

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Задание №2

по дисциплине «Расширенный курс по
базам данных»

Студент

Закиров Р.Р.

Группа АС-20-1

Руководитель

Мирошников А.И.

Липецк 2023 г.

Создать таблицы, содержание поля типов:

1. Стандартные типы данных (Числовой, символьный, логический, дата/время)

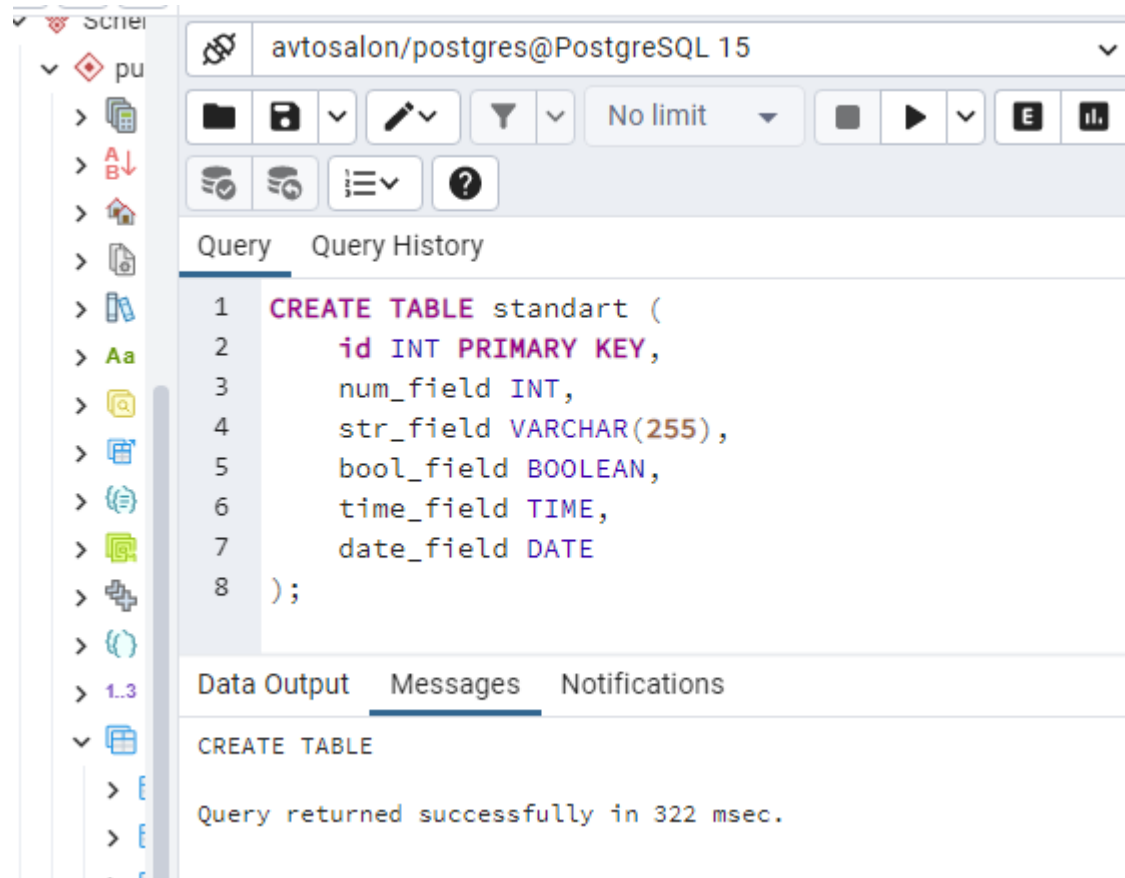


Рис.1. Стандартные типы данных

INSERT и SELECT:

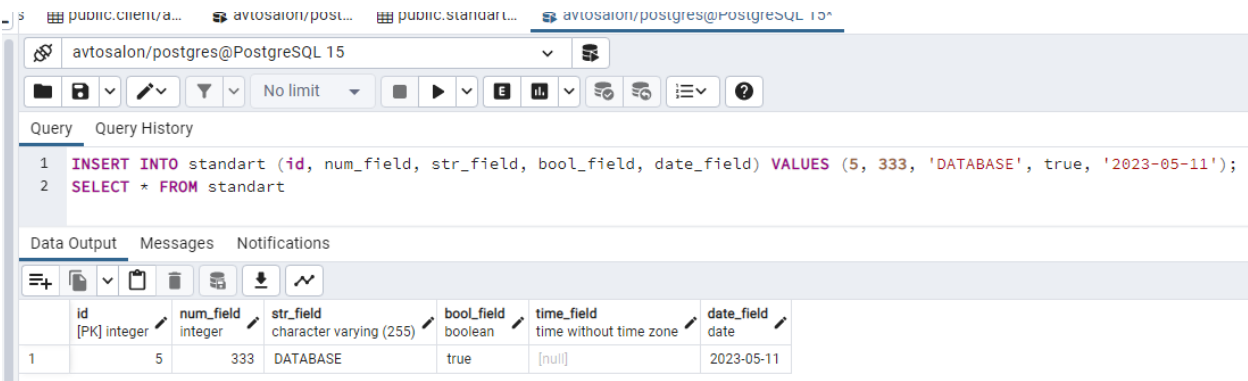


Рис.2. INSERT и SELECT

2. Перечисления

В SQL для создания перечислений можно использовать тип данных ENUM. Пример создания таблицы с полем типа ENUM:

✓ MySQL вернула пустой результат (т.е. ноль строк). (Запрос занял 0,0082 сек.)

```
CREATE TABLE exampl ( id INT PRIMARY KEY, color ENUM('red', 'green', 'blue') );
```

Рис.3. Перечисления

В данном примере создается таблица "exampl" с двумя полями: "id" типа INT и "color" типа ENUM, который может принимать только одно из трех значений: "red", "green" или "blue".

INSERT и SELECT:

✓ Добавлена 1 строка. (Запрос занял 0,0025 сек.)

```
INSERT INTO exampl (id, color) VALUES (1, 'red');
```

[[Построчное редактирование](#)] [[Изменить](#)] [[Создать PHP-код](#)]

✓ Отображение строк 0 - 0 (1 всего, Запрос занял 0,0003 сек.)

```
SELECT * FROM exampl WHERE color = 'red';
```

☐ Профилирование

[[Построчное редактирование](#)] [[Изменить](#)] [[Анализ SQL запроса](#)] [[Создать PHP-код](#)] [[Обновить](#)]

☐ Показать все

Количество строк: 50 ▾

Фильтровать строки:

Extra options

←T→	id	color
<input type="checkbox"/>   	1	red

Рис.4. INSERT и SELECT

3. Массивы

SQL не поддерживает массивы в структуре таблиц. Вместо этого, можно создать отдельную таблицу для хранения значений массива и связать ее с основной таблицей через внешний ключ. Например:

✓ MySQL вернула пустой результат (т.е. ноль строк). (Запрос занял 0,0080 сек.)

```
CREATE TABLE uset ( id INT PRIMARY KEY, name VARCHAR(50), gender ENUM('male', 'female'), status ENUM('active', 'inactive') );
```

[[Построчное редактирование](#)] [[Изменить](#)] [[Создать PHP-код](#)]

✓ MySQL вернула пустой результат (т.е. ноль строк). (Запрос занял 0,0091 сек.)

```
CREATE TABLE usethobbies ( userid INT, hobby VARCHAR(50), PRIMARY KEY (userid, hobby), FOREIGN KEY (userid) REFERENCES uset(id) );
```

[[Построчное редактирование](#)] [[Изменить](#)] [[Создать PHP-код](#)]

Рис.5. Массивы

В этом примере таблица usethobbies содержит хобби для каждого пользователя. Столбец userid является внешним ключом, который ссылается на столбец id таблицы uset. Ключевое слово PRIMARY KEY определяет состав первичного ключа таблицы usethobbies, который состоит из столбцов userid и hobby.

INSERT и SELECT:

✓ Добавлена 1 строка. (Запрос занял 0,0112 сек.)

```
INSERT INTO uset (id, name, gender, status) VALUES (1, 'John', 'male', 'active');
```

[[Построчное редактирование](#)] [[Изменить](#)] [[Создать PHP-код](#)]

✓ MySQL вернула пустой результат (т.е. ноль строк). (Запрос занял 0,0003 сек.)

```
SELECT * FROM uset WHERE gender = 'female';
```

☐ Профилирование

[[Построчное редактирование](#)] [[Изменить](#)] [[Анализ SQL запроса](#)] [[Создать PHP-код](#)] [[Обновить](#)]

id	name	gender	status
----	------	--------	--------

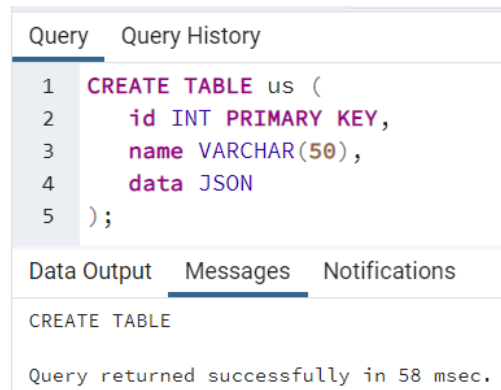
✓ Добавлена 1 строка. (Запрос занял 0,0020 сек.)

```
INSERT INTO usethobbies (userid, hobby) VALUES (1, 'reading');
```

Рис.6-7. INSERT и SELECT

4. XML и JSON

SQL не поддерживает нативную работу с XML и JSON, но существуют специальные типы данных и функции для работы с ними в различных СУБД. Например, в PostgreSQL можно использовать тип данных JSON:



```
Query    Query History
1 CREATE TABLE us (
2     id INT PRIMARY KEY,
3     name VARCHAR(50),
4     data JSON
5 );
```

Data Output Messages Notifications


CREATE TABLE

Query returned successfully in 58 msec.

Рис.8. XML и JSON

В этом примере таблица us содержит столбец data типа JSON, который может хранить данные в формате JSON. Для работы с данным столбцом можно использовать функции JSONEXTRACT и JSONSET.

INSERT и SELECT:



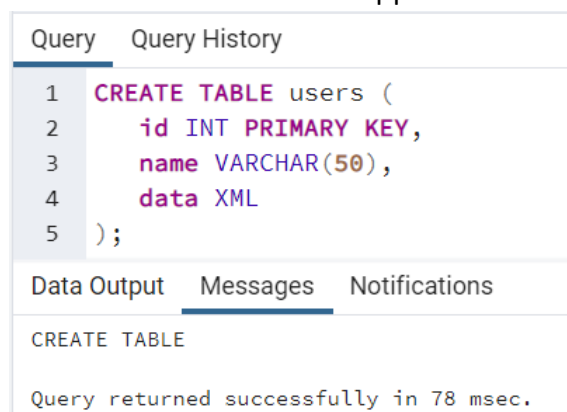
```
Query    Query History
1 INSERT INTO us (id, name, data) VALUES (2, 'John', '{"age": 30, "city": "New York"}');
2 SELECT * FROM us WHERE data->>'city' = 'New York';
```

Data Output Messages Notifications

	id [PK] integer	name character varying (50)	data json
1	2	John	{"age": 30, "city": "New York"}

Рис.9. INSERT и SELECT

В PostgreSQL можно использовать тип данных XML:



```
Query    Query History
1 CREATE TABLE users (
2     id INT PRIMARY KEY,
3     name VARCHAR(50),
4     data XML
5 );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 78 msec.

Рис.10. Создание таблицы с XML

В этом примере таблица users содержит столбец data типа XML, который может хранить данные в формате XML. Для работы с данным столбцом можно использовать функции XPath и XQuery.

INSERT и SELECT:

Query		Query History	
1	INSERT INTO	users (id, name, data)	VALUES (1, 'John', '<user><age>30</age><gender>Male</gender></user>');
2	SELECT	* FROM users	WHERE id = 1;
Data Output			
Messages			
Notifications			
	id	name	data
	[PK] integer	character varying (50)	xml
1	1	John	<user><age>30</age><gender>Male</gender></user>

Рис.11. INSERT и SELECT

5. Составные типы

В SQL можно создавать составные типы данных, которые представляют собой набор полей различных типов. Обычно такие типы используются для описания сложных объектов, например, заказов или платежей.

Для создания составного типа в SQL используется оператор CREATE TYPE, за которым следует имя типа и список полей соответствующих типов. Например, в PostgreSQL можно создать тип данных ord:

Query		Query History	
1	CREATE TYPE	ord AS (
2		id INT,	
3		customer VARCHAR(50),	
4		items TEXT,	
5		total NUMERIC	
6);	
Data Output			
Messages			
Notifications			
CREATE TYPE			
Query returned successfully in 91 msec.			

Рис.12. Создание типа ord

В этом примере тип ord содержит четыре поля: id типа INT, customer типа VARCHAR(50), items типа TEXT (массив строк) и total типа NUMERIC.

После создания типа его можно использовать для создания таблиц, например:

```
Query  Query History
1  CREATE TABLE ordes (
2      id SERIAL PRIMARY KEY,
3      orderdata ord
4  );

Data Output  Messages  Notifications

CREATE TABLE

Query returned successfully in 58 msec.
```

Рис.13. Пример создания таблицы с типом order

В этом примере таблица ordes содержит столбец orderdata типа ord, который может хранить данные о заказе.

INSERT и SELECT:

```
Query  Query History
1  INSERT INTO orders (orderdata) VALUES (1, 'John', 'apple, banana, orange', 5.99);
2  SELECT * FROM orders WHERE orderdata->'customer' = 'John';
```

Рис.14. INSERT и SELECT

6. Прочие типы: денежный, двоичный, геометрический, битовые строки, UUID

Денежный тип:

```
Query  Query History
1  CREATE TABLE transactions (
2      id SERIAL PRIMARY KEY,
3      amount MONEY,
4      date DATE
5  );

Data Output  Messages  Notifications

CREATE TABLE

Query returned successfully in 294 msec.
```

Рис.15. Денежный тип

В этом примере таблица transactions содержит столбец amount типа MONEY, который может хранить данные о сумме транзакции.

INSERT и SELECT:

Query

Query History

1

INSERT INTO transactions (amount, date) VALUES (1000.50, '2021-09-15');

2

SELECT * FROM transactions WHERE id > 1;

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑

🗄

⬇

📈

	id [PK] integer	amount money	date date
1	2	1 000,50 ?	2021-09-15

Рис.16. Денежный тип

Двоичный тип:

Query	Query History
1 2 3 4	<pre>CREATE TABLE images (id SERIAL PRIMARY KEY, imagedata BYTEA);</pre>
Data Output	Messages
CREATE TABLE	
Query returned successfully in 144 msec.	

Рис.17. Двоичный тип

INSERT и SELECT:

Query

Query History

1

INSERT INTO images (imagedata) VALUES ('\\x89504E470D0A1A0A0000000D49484452000002880000012F080200000907C8A730000017352474200AECE1

2

CE90000000467414D410000B18F0BFC6105000000097048597300000EC300000EC301C76FA8640000001974455

3

874536F6674776172650041646F6265204669726566616F7820322E303100866951DC0000004674524E5300AE4

4

26082'::bytea);

5

SELECT * FROM images;

Data Output

Messages

Notifications

id

[PK] integer

imagedata

bytea

1

1

[binary data]

Рис.18. INSERT и SELECT

В этом примере таблица images содержит столбец imagedata типа BYTEA, который может хранить бинарные данные, например, изображения.

Геометрический тип:

```
Query  Query History
1  CREATE TABLE locations (
2      id SERIAL PRIMARY KEY,
3      name VARCHAR(50),
4      coordinates POINT
5  );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 207 msec.

Рис.19. Геометрический тип

В этом примере таблица locations содержит столбец coordinates типа POINT, который может хранить координаты местоположения.

INSERT и SELECT:

```
Query  Query History
1  INSERT INTO locations (name, coordinates) VALUES ('Central Park', POINT(40.7829, -73.9654));
2  SELECT * FROM locations WHERE name = 'Central Park';
```

Data Output Messages Notifications

	id [PK] integer	name character varying (50)	coordinates point
1	1	Central Park	(40.7829,-73.9654)

Рис.20. INSERT и SELECT

Битовые строки:

```
Query  Query History
1  CREATE TABLE flags (
2      id SERIAL PRIMARY KEY,
3      flagdata BIT(8)
4  );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 165 msec.

Рис.21. Битовые строки

В этом примере таблица flags содержит столбец flagdata типа BIT(8), который может хранить битовые флаги.

INSERT и SELECT:

Query	Query History
1	<code>INSERT INTO flags (flagdata) VALUES (B'10101010');</code>
2	<code>SELECT * FROM flags WHERE flagdata = B'10101010';</code>

Data Output	Messages	Notifications						
<div><div><div>≡</div><div>+</div></div><div><div>📄</div><div>▼</div></div><div><div>📋</div><div>🗑️</div></div><div><div>🗄️</div><div>⬇️</div></div><div><div>📈</div></div></div>								
<table><thead><tr><th></th><th>id [PK] integer</th><th>flagdata bit</th></tr></thead><tbody><tr><td>1</td><td></td><td>10101010</td></tr></tbody></table>		id [PK] integer	flagdata bit	1		10101010		
	id [PK] integer	flagdata bit						
1		10101010						

Рис.22. INSERT и SELECT

UUID:

Query	Query History
1	<code>CREATE TABLE example_table (</code>
2	<code>id UUID PRIMARY KEY,</code>
3	<code>name VARCHAR(50),</code>
4	<code>description TEXT</code>
5	<code>);</code>

Data Output	Messages	Notifications
	CREATE TABLE	
	Query returned successfully in 138 msec.	

Рис.23. UUID

В этом примере мы создаем таблицу `example_table` с тремя столбцами: `id`, `name` и `description`. Столбец `id` имеет тип данных `UUID` и является первичным ключом таблицы. Остальные столбцы имеют обычные типы данных `VARCHAR` и `TEXT`.

Пример запроса INSERT:

Query	Query History
1	<code>INSERT INTO example_table (id, name) VALUES ('a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11', 'John');</code>

Data Output	Messages	Notifications
	INSERT 0 1	
	Query returned successfully in 123 msec.	

Рис.24. Пример запроса INSERT

Здесь мы добавляем новую запись в таблицу `example_table` с указанием значения `UUID` в столбце `id` и имени "John" в столбце `name`.

Пример запроса SELECT с условием:

The screenshot shows a database query interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `1 SELECT * FROM example_table WHERE id = 'a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11';`. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table of results. Above the table is a toolbar with icons for various actions like expand, save, undo, redo, delete, refresh, and zoom. The table has four columns: 'id' (PK, uuid), 'name' (character varying (50)), 'description' (text), and an unnamed column. The first row of data shows the value 'a0eebc99...' for 'id', 'John' for 'name', and '[null]' for 'description'.

	id [PK] uuid	name character varying (50)	description text
1	a0eebc99...	John	[null]

Рис.25. Пример запроса SELECT с условием

Здесь мы выбираем все записи из таблицы `example_table`, где значение столбца `id` равно указанному UUID.