SQL запросы

Оглавление

[Создание новой таблицы на основании данных другой таблицы 1](#_Toc198254429)

[Создание таблицы с внешним ключом и описанием действий при удалении 1](#_Toc198254430)

[Создание таблиц со связями, условиями и индексами 2](#_Toc198254431)

[INSERT со вложенным запросом 2](#_Toc198254432)

[LENGTH со вложенным запросом 2](#_Toc198254433)

[Вложенные запросы в операторах соединения 3](#_Toc198254434)

[Запрос на обновление в связанных таблицах 3](#_Toc198254435)

## Создание новой таблицы на основании данных другой таблицы

*Создать новую таблицу****back\_payment****, куда внести информацию о неоплаченных штрафах (Фамилию и инициалы водителя, номер машины, нарушение, сумму штрафа  и  дату нарушения) из таблицы****fine****.*

CREATE TABLE back\_payment as

SELECT

name,

number\_plate,

violation,

sum\_fine,

date\_violation

FROM

fine

WHERE

date\_payment is NULL;

SELECT \* FROM back\_payment

## Создание таблицы с внешним ключом и описанием действий при удалении

*Создать таблицу****book****. Будем считать, что при удалении автора из таблицы****author****, должны удаляться все записи о книгах из таблицы****book****, написанные этим автором. А при удалении жанра из таблицы****genre****для соответствующей записи****book****установить значение****Null****в столбце****genre\_id****.*

CREATE TABLE book (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(50),

author\_id INT NOT NULL,

genre\_id INT,

price DECIMAL(8,2),

amount INT,

FOREIGN KEY (author\_id) REFERENCES author (author\_id) ON DELETE CASCADE,

FOREIGN KEY (genre\_id) REFERENCES genre (genre\_id) ON DELETE SET NULL);

## Создание таблиц со связями, условиями и индексами

CREATE TABLE TRANSACTIONS (

transaction\_id SERIAL PRIMARY KEY,

card\_id INTEGER,

terminal\_id INTEGER,

online\_payment\_id INTEGER,

amount DECIMAL(12, 2) NOT NULL CHECK (amount > 0),

currency VARCHAR(50),

status VARCHAR(20) CHECK (status IN ('В процессе', 'Успешно', 'Отклонена', 'Аннулирована')),

timestamp TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

payment\_method VARCHAR(10) NOT NULL CHECK (payment\_method IN ('terminal', 'website')),

is\_authorized BOOLEAN,

FOREIGN KEY (card\_id) REFERENCES BANK\_CARD(card\_id) ON DELETE SET NULL,

FOREIGN KEY (terminal\_id) REFERENCES PAYMENT\_TERMINAL(terminal\_id) ON DELETE SET NULL,

FOREIGN KEY (online\_payment\_id) REFERENCES ONLINE\_PAYMENT(online\_payment\_id) ON DELETE SET NULL,

CONSTRAINT chk\_payment\_method CHECK (

(payment\_method = 'terminal' AND terminal\_id IS NOT NULL AND online\_payment\_id IS NULL) OR

(payment\_method = 'website' AND online\_payment\_id IS NOT NULL AND terminal\_id IS NULL)

)

);

CREATE INDEX idx\_TRANSACTIONS\_card ON TRANSACTIONS (card\_id);

CREATE INDEX idx\_TRANSACTIONS\_status ON TRANSACTIONS (status);

CREATE INDEX idx\_seller\_tin ON SELLER(tin);

CREATE INDEX idx\_seller\_name ON SELLER(legal\_name);

CREATE INDEX idx\_account\_bic ON SELLER\_ACCOUNT(bic);

## INSERT со вложенным запросом

*Занести из таблицы****supply****в таблицу****book****только те книги, авторов которых нет в****book.***

INSERT INTO book (title, author, price, amount)

SELECT title, author, price, amount

FROM supply

WHERE author NOT IN (

SELECT author

FROM book

);

SELECT \* FROM book;

## LENGTH со вложенным запросом

*Выведите пассажиров с самым длинным ФИО. Пробелы, дефисы и точки считаются частью имени.*

SELECT name FROM Passenger

WHERE

LENGTH(name)=

(SELECT

max(LENGTH(name))

FROM Passenger)

## Вложенные запросы в операторах соединения

*Вывести информацию о книгах (название книги, фамилию и инициалы автора, название жанра, цену и количество экземпляров книги), написанных в самых популярных жанрах, в отсортированном в алфавитном порядке по названию книг виде. Самым популярным считать жанр, общее количество экземпляров книг которого на складе максимально.*

select title, name\_author, name\_genre, price, amount

FROM author

INNER JOIN book ON author.author\_id = book.author\_id

INNER JOIN genre ON book.genre\_id = genre.genre\_id

where genre.genre\_id IN

(SELECT query\_in\_1.genre\_id

FROM

(SELECT genre\_id, SUM(amount) AS sum\_amount

FROM book

GROUP BY genre\_id

) as query\_in\_1

INNER JOIN

(SELECT genre\_id, SUM(amount) AS sum\_amount

FROM book

GROUP BY genre\_id

ORDER BY sum\_amount DESC

LIMIT 1

) as query\_in\_2

ON query\_in\_1.sum\_amount= query\_in\_2.sum\_amount)

order by title asc

## Запрос на обновление в связанных таблицах

*Для книг, которые уже есть на складе (в таблице book), но по другой цене, чем в поставке (supply),  необходимо в таблице book увеличить количество на значение, указанное в поставке,  и пересчитать цену. А в таблице  supply обнулить количество этих книг*

UPDATE book

INNER JOIN author ON author.author\_id = book.author\_id

INNER JOIN supply ON book.title = supply.title

and supply.author = author.name\_author

SET book.amount = book.amount + supply.amount,

supply.amount = 0,

book.price=(book.price\*book.amount+supply.price\*supply.amount)/(book.amount+supply.amount)

WHERE book.price <> supply.price;

SELECT \* FROM book;

SELECT \* FROM supply;