**ТЕМА 5. DDL СОЗДАНИЕ БД, ТАБЛИЦ И ИХ МОДИФИКАЦИЯ**

1. **Управление таблицами: создание, изменение, удаление**

Создание таблицы

Для создания таблицы в PostgreSQL используется оператор CREATE TABLE. Вот пример создания таблицы с несколькими полями:

CREATE TABLE employees (

employee\_id serial PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

hire\_date DATE

);

В этом примере создана таблица "employees" с четырьмя столбцами: "employee\_id", "first\_name", "last\_name" и "hire\_date". Столбец "employee\_id" определен как серийный (serial), что означает, что PostgreSQL автоматически будет увеличивать его значение при вставке новых записей, и он будет использоваться как первичный ключ.

Изменение таблицы

Для изменения структуры существующей таблицы в PostgreSQL используется оператор ALTER TABLE. Вот пример добавления нового столбца к таблице:

ALTER TABLE employees

ADD email VARCHAR(100);

Этот запрос добавляет столбец "email" к существующей таблице "employees".

Удаление таблицы

Чтобы удалить таблицу, используется оператор DROP TABLE. Например:

DROP TABLE employees;

Этот запрос удалит таблицу "employees" и все ее данные.

1. **Создание первичного ключа (PRIMARY KEY)**

Первичный ключ (PRIMARY KEY) используется для уникальной идентификации записей в таблице. Для создания первичного ключа, можно добавить ключевое слово PRIMARY KEY к одному или нескольким столбцам при создании таблицы. Пример:

DROP TABLE chair;

CREATE TABLE chair

(

cathedra\_id serial PRIMARY KEY,

chair\_name varchar,

dean varchar

);

INSERT INTO chair

VALUES (1, 'name', 'dean');

--no duplicates

INSERT INTO chair

VALUES (1, 'name', 'dean');

--no NULLs

INSERT INTO chair

VALUES (NULL, 'name', 'dean');

--only UNIQUE NOT NULLs

INSERT INTO chair

VALUES (2, 'name', 'dean');

--equivalent (almost) to:

DROP TABLE chair;

CREATE TABLE chair

(

cathedra\_id serial UNIQUE NOT NULL,

chair\_name varchar,

dean varchar

);

ALTER TABLE chair

DROP CONSTRAINT chair\_cathedra\_id\_key

select constraint\_name

from information\_schema.key\_column\_usage

where table\_name = 'chair'

and table\_schema = 'public'

and column\_name = 'cathedra\_id';

ALTER TABLE chair

ADD PRIMARY KEY(chair\_id);

1. **Создание внешнего ключа (FOREIGN KEY)**

Внешний ключ (FOREIGN KEY) используется для связи двух таблиц. Для создания внешнего ключа, нужно указать столбец, который ссылается на столбец в другой таблице. Пример:

CREATE TABLE public.publisher

(

publisher\_id integer NOT NULL,

publisher\_name varchar(128) NOT NULL,

address text NOT NULL,

CONSTRAINT PK\_publisher\_id PRIMARY KEY(publisher\_id)

);

CREATE TABLE public.book

(

book\_id integer NOT NULL,

title text NOT NULL,

isbn character varying(32) NOT NULL,

publisher\_id integer NOT NULL,

CONSTRAINT PK\_book\_book\_id PRIMARY KEY(book\_id)

);

--добавим немного данных в publisher

INSERT INTO publisher

VALUES

(1, 'Everyman''s Library', 'NY'),

(2, 'Oxford University Press', 'NY'),

(3, 'Grand Central Publishing', 'Washington'),

(4, 'Simon & Schuster', 'Chicago');

--без FK мы можем пихать любые значения

INSERT INTO book

VALUES

(1, 'The Diary of a Young Girl', '0199535566', 10); -- Everyman's Library

SELECT \*

FROM book;

TRUNCATE TABLE book;

ALTER TABLE book

ADD CONSTRAINT fk\_books\_publisher FOREIGN KEY(publisher\_id) REFERENCES publisher(publisher\_id);

-- теперь всякую дичь на вставить

insert into book

values

(1, 'The Diary of a Young Girl', '0199535566', 10);

--если хотим сразу задать ограничение при создании таблицы

DROP TABLE book;

CREATE TABLE public.book

(

book\_id integer NOT NULL,

title text NOT NULL,

isbn character varying(32) NOT NULL,

publisher\_id integer NOT NULL,

CONSTRAINT PK\_book\_book\_id PRIMARY KEY(book\_id),

CONSTRAINT FK\_book\_publisher FOREIGN KEY (publisher\_id) REFERENCES publisher(publisher\_id)

);

--если хотим удалить ограничение

ALTER TABLE book

DROP CONSTRAINT FK\_book\_publisher;

1. **Ограничение CHECK**

Ограничение CHECK позволяет задать условие, которое должно выполняться для значений в столбце. Например, можно создать ограничение CHECK, чтобы гарантировать, что возраст сотрудника всегда положителен:

DROP TABLE IF EXISTS book;

CREATE TABLE public.book

(

book\_id integer NOT NULL,

title text NOT NULL,

isbn character varying(32) NOT NULL,

publisher\_id integer NOT NULL,

CONSTRAINT PK\_book\_book\_id PRIMARY KEY(book\_id)

);

ALTER TABLE book

ADD COLUMN price decimal CONSTRAINT CHK\_book\_price CHECK (price > 0);

INSERT INTO book

VALUES

(1, 'title', 'isbn', 1, -1.5);

1. **DEFAULT - значения по умолчанию**

Можно указать значения по умолчанию для столбцов, которые будут использоваться, если не указано другое значение при вставке. Пример:

CREATE TABLE customer

(

customer\_id serial,

full\_name text,

status char DEFAULT 'r',

CONSTRAINT PK\_customer\_id PRIMARY KEY(customer\_id),

CONSTRAINT CHK\_customer\_status CHECK (status = 'r' or status = 'p')

);

INSERT INTO customer

VALUES

(1, 'name');

SELECT \*

FROM customer;

INSERT INTO customer

VALUES

(1, 'name', 'd');

ALTER TABLE customer

ALTER COLUMN status DROP DEFAULT;

ALTER TABLE customer

ALTER COLUMN status SET DEFAULT 'r';

1. **Последовательности в PostgreSQL - CREATE SEQUENCE**

Последовательности (sequences) используются для автоматической генерации уникальных числовых значений, как, например, для столбцов с типом данных serial. Для создания последовательности используется оператор CREATE SEQUENCE. Пример:

CREATE SEQUENCE seq;

SELECT nextval('seq');

SELECT currval('seq');

SELECT lastval();

--

SELECT setval('seq', 10);

SELECT currval('seq');

SELECT nextval('seq');

SELECT setval('seq', 16, false);

SELECT currval('seq');

SELECT nextval('seq');

--

CREATE SEQUENCE IF NOT EXISTS seq2 INCREMENT 16;

SELECT nextval('seq2');

--

CREATE SEQUENCE IF NOT EXISTS seq3

INCREMENT 16

MINVALUE 0

MAXVALUE 128

START WITH 0

SELECT nextval('seq3');

ALTER SEQUENCE seq3 RENAME TO seq4

ALTER SEQUENCE seq4 RESTART WITH 16

SELECT nextval('seq4');

DROP SEQUENCE seq4;

DROP TABLE IF EXISTS book;

CREATE TABLE public.book

(

book\_id int NOT NULL,

title text NOT NULL,

isbn varchar(32) NOT NULL,

publisher\_id int NOT NULL,

CONSTRAINT PK\_book\_book\_id PRIMARY KEY(book\_id)

);

CREATE SEQUENCE IF NOT EXISTS book\_book\_id\_seq

START WITH 1 OWNED BY book.book\_id;

-- doesn't work

INSERT INTO book (title, isbn, publisher\_id)

VALUES ('title', 'isbn', 3);

--we need to set default

ALTER TABLE book

ALTER COLUMN book\_id SET DEFAULT nextval('book\_book\_id\_seq');

--now should work

INSERT INTO book (title, isbn, publisher\_id)

VALUES ('title', 'isbn', 3);

SELECT \*

FROM book;

INSERT INTO book (title, isbn, publisher\_id)

VALUES ('title2', 'isbn2', 3);

SELECT \*

FROM book;

--BTW, we need the list of columns if we skip some of them inserting values

--so the following doesn't work

INSERT INTO book

VALUES ('title2', 'isbn2', 3);

--returning

INSERT INTO book (title, isbn, publisher\_id)

VALUES ('title3', 'isbn3', 3)

RETURNING book\_id;

1. **Последовательности и таблицы - SERIAL**

Тип данных SERIAL в PostgreSQL используется для создания столбцов, автоматически заполняемых уникальными числовыми значениями. Он использует последовательности (sequences) под капотом. Пример:

CREATE TABLE employees (

employee\_id serial PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50)

);

В этом примере "employee\_id" будет автоматически заполняться уникальными значениями при вставке новых записей.

1. **INSERT - Вставка данных в таблицу**

Чтобы вставить данные в таблицу, используется оператор INSERT INTO. Пример:

INSERT INTO author

VALUES (10, 'John Silver', 4.5);

SELECT \* FROM author;

INSERT INTO author(author\_id, full\_name)

VALUES

(12, 'Name 1'),

(13, 'Name 2'),

(14, 'Name 3');

SELECT \*

INTO best\_authors

FROM author

WHERE rating > 4.5;

SELECT \* FROM best\_authors;

1. **UPDATE, DELETE, RETURNING - обновление и удаление данных из таблицы**

Для обновления данных в таблице используется оператор UPDATE, а для удаления данных - оператор DELETE.

Пример обновления данных:

SELECT \* FROM author;

UPDATE author

SET full\_name = 'Elias', rating = 5

WHERE author\_id = 1;

DELETE FROM author

WHERE rating < 4.5;

DELETE FROM author;

TRUNCATE TABLE author;

DROP TABLE book;

CREATE TABLE book

(

book\_id serial,

title text NOT NULL,

isbn varchar(32) NOT NULL,

publisher\_id int NOT NULL,

CONSTRAINT PK\_book\_book\_id PRIMARY KEY(book\_id)

);

INSERT INTO book(title, isbn, publisher\_id) VALUES ('title', 'isbn', 3)

RETURNING \*;

UPDATE author

SET full\_name = 'Walter', rating = 5

WHERE author\_id = 1

RETURNING author\_id;

DELETE FROM author

WHERE rating = 5

RETURNING \*;