', 'Burgwedel bei Hannover'),
24         (NULL, 10, 'Mannheimer Strasse', '175', '69123', 'Heidelberg'),
25         (5, NULL, 'Immenstaedter Strasse', '67/1/22', '87435', 'Kempten (Allgaeu));
26
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 39 msec.

```
29 --Testfall der Tabelle ADDRESSES
30
31 SELECT * FROM ADDRESSES;
32
33 --Erwartetes Ergebnis - Auflistung aller Anschriften,
34 --die der Tabelle ADDRESSES hinzugefügt wurden
35
```

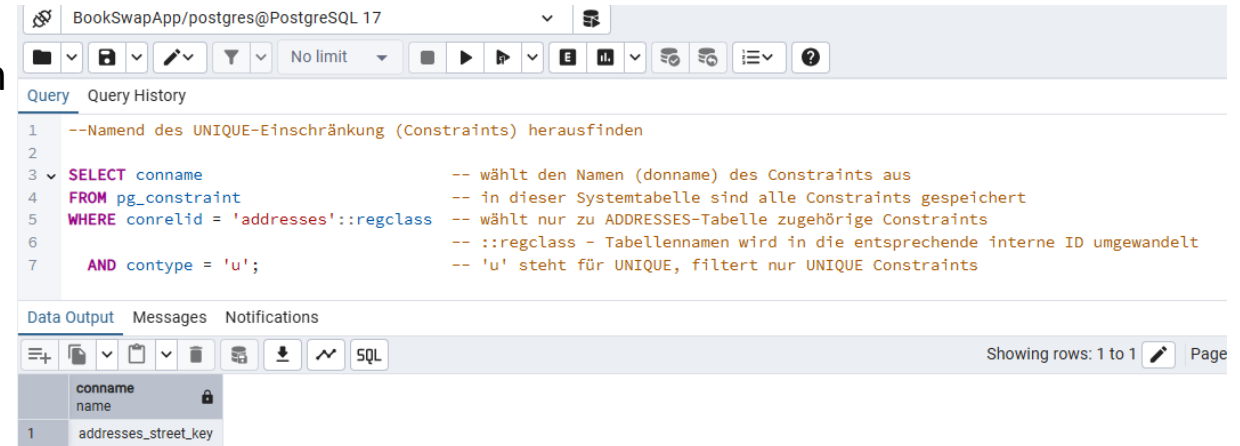
Data Output Messages Notifications

	addressid [PK] integer	userid integer	pubhouseid integer	street character varying (100)	houseid character varying (20)	zipcode character varying (6)	place character varying (100)
1	11	3	[null]	Langenweg	3	01589	Berlin
2	12	[null]	1	Rotebuehlstrasse	77	70178	Stuttgart
3	13	[null]	2	Europaplatz	3	69115	Heidelberg
4	14	1	[null]	Hauptstrasse	11B	40213	Duesseldorf
5	15	[null]	9	Cassellastrasse	30-32	60386	Frankfurt am Main
6	16	9	[null]	Landhausweg	59	50672	Koeln
7	17	8	[null]	Friedrich-Ebert-Allee	123	53113	Bonn
8	18	10	[null]	Isernhägener Strasse	16	30938	Burgwedel bei Hannover
9	19	[null]	10	Mannheimer Strasse	175	69123	Heidelberg
10	20	5	[null]	Immenstaedter Strasse	67/1/22	87435	Kempten (Allgaeu)

# ADDRESSES - CONSTRAINTS

Das Attribut Street ist als UNIQUE deklariert. Dies könnte Probleme bereiten, weil mehrere User auch auf derselben Straße wohnen können. Um Mehrfachnennungen bei Streets zu erlauben, muss das UNIQUE entfernt werden. Dies passiert in 2 Schritten:

1. UNIQUE-Einschränkung (Constraint) muss zuerst gefunden werden,
2. Constraint wird entfernt.



```
1 --Namend des UNIQUE-Einschränkung (Constraints) herausfinden
2
3 SELECT conname                                -- wählt den Namen (donname) des Constraints aus
4 FROM pg_constraint                            -- in dieser Systemtabelle sind alle Constraints gespeichert
5 WHERE conrelid = 'addresses'::regclass        -- wählt nur zu ADDRESSES-Tabelle zugehörige Constraints
6                                              -- ::regclass - Tabellennamen wird in die entsprechende interne ID umgewandelt
7 AND contype = 'u';                          -- 'u' steht für UNIQUE, filtert nur UNIQUE Constraints
```

conname
addresses_street_key



```
1 --UNIQUE-Einschränkung (Constraints) löschen
2
3 ALTER TABLE addresses                        -- ändere die Tabelle ADDRESSES
4 DROP CONSTRAINT addresses_street_key;        -- lösche Constraint mit dem Namen "addresses_street_key"
5
6 -- Ergebnis: die Spalte street ist nicht mehr einzigartig, es dürfen mehrere Adressen die selbe
7 -- Strasse haben
```

ALTER TABLE

Query returned successfully in 49 msec.

# CATEGORIES

```
37 --Erstellen der CATEGORIES-Tabelle
38
39 CREATE TABLE CATEGORIES (
40     CategoryID SERIAL PRIMARY KEY, --Eindeutige, automatisch inkrementierende Kennung der Kategorie
41     CatName VARCHAR(255)           --Kategorie Bezeichnung bis 255 Zeichen
42 );
43
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 47 msec.

```
45
46 INSERT INTO CATEGORIES (CatName)
47 VALUES ('Sachbuch' ),
48         ('Biografie'),
49         ('Historisch'),
50         ('Romantik'),
51         ('Thriller'),
52         ('Fantasy'),
53         ('Horror'),
54         ('Fiktion'),
55         ('Science Fiction'),
56         ('Kinderbuch');
57
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 36 msec.

```
60 --Testfall der Tabelle CATEGORIES
61
62 SELECT * FROM CATEGORIES;
63
64 --Erwartetes Ergebnis - Auflistung aller Kategorien,
65 --die der Tabelle CATEGORIES hinzugefügt wurden
66
```

Data Output Messages Notifications

SQL

	categoryid [PK] integer	catname character varying (255)
1	1	Sachbuch
2	2	Biografie
3	3	Historisch
4	4	Romantik
5	5	Thriller
6	6	Fantasy
7	7	Horror
8	8	Fiktion
9	9	Science Fiction
10	10	Kinderbuch



# SHIPPINGOPTIONS

```
80 --Erstellen der SHIPPINGOPTIONS-Tabelle
81
82 CREATE TABLE SHIPPINGOPTIONS (
83     ShippingID SERIAL PRIMARY KEY, --Eindeutige, automatisch inkrementierende Kennung der Versandoption
84     ShippingOption VARCHAR(100) CHECK (ShippingOption IN ('Mit Versand', 'Nur Abholung')), --Versandoption - Versand oder Abholung
85     ShippingDate TIMESTAMP, --Datum an dem das Buch versendet / abgeholt wurde
86     ShipmentNo VARCHAR(100), --Sendungsnummer / Tracking-Nummer zur Nachverfolgung beim Versand
87     ShipmentStatus VARCHAR(100) CHECK (ShipmentStatus IN ('Versendet', 'Zugestellt')), --Status des Versands (Versendet, Zugestellt)
88     AddressID INT REFERENCES ADDRESSES(AddressID) --Fremdschlüssel AddressID mit Bezug auf Tabelle ADDRESSES für Lieferadresse
89 );
90
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 34 msec.

```
91 --Beispieldaten für SHIPPINGOPTIONS-Tabelle
92
93 INSERT INTO SHIPPINGOPTIONS (ShippingOption, ShippingDate, ShipmentNo, ShipmentStatus, AddressID)
94 VALUES ('Mit Versand', '2025-07-11', 'SN123456996254', 'Versendet', 11),
95 ('Mit Versand', '2025-04-13', 'H6543212221514', 'Zugestellt', 12),
96 ('Nur Abholung', NULL, NULL, NULL, NULL),
97 ('Nur Abholung', NULL, NULL, NULL, NULL),
98 ('Mit Versand', '2025-01-23', 'SN901234', 'Zugestellt', 15),
99 ('Nur Abholung', NULL, NULL, NULL, NULL),
100 ('Nur Abholung', NULL, NULL, NULL, NULL),
101 ('Nur Abholung', NULL, NULL, NULL, NULL),
102 ('Nur Abholung', NULL, NULL, NULL, NULL),
103 ('Mit Versand', '2025-06-01', 'SN012678', 'Versendet', 20);
104
105
106
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 32 msec.

```
107 --Testfall der Tabelle SHIPPINGOPTIONS
108
```

```
109 SELECT * FROM SHIPPINGOPTIONS;
```

```
110
111 --Erwartetes Ergebnis - Auflistung aller Versandoptionen,
112 --die der Tabelle SHIPPINGOPTIONS hinzugefügt wurden
```

Data Output Messages Notifications

	shippingid [PK] integer	shippingoption character varying (100)	shippingdate timestamp without time zone	shipmentno character varying (100)	shipmentstatus character varying (100)	addressid integer
1	11	Mit Versand	2025-07-11 00:00:00	SN123456996254	Versendet	11
2	12	Mit Versand	2025-04-13 00:00:00	H6543212221514	Zugestellt	12
3	13	Nur Abholung	[null]	[null]	[null]	[null]
4	14	Nur Abholung	[null]	[null]	[null]	[null]
5	15	Mit Versand	2025-01-23 00:00:00	SN901234	Zugestellt	15
6	16	Nur Abholung	[null]	[null]	[null]	[null]
7	17	Nur Abholung	[null]	[null]	[null]	[null]
8	18	Nur Abholung	[null]	[null]	[null]	[null]
9	19	Nur Abholung	[null]	[null]	[null]	[null]
10	20	Mit Versand	2025-06-01 00:00:00	SN012678	Versendet	20

# BOOKS

```
17 --Beispieldaten für BOOKS-Tabelle
18
19 ▾ INSERT INTO BOOKS (AuthorID, PubHouseID, CategoryID, LanguageID, ShippingID, PublicationYear, Titel, ISBN, BookCondition, Pages)
20     VALUES (1, 1, 1, 5, 11, 1808, 'Faust', '978-3-16-148410-0', 'Neu', 200),
21             (2, 2, 2, 1, 12, 1813, 'Stolz und Vorurteil', '978-1-23-456789-7', 'Gebraucht', 300),
22             (3, 3, 4, 4, 13, 1599, 'Hamlet', '978-0-12-345678-9', 'Neu', 150),
23             (4, 4, 5, 3, 14, 1869, 'Krieg und Frieden', '978-3-16-148411-7', 'Gebraucht', 1200),
24             (5, 5, 3, 10, 15, 1869, 'Tom Sawyer', '978-0-98-765432-1', 'Neu', 350),
25             (6, 6, 6, 2, 16, 1987, 'Norwegian Wood', '978-4-20-123456-7', 'Neu', 400),
26             (7, 7, 7, 1, 17, 1929, 'Mrs Dalloway', '978-0-14-118247-5', 'Gebraucht', 250),
27             (8, 8, 8, 1, 18, 1795, 'Wallenstein', '978-3-16-148412-4', 'Neu', 180),
28             (9, 9, 9, 2, 19, 1859, 'A Tale of Two Cities', '978-0-14-143960-0', 'Neu', 320),
29             (10, 10, 10, 2, 20, 1926, 'The Sun Also Rises', '978-0-14-118263-5', 'Gebraucht', 280);
30
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 31 msec.

```
33 --Testfall der Tabelle BOOKS
34
35 SELECT * FROM BOOKS;
36
37 --Erwartetes Ergebnis - Auflistung aller Bücher,
38 --die der Tabelle BOOKS hinzugefügt wurden
39
```

Data Output Messages Notifications

	bookid [PK] integer	authorid integer	pubhouseid integer	categoryid integer	languageid integer	shippingid integer	publicationyear integer	titel character varying (255)	isbn character varying (20)	bookcondition character varying (255)	pages integer
1	1	1	1	1	5	11	1808	Faust	978-3-16-148410-0	Neu	200
2	2	2	2	2	1	12	1813	Stolz und Vorurteil	978-1-23-456789-7	Gebraucht	300
3	3	3	3	4	4	13	1599	Hamlet	978-0-12-345678-9	Neu	150
4	4	4	4	5	3	14	1869	Krieg und Frieden	978-3-16-148411-7	Gebraucht	1200
5	5	5	5	3	10	15	1869	Tom Sawyer	978-0-98-765432-1	Neu	350
6	6	6	6	6	2	16	1987	Norwegian Wood	978-4-20-123456-7	Neu	400
7	7	7	7	7	1	17	1929	Mrs Dalloway	978-0-14-118247-5	Gebraucht	250
8	8	8	8	8	1	18	1795	Wallenstein	978-3-16-148412-4	Neu	180
9	9	9	9	9	2	19	1859	A Tale of Two Cities	978-0-14-143960-0	Neu	320
10	10	10	10	10	2	20	1926	The Sun Also Rises	978-0-14-118263-5	Gebraucht	280

Query Query History

```
1 --Erstellen der BOOKS-Tabelle
2
3 ▾ CREATE TABLE BOOKS (
4     BookID SERIAL PRIMARY KEY, --Eindeutige, automatisch inkrementierende Kennung des Buches
5     AuthorID INT NOT NULL REFERENCES AUTHORS(AuthorID), --Fremdschlüssel AuthorID mit Bezug auf Tabelle AUTHORS
6     PubHouseID INT REFERENCES PUBLISHINGHOUSES(PubHouseID), --Fremdschlüssel PubHouseID mit Bezug auf Tabelle PUBLISHINGHOUSES
7     CategoryID INT REFERENCES CATEGORIES(CategoryID), --Fremdschlüssel CategoryID mit Bezug auf Tabelle CATEGORIES
8     LanguageID INT REFERENCES LANGUAGES(LanguageID), --Fremdschlüssel LanguageID mit Bezug auf Tabelle LANGUAGES
9     ShippingID INT REFERENCES SHIPPINGOPTIONS(ShippingID), --Fremdschlüssel ShippingID mit Bezug auf Tabelle SHIPPINGOPTIONS
10    PublicationYear INT, --Jahr der Veröffentlichung des Buches
11    Titel VARCHAR(255) NOT NULL, --Pflichtfeld Titel des Buches bis 255 Zeichen
12    ISBN VARCHAR(20) UNIQUE NOT NULL, --Eindeutige Internationale Standardbuchnummer
13    BookCondition VARCHAR(255), --Zustand des Buches bis 255 Zeichen
14    Pages INT --Anzahl der Seiten eines Buches
15 );
16
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 46 msec.

# LOANS

```
43 --Erstellen der LOANS-Tabelle
44
45 CREATE TABLE LOANS (
46     LoanID SERIAL PRIMARY KEY,           --Eindeutige, automatisch inkrementierende Kennung der Ausleihe
47     BookID INT REFERENCES BOOKS(BookID), --Fremdschlüssel BookID mit Bezug auf Tabelle Books
48     UserID INT REFERENCES USERS(UserID),  --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
49     LoanDate TIMESTAMP,                  --Datum an dem das Buch ausgeliehn würde
50     LoanPeriod INT,                      --Dauer der Ausleihe in Tagen
51     ReturnDate TIMESTAMP                 --Datum der Rückgabe des Buches
52 );
53
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 82 msec.

```
53
54 --Beispieldaten für LOANS-Tabelle
55
56 INSERT INTO LOANS (BookID, UserID, LoanDate, LoanPeriod, ReturnDate)
57 VALUES (1, 1, '2025-06-10 10:00:00', 14, '2025-06-24 10:00:00'),
58         (2, 2, '2025-05-12 14:30:00', 7, '2025-05-19 14:30:00'),
59         (3, 3, '2025-04-15 09:00:00', 21, NULL),
60         (4, 4, '2025-04-20 16:45:00', 14, NULL),
61         (5, 5, '2025-04-22 11:15:00', 10, NULL),
62         (6, 6, '2025-04-25 13:00:00', 14, NULL),
63         (7, 7, '2025-04-28 17:30:00', 7, NULL),
64         (8, 8, '2025-05-30 10:00:00', 14, NULL),
65         (9, 9, '2025-07-01 09:45:00', 21, NULL),
66         (10, 10, '2025-06-03 15:20:00', 14, '2025-06-17 18:35:00');
67
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 40 msec.

```
70 --Testfall der Tabelle LOANS
71
72 SELECT * FROM LOANS;
73
74 --Erwartetes Ergebnis - Auflistung aller Ausleihen,
75 --die der Tabelle LOANS hinzugefügt wurden
76
```

Data Output Messages Notifications

	loanid [PK] integer	bookid integer	userid integer	loandate timestamp without time zone	loanperiod integer	returndate timestamp without time zone
1	1	1	1	2025-06-10 10:00:00	14	2025-06-24 10:00:00
2	2	2	2	2025-05-12 14:30:00	7	2025-05-19 14:30:00
3	3	3	3	2025-04-15 09:00:00	21	[null]
4	4	4	4	2025-04-20 16:45:00	14	[null]
5	5	5	5	2025-04-22 11:15:00	10	[null]
6	6	6	6	2025-04-25 13:00:00	14	[null]
7	7	7	7	2025-04-28 17:30:00	7	[null]
8	8	8	8	2025-05-30 10:00:00	14	[null]
9	9	9	9	2025-07-01 09:45:00	21	[null]
10	10	10	10	2025-06-03 15:20:00	14	2025-06-17 18:35:00

# COMMENTS

Query Query History

```
1  --Erstellen der COMMENTS-Tabelle
2
3  CREATE TABLE COMMENTS (
4      CommentID SERIAL PRIMARY KEY,          --Eindeutige, automatisch inkrementierende Kennung des Kommentars
5      BookID INT REFERENCES BOOKS(BookID),    --Fremdschlüssel BookID mit Bezug auf Tabelle Books
6      UserID INT REFERENCES USERS(UserID) ,    --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
7      Score INT CHECK (Score BETWEEN 1 AND 5), --Sterne der Bewertung von 1 bis 5
8      Text TEXT,                             --Kommentar-Text
9      CommentDate TIMESTAMP                  --Aktuelles Datum der Abgabe eines Kommentars
10 );
11
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 48 msec.

Query Query History

```
11
12  --Beispieldaten für COMMENTS-Tabelle
13
14  INSERT INTO COMMENTS (BookID, UserID, Score, Text, CommentDate)
15      VALUES (1, 1, 5, 'Sehr Spannend!', '2025-07-14 10:00:00'),
16              (2, 2, 4, 'Ganz ok.', '2025-06-19 14:30:00'),
17              (3, 3, 3, 'Klassiker, muss man mögen.', '2025-05-15 09:00:00'),
18              (4, 4, 2, 'Sehr traurig und teilweise langweilig', '2025-05-20 16:45:00'),
19              (5, 5, 1, 'Naja', '2025-05-22 11:15:00'),
20              (6, 6, 2, 'Not the best.', '2025-05-25 13:00:00'),
21              (7, 7, 3, 'Buch ist ziemlich verbraucht, manche Seiten beschädigt, zum Lesen aber reicht.', '2025-05-28 17:30:00'),
22              (8, 8, 4, 'Für Drama- und Gedichtliebhaber ausgezeichnet', '2025-06-30 10:00:00'),
23              (9, 9, 5, 'Wer historische Romane mag, ist das klare Empfehlung', '2025-07-15 09:45:00'),
24              (10, 10, 1, 'Schwierig zu Lesen, somit hat es mich nicht gefesselt.', '2025-07-15 18:35:00');
25
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 38 msec.

```
28  --Testfall der Tabelle COMMENTS
29
30  SELECT * FROM COMMENTS;
31
32  --Erwartetes Ergebnis - Auflistung aller Kommentaren,
33  --die der Tabelle COMMENTS hinzugefügt wurden
34
```

Data Output Messages Notifications

	commentid [PK] integer	bookid integer	userid integer	score integer	text text	commentdate timestamp without time zone
1	1	1	1	5	Sehr Spannend!	2025-07-14 10:00:00
2	2	2	2	4	Ganz ok.	2025-06-19 14:30:00
3	3	3	3	3	Klassiker, muss man mögen.	2025-05-15 09:00:00
4	4	4	4	2	Sehr traurig und teilweise langweilig	2025-05-20 16:45:00
5	5	5	5	1	Naja	2025-05-22 11:15:00
6	6	6	6	2	Not the best.	2025-05-25 13:00:00
7	7	7	7	3	Buch ist ziemlich verbraucht, manche Seiten beschädigt, zum Lesen aber reicht.	2025-05-28 17:30:00
8	8	8	8	4	Für Drama- und Gedichtliebhaber ausgezeichnet	2025-06-30 10:00:00
9	9	9	9	5	Wer historische Romane mag, ist das klare Empfehlung	2025-07-15 09:45:00
10	10	10	10	1	Schwierig zu Lesen, somit hat es mich nicht gefesselt.	2025-07-15 18:35:00

# VALUTATIONS

```
1  --Erstellen der VALUATIONS-Tabelle
2
3  CREATE TABLE VALUATIONS (
4      ValuationID SERIAL PRIMARY KEY,      --Eindeutige, automatisch inkrementierende Kennung der Userbewertung
5      BookID INT REFERENCES BOOKS(BookID),  --Fremdschlüssel BookID mit Bezug auf Tabelle Books
6      UserID INT REFERENCES USERS(UserID),   --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
7      Score INT CHECK (Score BETWEEN 1 AND 5), --Sterne der Bewertung von 1 bis 5
8      Text TEXT,                             --Userbewertung-Text
9      ValuationDate TIMESTAMP                --Aktuelles Datum der Abgabe einer Bewertung
10 );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 53 msec.

```
12 --Beispieldaten für VALUATIONS-Tabelle
13
14 INSERT INTO VALUATIONS (BookID, UserID, Score, Text, ValuationDate)
15 VALUES (1, 1, 5, 'Sehr zuverlässig', '2025-07-15 10:00:00'),
16         (2, 2, 4, 'Alles bestens', '2025-07-10 14:30:00'),
17         (3, 3, 3, 'Ganz ok', '2025-06-15 09:00:00'),
18         (4, 4, 2, 'Total unflexibel', '2025-06-20 16:45:00'),
19         (5, 5, 1, 'Geht gar nicht', '2025-06-22 11:15:00'),
20         (6, 6, 2, 'Not the best.', '2025-06-25 13:00:00'),
21         (7, 7, 3, 'Buch ist ziemlich verbraucht, bin enttäuscht.', '2025-06-28 17:30:00'),
22         (8, 8, 4, 'Toller Kontakt', '2025-07-03 10:00:00'),
23         (9, 9, 5, 'Sehr nette Person, pünktlich und die Bücher sind 1A', '2025-07-15 11:45:00'),
24         (10, 10, 1, 'Kann ich leider nicht empfehlen.', '2025-07-15 18:40:00');
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 36 msec.

```
28 --Testfall der Tabelle VALUATIONS
29
30 SELECT * FROM VALUATIONS;
31
32 --Erwartetes Ergebnis - Auflistung aller Bewertungen,
33 --die der Tabelle VALUATIONS hinzugefügt wurden
34
```

Data Output Messages Notifications

	valuationid [PK] integer	bookid integer	userid integer	score integer	text text	valuationdate timestamp without time zone
1	1	1	1	5	Sehr zuverlässig	2025-07-15 10:00:00
2	2	2	2	4	Alles bestens	2025-07-10 14:30:00
3	3	3	3	3	Ganz ok	2025-06-15 09:00:00
4	4	4	4	2	Total unflexibel	2025-06-20 16:45:00
5	5	5	5	1	Geht gar nicht	2025-06-22 11:15:00
6	6	6	6	2	Not the best.	2025-06-25 13:00:00
7	7	7	7	3	Buch ist ziemlich verbraucht, bin enttäuscht.	2025-06-28 17:30:00
8	8	8	8	4	Toller Kontakt	2025-07-03 10:00:00
9	9	9	9	5	Sehr nette Person, pünktlich und die Bücher sind 1A	2025-07-15 11:45:00
10	10	10	10	1	Kann ich leider nicht empfehlen.	2025-07-15 18:40:00



# RESERVATIONS

Query Query History

```
1  --Erstellen der RESERVATIONS-Tabelle
2
3  CREATE TABLE RESERVATIONS (
4      ReservationID SERIAL PRIMARY KEY,      --Eindeutige, automatisch inkrementierende Kennung der Reservierung
5      BookID INT REFERENCES BOOKS(BookID),   --Fremdschlüssel BookID mit Bezug auf Tabelle Books
6      UserID INT REFERENCES USERS(UserID),    --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
7      ReservationFrom TIMESTAMP,             --Startdatum der Reservierung
8      ReservationUntill TIMESTAMP            --Enddatum der Reservierung
9  );
10
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 54 msec.

Query Query History

```
11  --Beispieldaten für RESERVATIONS-Tabelle
12
13  INSERT INTO RESERVATIONS (BookID, UserID, ReservationFrom, ReservationUntill)
14      VALUES
15      (1, 10, '2025-08-15 10:00:00', '2025-08-18 10:00:00'),
16      (2, 9, '2025-08-10 14:30:00', '2025-08-13 14:30:00'),
17      (3, 8, '2025-07-15 09:00:00', '2025-07-18 09:00:00'),
18      (4, 7, '2025-07-20 16:45:00', '2025-07-23 16:45:00'),
19      (5, 6, '2025-07-22 11:15:00', '2025-07-25 11:15:00'),
20      (6, 5, '2025-07-25 13:00:00', '2025-07-28 13:00:00'),
21      (7, 4, '2025-07-28 17:30:00', '2025-07-30 17:30:00'),
22      (8, 3, '2025-08-03 10:00:00', '2025-08-07 10:00:00'),
23      (9, 2, '2025-08-15 11:45:00', '2025-08-18 11:45:00'),
24      (10, 1, '2025-08-15 18:40:00', '2025-08-18 18:40:00');
```

Data Output Messages Notifications

INSERT 0 10

Query returned successfully in 44 msec.

```
26
27  --Testfall der Tabelle RESERVATIONS
28
29  SELECT * FROM RESERVATIONS;
30
31  --Erwartetes Ergebnis - Auflistung aller Buchreservierungen,
32  --die der Tabelle RESERVATIONS hinzugefügt wurden
33
```

Data Output Messages Notifications

	reservationid [PK] integer	bookid integer	userid integer	reservationfrom timestamp without time zone	reservationuntill timestamp without time zone
1	1	1	10	2025-08-15 10:00:00	2025-08-18 10:00:00
2	2	2	9	2025-08-10 14:30:00	2025-08-13 14:30:00
3	3	3	8	2025-07-15 09:00:00	2025-07-18 09:00:00
4	4	4	7	2025-07-20 16:45:00	2025-07-23 16:45:00
5	5	5	6	2025-07-22 11:15:00	2025-07-25 11:15:00
6	6	6	5	2025-07-25 13:00:00	2025-07-28 13:00:00
7	7	7	4	2025-07-28 17:30:00	2025-07-30 17:30:00
8	8	8	3	2025-08-03 10:00:00	2025-08-07 10:00:00
9	9	9	2	2025-08-15 11:45:00	2025-08-18 11:45:00
10	10	10	1	2025-08-15 18:40:00	2025-08-18 18:40:00

# SEARCHES

```
Query  Query History
1  --Erstellen der SEARCHES-Tabelle
2
3  CREATE TABLE SEARCHES (
4      SearchID SERIAL PRIMARY KEY,      --Eindeutige, automatisch inkrementierende Kennung des Suchvorgangs
5      UserID INT REFERENCES USERS(UserID), --Fremdschlüssel UserID mit Bezug auf Tabelle USERS
6      SearchTerm Varchar(255),          --Texteingabe der Suche (Titel, Autor, Verlag...)
7      BookLocation Varchar(255),        --Standort - Adresseingabe
8      Availability BOOLEAN              --Verfügbarkeit , ob nur verfügbare Bücher aufgelistet werden sollen
9  );
10
```

Data Output Messages Notifications

```
CREATE TABLE
```

Query returned successfully in 65 msec.

```
11  --Beispieldaten für SEARCHES-Tabelle
12
13  INSERT INTO SEARCHES (UserID, SearchTerm, BookLocation, Availability)
14      VALUES (1, 'Kinderbuch',NULL, TRUE),
15              (2, 'Kafka',NULL, FALSE),
16              (3, 'Hoover','Koeln', FALSE),
17              (4, NULL,'Berlin', TRUE),
18              (5, NULL,NULL,TRUE),
19              (6, 'Klett Gruppe',NULL, TRUE),
20              (7, 'Fischer Verlag',NULL, FALSE),
21              (8, 'Romantik','Stuttgart', FALSE),
22              (9, 'Fantasy',NULL, FALSE),
23              (10, NULL,'Bonn', TRUE);
24
```

Data Output Messages Notifications

```
INSERT 0 10
```

Query returned successfully in 52 msec.

Diese Tabelle dient als Eingabequelle, hier gibt es statt JOINS Textvergleiche. Sie kann temporär sein und nur ein Datensatz beinhalten, oder als Such-Historie mit Mehreren Datensätzen (Benutzerkontext) wie in diesem Beispiel.

```
27  --Testfall der Tabelle SEARCHES
28
29  SELECT * FROM SEARCHES;
30
31  --Erwartetes Ergebnis - Auflistung aller Suchvorgängen pro User,
32  --die der Tabelle SEARCHES hinzugefügt wurden
33
```

Data Output Messages Notifications

	searchid [PK] integer	userid integer	searchterm character varying (255)	booklocation character varying (255)	availability boolean
1	1	1	Kinderbuch	[null]	true
2	2	2	Kafka	[null]	false
3	3	3	Hoover	Koeln	false
4	4	4	[null]	Berlin	true
5	5	5	[null]	[null]	true
6	6	6	Klett Gruppe	[null]	true
7	7	7	Fischer Verlag	[null]	false
8	8	8	Romantik	Stuttgart	false
9	9	9	Fantasy	[null]	false
10	10	10	[null]	Bonn	true

# TESTFÄLLE

BookSwapApp/postgres@PostgreSQL 17

Query

Query History

```
1 -- Suche nach Bücher in mehreren Feldern der BOOKS-Tabelle anhand der letzter Suche von User 8
2
3 SELECT B.* --Liste alles
4 FROM BOOKS B --aus Tabelle BOOKS
5 JOIN AUTHORS A ON B.AuthorID = A.AuthorID --Verknüpfe die Tabelle BOOKS mit AUTHORS
6 JOIN PUBLISHINGHOUSES P ON B.PubHouseID = P.PubHouseID --Verknüpfe die Tabelle BOOKS mit PUBLISHINGHOUSES
7 JOIN CATEGORIES C ON B.CategoryID = C.CategoryID --Verknüpfe die Tabelle BOOKS mit CATEGORIES
8 WHERE --Unter Bedingung
9     B.Titel ILIKE (SELECT SearchTerm FROM SEARCHES WHERE UserID = 8 ORDER BY SearchID DESC LIMIT 1)
10 -- das Buchtitel mit der letzten Sucheingabe des Users 8 übereinstimmt
11 OR A.Surname ILIKE (SELECT SearchTerm FROM SEARCHES WHERE UserID = 8 ORDER BY SearchID DESC LIMIT 1)
12 -- oder Nachname des Autors mit der letzten Sucheingabe des Users 8 übereinstimmt
13 OR P.Name ILIKE (SELECT SearchTerm FROM SEARCHES WHERE UserID = 8 ORDER BY SearchID DESC LIMIT 1)
14 -- oder Verlagsname mit der letzten Sucheingabe des Users 8 übereinstimmt
15 OR C.CatName ILIKE (SELECT SearchTerm FROM SEARCHES WHERE UserID = 8 ORDER BY SearchID DESC LIMIT 1);
16 -- oder die Kategoriebezeichnung eines Buches mit der letzten Sucheingabe des Users 8 übereinstimmt
17
18 -- ORDER BY SearchID DESC = Sortiert nach der SearchID absteigend, d.h. die höchste ID kommt zuerst
19 -- LIMIT 1 = Es wird nur der letzte (neueste) Eintrag zurückgegeben
```

Data Output

Messages

Notifications

	bookid integer	authorid integer	pubhouseid integer	categoryid integer	languageid integer	shippingid integer	publicationyear integer	titel character varying (255)	isbn character varying (20)	bookcondition character varying (255)	pages integer	name character varying (255)
1	2	2	2	2	1	12	1813	Stolz und Vorurteil	978-1-23-456789-7	Gebraucht	300	Springer Nature
2	5	5	5	3	10	15	1869	Tom Sawyer	978-0-98-765432-1	Neu	350	Suhrkamp Verlag GmbH

BookSwapApp/postgres@PostgreSQL 17

Query

Query History

```
1 -- Suche nach Bücher, die einem Verlag mit der Anfangsbuchstabe "S" gehören
2 SELECT B.*, P.Name --Liste alles aus Tabelle BOOKS + Name des Verlags
3 FROM BOOKS B --aus Tabelle BOOKS
4 JOIN PUBLISHINGHOUSES P ON B.PubHouseID = P.PubHouseID --Verknüpfe die Tabelle BOOKS mit PUBLISHINGHOUSES
5 WHERE --Unter Bedingung
6     P.Name ILIKE 'S%'; --dass Name des Verlags mit S
```

Data Output

Messages

Notifications

	bookid integer	authorid integer	pubhouseid integer	categoryid integer	languageid integer	shippingid integer	publicationyear integer	titel character varying (255)	isbn character varying (20)	bookcondition character varying (255)	pages integer	name character varying (255)
1	2	2	2	2	1	12	1813	Stolz und Vorurteil	978-1-23-456789-7	Gebraucht	300	Springer Nature
2	5	5	5	3	10	15	1869	Tom Sawyer	978-0-98-765432-1	Neu	350	Suhrkamp Verlag GmbH

BookSwapApp/postgres@PostgreSQL 17

Query

Query History

```
1 -- Liste alle Reservierungen inkl. Username, Buchtitel, ISBN-Nummer, Buchzustand, Veröffentlichungsjahr und Seitenanzahl
2 SELECT R.*, U.Surname, B.Titel, B.ISBN, B.BookCondition, B.Publicationyear, B.Pages
3 FROM RESERVATIONS R --aus Tabelle RESERVATIONS
4 JOIN BOOKS B ON R.BookID = B.BookID --Verknüpfe die Tabelle RESERVATIONS mit BOOKS
5 JOIN USERS U ON R.UserID = U.UserID; --Verknüpfe die Tabelle RESERVATIONS mit USERS
6
```

Data Output

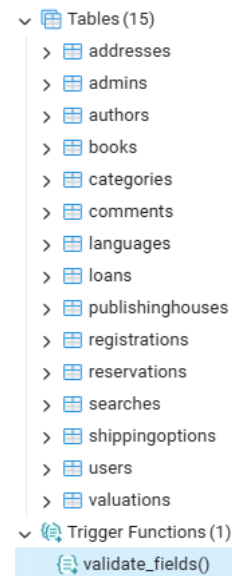
Messages

Notifications

	reservationid integer	bookid integer	userid integer	reservationfrom timestamp without time zone	reservationuntil timestamp without time zone	surname character varying (200)	titel character varying (255)	isbn character varying (20)	bookcondition character varying (255)	publicationyear integer	pages integer
1	1	1	10	2025-08-15 10:00:00	2025-08-18 10:00:00	Weis	Faust	978-3-16-148410-0	Neu	1808	200
2	2	2	9	2025-08-10 14:30:00	2025-08-13 14:30:00	Akylol	Stolz und Vorurteil	978-1-23-456789-7	Gebraucht	1813	300
3	3	3	8	2025-07-18 09:00:00	2025-07-18 09:00:00	Baecker	Hamlet	978-0-12-345678-9	Neu	1599	150
4	4	4	7	2025-07-23 16:45:00	2025-07-23 16:45:00	Burger	Krieg und Frieden	978-3-16-148411-7	Gebraucht	1869	1200
5	5	5	6	2025-07-22 11:15:00	2025-07-25 11:15:00	Sari	Tom Sawyer	978-0-98-765432-1	Neu	1869	350
6	6	6	5	2025-07-25 13:00:00	2025-07-28 13:00:00	Keller	Norwegian Wood	978-4-20-123456-7	Neu	1987	400
7	7	7	4	2025-07-28 17:30:00	2025-07-30 17:30:00	Schlauer	Mrs Dalloway	978-0-14-118247-5	Gebraucht	1929	250
8	8	8	3	2025-08-03 10:00:00	2025-08-03 10:00:00	Koncelmann	Wallenstein	978-3-16-148412-4	Neu	1795	180
9	9	9	2	2025-08-15 11:45:00	2025-08-18 11:45:00	Muster	A Tale of Two Cities	978-0-14-143960-0	Neu	1859	320
10	10	10	1	2025-08-15 18:40:00	2025-08-18 18:40:00	Mueller	The Sun Also Rises	978-0-14-118263-5	Gebraucht	1926	280

# VALIDIERUNGSREGELN FÜR DATENFORMATE

Um die Datenintegrität sowie Benutzerfreundlichkeit zu verbessern, können Validierungsregeln für Bestimmte Datenformate wie Telefonnummer, ISBN-Nummer oder E-Mails festgelegt werden. Dafür werden Funktionen – Trigger erstellt.



```
BookSwapApp/postgres@PostgreSQL 17
Query Query History
1 CREATE OR REPLACE FUNCTION validate_fields() RETURNS TRIGGER AS $$
2 BEGIN
3     --E-Mail-Validierung
4     IF NEW.Email IS NOT NULL AND NEW.Email !~* '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' THEN
5         RAISE EXCEPTION 'Ungültige E-Mail-Adresse: %', NEW.Email;
6     END IF;
7     -- Stellt sicher, dass E-Mails korrekt aufgebaut sind
8
9     --Telefonnummer-Validierung (nur Ziffern, +, -, /, Leerzeichen erlaubt)
10 IF NEW.Phone IS NOT NULL AND NEW.Phone !~* '[0-9+ \- / \s]+' THEN
11     RAISE EXCEPTION 'Ungültige Telefonnummer: %', NEW.Phone;
12 END IF;
13 --Prüft die Telefonnummern auf unerlaubte Zeichen
14
15 --ISBN-Validierung (nur für Tabelle BOOKS)
16 IF TG_TABLE_NAME = 'BOOKS' AND (NEW.ISBN IS NULL OR NEW.ISBN !~* '^(d{10}|d{13})$') THEN
17     RAISE EXCEPTION 'Ungültige ISBN (muss 10 oder 13 Ziffern enthalten): %', NEW.ISBN;
18 END IF;
19 --Prüft grob, ob die ISBN-Struktur passt
20
21 RETURN NEW;
22 END;
23 $$ LANGUAGE plpgsql;
```

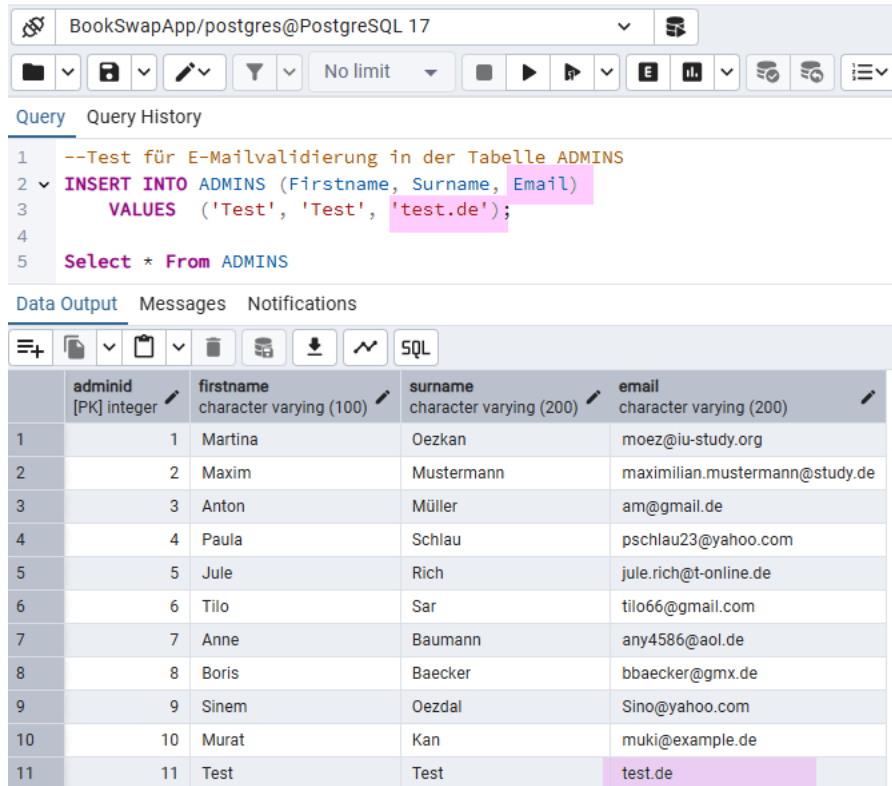
Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 34 msec.

# WEITERE TESTFÄLLE

Die ersten Tests verliefen zunächst vielversprechend. Im weiteren Verlauf zeigte sich jedoch, dass der Testumfang unzureichend war.



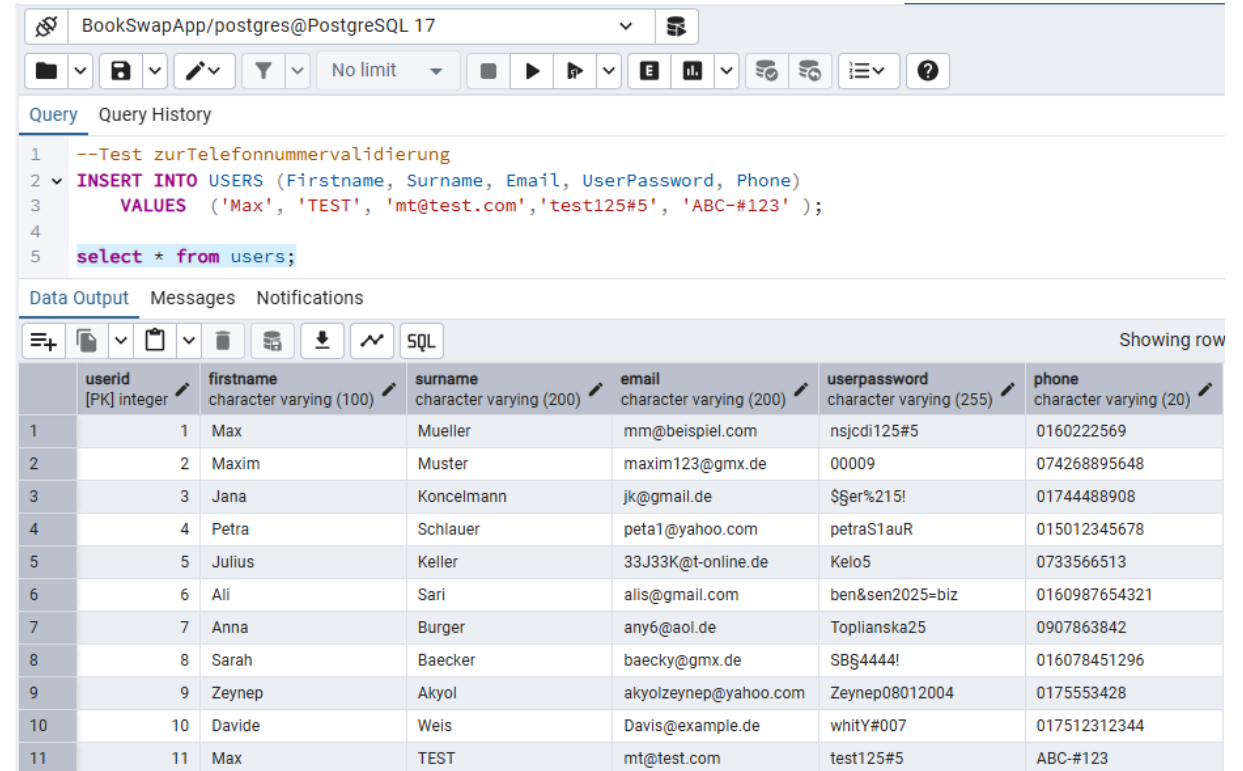
BookSwapApp/postgres@PostgreSQL 17

Query Query History

```
1 --Test für E-Mailvalidierung in der Tabelle ADMINS
2 INSERT INTO ADMINS (Firstname, Surname, Email)
3 VALUES ('Test', 'Test', 'test.de');
4
5 Select * From ADMINS
```

Data Output Messages Notifications

	adminid [PK] integer	firstname character varying (100)	surname character varying (200)	email character varying (200)
1	1	Martina	Oezkan	moez@iu-study.org
2	2	Maxim	Mustermann	maximilian.mustermann@study.de
3	3	Anton	Müller	am@gmail.de
4	4	Paula	Schlau	pschlau23@yahoo.com
5	5	Jule	Rich	jule.rich@t-online.de
6	6	Tilo	Sar	tilo66@gmail.com
7	7	Anne	Baumann	any4586@aol.de
8	8	Boris	Baecker	bbaecker@gmx.de
9	9	Sinem	Oezdal	Sino@yahoo.com
10	10	Murat	Kan	muki@example.de
11	11	Test	Test	test.de



BookSwapApp/postgres@PostgreSQL 17

Query Query History

```
1 --Test zurTelefonnummervalidierung
2 INSERT INTO USERS (Firstname, Surname, Email, UserPassword, Phone)
3 VALUES ('Max', 'TEST', 'mt@test.com', 'test125#5', 'ABC-#123' );
4
5 select * from users;
```

Data Output Messages Notifications

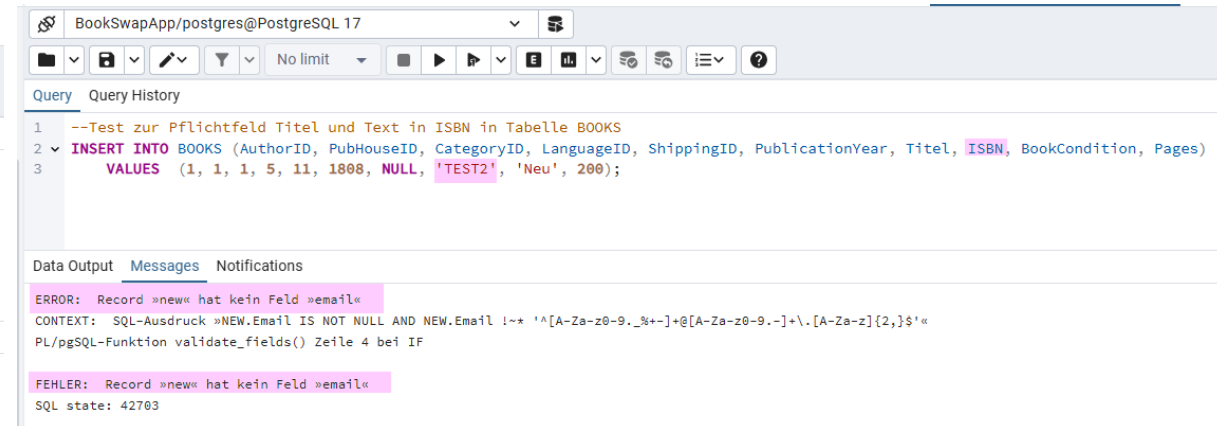
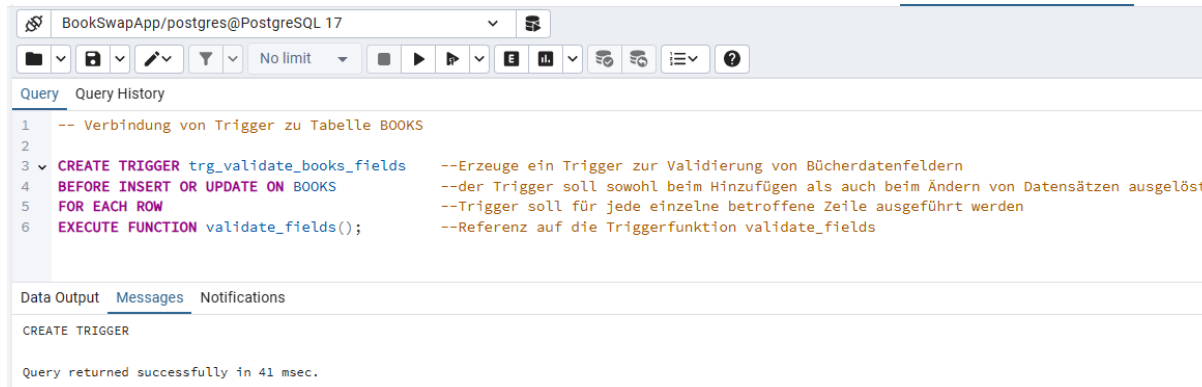
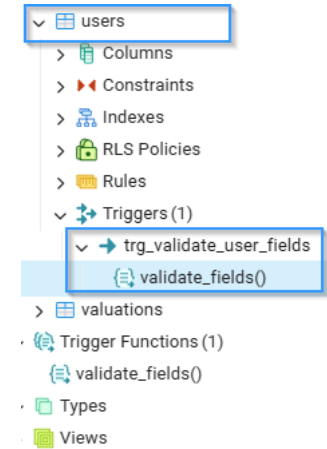
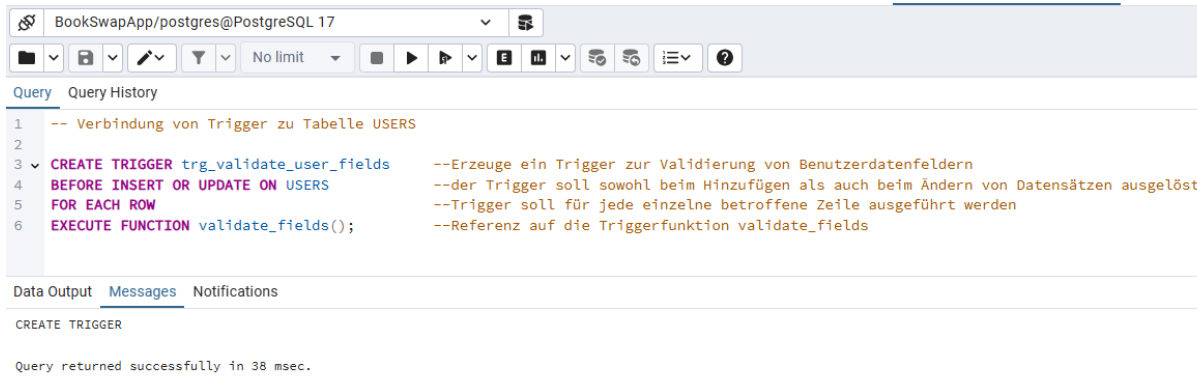
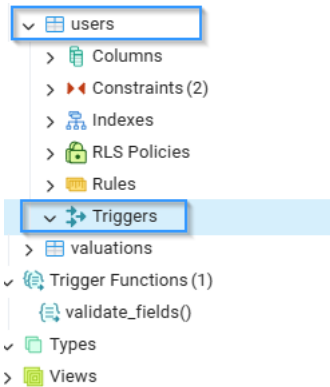
	userid [PK] integer	firstname character varying (100)	surname character varying (200)	email character varying (200)	userpassword character varying (255)	phone character varying (20)
1	1	Max	Mueller	mm@beispiel.com	nsjcdi125#5	0160222569
2	2	Maxim	Muster	maxim123@gmx.de	00009	074268895648
3	3	Jana	Koncelmann	jk@gmail.de	\$Ser%215!	01744488908
4	4	Petra	Schlauer	peta1@yahoo.com	petraS1auR	015012345678
5	5	Julius	Keller	33J33K@t-online.de	Kelo5	0733566513
6	6	Ali	Sari	alis@gmail.com	ben&sen2025=biz	0160987654321
7	7	Anna	Burger	any6@aol.de	Toplianska25	0907863842
8	8	Sarah	Baecker	baecky@gmx.de	SB\$4444!	016078451296
9	9	Zeynep	Akyol	akyolzeynep@yahoo.com	Zeynep08012004	0175553428
10	10	Davide	Weis	Davis@example.de	whitY#007	017512312344
11	11	Max	TEST	mt@test.com	test125#5	ABC-#123

Insbesondere bei der Validierung fehlerhafter Eingaben – etwa bei E-Mail-Adressen oder Telefonnummern – wurde keine Fehlermeldung angezeigt. Trotz einer vorhandenen Trigger-Funktion war es dadurch möglich, ungültige Daten in das System zu übernehmen.



# ZUWEISUNG VON TRIGGERN

Dabei wurde festgestellt, dass zwar eine Trigger-Funktion existiert, diese jedoch nicht den entsprechenden Tabellen zugewiesen war.



# KORREKTUR DER TRIGGER-FUNKTION

Die Funktion `validate_fields()` muss erweitert werden, da sie aktuell pauschal versucht das Feld `NEW.Email` aufzurufen, was in der Tabelle `BOOKS` nicht vorhanden ist.

Somit muss eine feldabhängige Validierung gezielt prüfen, ob dieses Feld auch vorhanden ist:

```
Query Query History
1 --Korrektur der Trigger Funktion
2
3 CREATE OR REPLACE FUNCTION validate_fields() RETURNS TRIGGER AS $$      --aktualisiert bestehende Funktion in-place
4 DECLARE
5     isbn_clean TEXT;                                                    --Variable für bereinigte ISBN (Nur Ziffern)
6     sum INT := 0;                                                       --Zwischensumme für ISBN-Prüfzifferberechnung
7     digit INT;                                                          --Variable speichert der jeweiligen Ziffer
8 BEGIN
9     -- E-Mail-Prüfung nur in relevanten Tabellen USERS, ADMINS und PUBLISHINGHOUSES
10    IF TG_TABLE_NAME IN ('users', 'admins', 'publishinghouses') THEN
11        IF NEW.Email IS NOT NULL AND NEW.Email !~* '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' THEN
12            RAISE EXCEPTION 'Ungültige E-Mail-Adresse in Tabelle %: %', TG_TABLE_NAME, NEW.Email;
13        END IF;
14    END IF;
15
16    -- Telefonnummer-Prüfung nur in USERS und PUBLISHINGHOUSES
17    IF TG_TABLE_NAME IN ('users', 'publishinghouses') THEN
18        IF NEW.Phone IS NOT NULL AND NEW.Phone !~* '^[0-9+|-|/|s]+$' THEN
19            RAISE EXCEPTION 'Ungültige Telefonnummer in Tabelle %: %', TG_TABLE_NAME, NEW.Phone;
20        END IF;
21    END IF;
22
```

```
23 -- ISBN-Prüfung nur in BOOKS
24 IF TG_TABLE_NAME = 'books' AND
25     NEW.ISBN IS NOT NULL THEN
26     isbn_clean := regexp_replace(NEW.ISBN, '^[0-9Xx]', '', 'g');
27
28
29 -- ISBN-10 mit Prüfziffer
30 IF length(isbn_clean) = 10 THEN
31     FOR i IN 1..9 LOOP
32         digit := CAST(substring(isbn_clean, i, 1) AS INTEGER);
33         sum := sum + digit * i;
34     END LOOP;
35 IF upper(substring(isbn_clean, 10, 1)) = 'X' THEN
36     sum := sum + 10 * 10;
37 ELSE
38     digit := CAST(substring(isbn_clean, 10, 1) AS INTEGER);
39     sum := sum + digit * 10;
40 END IF;
41 IF sum % 11 <> 0 THEN
42     RAISE EXCEPTION 'Ungültige ISBN-10 Prüfziffer: %', NEW.ISBN;
43 END IF;
44
45 -- ISBN-13 mit Prüfziffer
46 ELSIF length(isbn_clean) = 13 THEN
47     sum := 0;
48     FOR i IN 1..12 LOOP
49         digit := CAST(substring(isbn_clean, i, 1) AS INTEGER);
50         IF i % 2 = 0 THEN
51             sum := sum + digit * 3;
52         ELSE
53             sum := sum + digit;
54         END IF;
55     END LOOP;
56     digit := CAST(substring(isbn_clean, 13, 1) AS INTEGER);
57 IF (10 - (sum % 10)) % 10 <> digit THEN
58     RAISE EXCEPTION 'Ungültige ISBN-13 Prüfziffer: %', NEW.ISBN;
59 END IF;
60 ELSE
61     RAISE EXCEPTION 'ISBN muss 10 oder 13 Zeichen lang sein: %', NEW.ISBN;
62 END IF;
63 END IF;
64
65 RETURN NEW;
66 END;
67 $$ LANGUAGE plpgsql;
```

# VALIDATIONSTESTS

BookSwapApp/postgres@PostgreSQL 17

Query Query History

```
1 --Test zur Pflichtfeld Titel und Text in ISBN in Tabelle BOOKS
2 INSERT INTO BOOKS (AuthorID, PubHouseID, CategoryID, LanguageID, ShippingID, PublicationYear, Titel, ISBN, BookCondition, Pages)
3 VALUES (1, 1, 1, 5, 11, 1808, NULL, 'TEST', 'Neu', 200);
```

Data Output Messages Notifications

ERROR: ISBN muss 10 oder 13 Zeichen lang sein: TEST  
CONTEXT: PL/pgSQL-Funktion validate\_fields() Zeile 59 bei RAISE

FEHLER: ISBN muss 10 oder 13 Zeichen lang sein: TEST  
SQL state: P0001

BookSwapApp/postgres@PostgreSQL 17

Query Query History

```
1 --Test zur Pflichtfeld Titel und Text in ISBN in Tabelle BOOKS
2 INSERT INTO BOOKS (AuthorID, PubHouseID, CategoryID, LanguageID, ShippingID, PublicationYear, Titel, ISBN, BookCondition, Pages)
3 VALUES (1, 1, 1, 5, 11, 1808, NULL, '0123456789', 'Neu', 200);
```

Data Output Messages Notifications

ERROR: NULL-Wert in Spalte »titel« von Relation »books« verletzt Not-Null-Constraint  
Fehlgeschlagene Zeile enthält (17, 1, 1, 1, 5, 11, 1808, null, 0123456789, Neu, 200).

FEHLER: NULL-Wert in Spalte »titel« von Relation »books« verletzt Not-Null-Constraint  
SQL state: 23502  
Detail: Fehlgeschlagene Zeile enthält (17, 1, 1, 1, 5, 11, 1808, null, 0123456789, Neu, 200).

BookSwapApp/postgres@PostgreSQL 17

Query Query History

```
1 -- Falsche Prüfziffer (ISBN-13)
2 INSERT INTO BOOKS (AuthorID, PubHouseID, Titel, ISBN)
3 VALUES (1, 1, 'Falsche ISBN 13', '9783161484101'); -- letzte Ziffer falsch
```

Data Output Messages Notifications

ERROR: Ungültige ISBN-13 Prüfziffer: 9783161484101  
CONTEXT: PL/pgSQL-Funktion validate\_fields() Zeile 56 bei RAISE

FEHLER: Ungültige ISBN-13 Prüfziffer: 9783161484101  
SQL state: P0001

Query Query History

```
1 --Ungültige Email in Teabell ADMINS eingügen
2
3 INSERT INTO ADMINS (Firstname, Surname, EMail)
4 VALUES ('Fehler', 'Admin', 'admin(at)email.de');
```

Data Output Messages Notifications

ERROR: Ungültige E-Mail-Adresse in Tabelle admins: admin(at)email.de  
CONTEXT: PL/pgSQL-Funktion validate\_fields() Zeile 10 bei RAISE

FEHLER: Ungültige E-Mail-Adresse in Tabelle admins: admin(at)email.de  
SQL state: P0001

BookSwapApp/postgres@PostgreSQL 17

Query Query History

```
1
2 --Reservierung über Jahreswechsel
3
4 INSERT INTO RESERVATIONS (BookID, UserID, ReservationFrom, ReservationUntill)
5 VALUES (1, 1, '2024-12-31 23:00:00', '2025-01-01 08:00:00');
6
7 --Prüfung: Daten korrekt gespeichert?
8 SELECT * FROM RESERVATIONS
9 WHERE ReservationFrom >= '2024-12-31 23:00:00' AND ReservationUntill <= '2025-01-01 08:00:00';
```

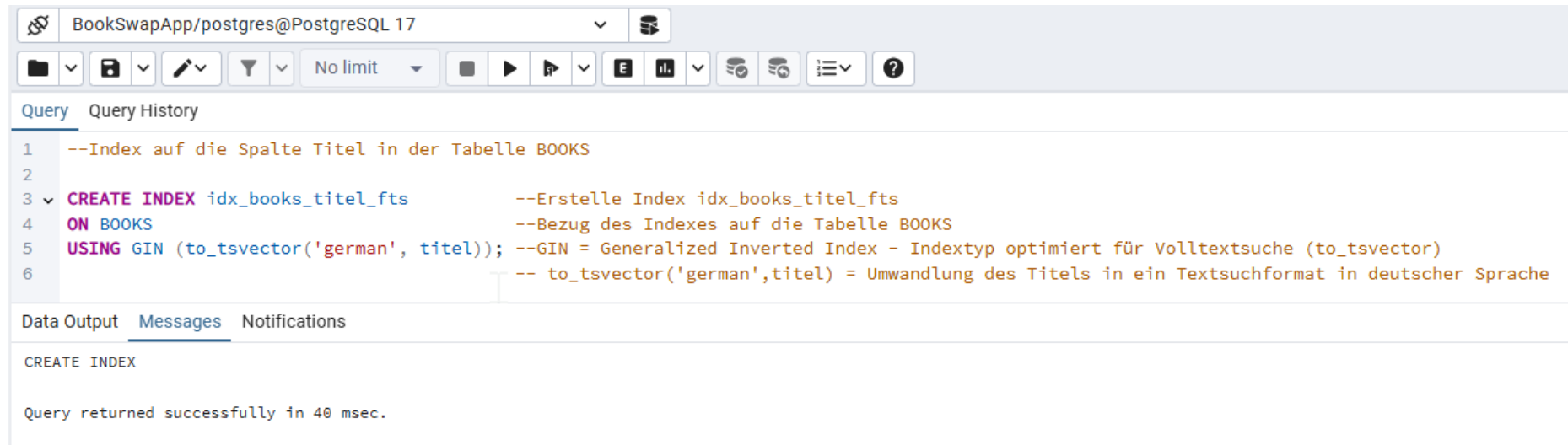
Data Output Messages Notifications

	reservationid [PK] integer	bookid integer	userid integer	reservationfrom timestamp without time zone	reservationuntil timestamp without time zone
1	11	1	1	2024-12-31 23:00:00	2025-01-01 08:00:00

# OPTIMIERUNG FÜR HÄUFIGE ABFRAGEN

## INDEXIERUNG

Um die Performance zu steigern, können für häufige Abfragen geeignete Indexe implementiert werden.



```
BookSwapApp/postgres@PostgreSQL 17
[Icons: Folder, Save, Edit, Filter, No limit, Run, Stop, Refresh, Export, Import, Help]
Query Query History
1  --Index auf die Spalte Titel in der Tabelle BOOKS
2
3  CREATE INDEX idx_books_titel_fts          --Erstelle Index idx_books_titel_fts
4  ON BOOKS                                --Bezug des Indexes auf die Tabelle BOOKS
5  USING GIN (to_tsvector('german', titel)); --GIN = Generalized Inverted Index - Indextyp optimiert für Volltextsuche (to_tsvector)
6                                           -- to_tsvector('german',titel) = Umwandlung des Titels in ein Textsuchformat in deutscher Sprache
Data Output Messages Notifications
CREATE INDEX
Query returned successfully in 40 msec.
```

# INDEXIERUNG MATERIALIZED VIEW

PostgreSQL erstellt für Spalten mit UNIQUE oder PRIMARY KEY die Indexe automatisch. Die JOIN-Performance und Filter verbessern die Indexe auf Fremdschlüsseln. Bei häufig genutzten Filter- und Suchabfragen kann auch eine Materialized View zur Gewinnung der Performance beitragen.

```
BookSwapApp/postgres@PostgreSQL 17

Query Query History
1  --Indexe anlegen-----
2
3  --BOOKS: für häufige Filter nach Autor, Verlag, Kategorie, Sprache, Versand
4  CREATE INDEX idx_books_authorid ON books (authorid);
5  CREATE INDEX idx_books_pubhouseid ON books (pubhouseid);
6  CREATE INDEX idx_books_categoryid ON books (categoryid);
7  CREATE INDEX idx_books_languageid ON books (languageid);
8  CREATE INDEX idx_books_shippingid ON books (shippingid);
9
10 --LOANS: häufig nach UserID und ReturnDate gesucht (z.B. offene Ausleihen)
11 CREATE INDEX idx_loans_userid_returndate ON loans (userid, returndate);
12
13 --RESERVATIONS: Suche nach UserID und BookID häufig sinnvoll
14 CREATE INDEX idx_reservations_userid ON reservations (userid);
15 CREATE INDEX idx_reservations_bookid ON reservations (bookid);
16
17 --ADDRESSES: Suche nach Ort/PLZ oder Verknüpfungen mit User/Verlag
18 CREATE INDEX idx_addresses_place ON addresses (place);
19 CREATE INDEX idx_addresses_zipcode ON addresses (zipcode);
20 CREATE INDEX idx_addresses_userid ON addresses (userid);
21 CREATE INDEX idx_addresses_pubhouseid ON addresses (pubhouseid);

Data Output Messages Notifications
CREATE INDEX
Query returned successfully in 95 msec.
```

```
Query Query History
1  --Diese Materialized View speichert den aktuellen Zustand aller verfügbaren Bücher
2
3  CREATE MATERIALIZED VIEW available_books AS
4  SELECT b.*
5  FROM books b
6  LEFT JOIN loans l ON b.bookid = l.bookid AND (l.returndate IS NULL OR l.returndate > NOW()) --Buch ist ausgeliehen und noch nicht zurückgegeben
7  --oder Rückgabedatum liegt in der Zukunft
8  LEFT JOIN reservations r ON b.bookid = r.bookid
9  AND NOW() BETWEEN r.reservationfrom AND r.reservationuntill --Buch ist reserviert und
10 --die aktuelle Zeit liegt im Reservierungszeitraum
WHERE l.bookid IS NULL AND r.bookid IS NULL; --kein aktiver Leihvorgang und keine aktive Reservierung vorhanden

Data Output Messages Notifications
SELECT 4
Query returned successfully in 59 msec.
```

# KOMPLEXERE ABFRAGE ÜBER MEHRERE TABELLEN

BookSwapApp/postgres@PostgreSQL 17

No limit

Query Query History

```
1  --Komplexere Abfrage über mehrere Tabellen BOOKS, AUTHORS, USERS, SHIPPINGOPTIONS, ADDRESSES,
2  --die noch nicht zurückgegebene Bücher filtert
3  SELECT
4      B.BookID,
5      B.Titel AS BookTitle,
6      CONCAT(A.Firstname, ' ', A.Surname) AS AuthorName,
7      CONCAT(U.Firstname, ' ', U.Surname) AS BorrowerName,
8      L.LoanDate,
9      (L.LoanDate + INTERVAL '1 day' * L.LoanPeriod) AS ExpectedReturnDate, --berechnet erwartete Rückgabe in Tagen
10     S.ShippingOption,
11     S.ShipmentStatus,
12     A2.Street || ' ' || A2.HouseNo || ', ' || A2.ZIPCode || ' ' || A2.Place AS ShippingAddress
13 FROM
14     LOANS L
15 JOIN BOOKS B ON L.BookID = B.BookID
16 JOIN AUTHORS A ON B.AuthorID = A.AuthorID
17 JOIN USERS U ON L.UserID = U.UserID
18 LEFT JOIN SHIPPINGOPTIONS S ON B.ShippingID = S.ShippingID
19 LEFT JOIN ADDRESSES A2 ON S.AddressID = A2.AddressID
20 WHERE
21     L.ReturnDate IS NULL
22 ORDER BY
23     L.LoanDate DESC;
```

Data Output Messages Notifications

SQL

Showing rows: 1 to 7  Page No: 1

	bookid integer	booktitle character varying (255)	authorname text	borrowername text	loandate timestamp without time zone	expectedreturndate timestamp without time zone	shippingoption character varying (100)	shipmentstatus character varying (100)	shippingaddress text
1	9	A Tale of Two Cities	Colleen Hoover	Zeynep Akyol	2025-07-01 09:45:00	2025-07-22 09:45:00	Nur Abholung	[null]	[null]
2	8	Wallenstein	Margaret Atwood	Sarah Baecker	2025-05-30 10:00:00	2025-06-13 10:00:00	Nur Abholung	[null]	[null]
3	7	Mrs Dalloway	Sally Rooney	Anna Burger	2025-04-28 17:30:00	2025-05-05 17:30:00	Nur Abholung	[null]	[null]
4	6	Norwegian Wood	Haruki Murakami	Ali Sari	2025-04-25 13:00:00	2025-05-09 13:00:00	Nur Abholung	[null]	[null]
5	5	Tom Sawyer	Toni Morrison	Julius Keller	2025-04-22 11:15:00	2025-05-02 11:15:00	Mit Versand	Zugestellt	Cassellastrasse 30-32, 60386 Frankfurt am Main
6	4	Krieg und Frieden	William Shakespeare	Petra Schlauer	2025-04-20 16:45:00	2025-05-04 16:45:00	Nur Abholung	[null]	[null]
7	3	Hamlet	Leo Tolstoi	Jana Koncelmann	2025-04-15 09:00:00	2025-05-06 09:00:00	Nur Abholung	[null]	[null]



# METADATEN

- ANZAHL DER TABELLEN = 15
- DATENBANKGRÖÖE = 9051 KB
- GESAMTANZAHL ALLER ZEILEN IN  
DER DATENBANK = 158





# VIELEN DANK

IU Internationale Hochschule

Martina Özkan

Business Intelligence (B.Sc.) | Martikel-Nr. 9227086

[martina.oezkan@iu-study.org](mailto:martina.oezkan@iu-study.org)