**Лабораторная работа №3**

**Тема:** СУБД PostgreSQL

**Цель:** познакомиться с интерфейсом взаимодействия с PostgreSQL, а также научиться применять некоторые нетривиальные возможности СУБД.

**Теоретический базис**

**Знакомство с PostgreSQL**

PostgreSQL — объектно-реляционная система управления базами данных (ОРСУБД), разработка которой в различных формах ведется с 1977 года. Работа началась с проекта Ingres в Калифорнийском университете (Беркли). Затем проект Ingres был переведен на коммерческую разработку в корпорации Relational Technologies/Ingres.

**Создание объектно-реляционных связей**

В PostgreSQL поддерживается механизм создания объектно-реляционных связей, называемый наследованием. Таблица может наследовать некоторые атрибуты своих полей от одной или нескольких других таблиц, что приводит к созданию отношений типа «предок—потомок». В результате производные таблицы («потомки») обладают теми же полями и ограничениями, что и их базовые таблицы («предки»), а также дополняются собственными полями.

Производная таблица создается командой SQL CREATE TABLE, в которую включается секция INHERITS. Секция состоит из ключевого слова INHERITS и имени базовой таблицы (или нескольких таблиц).

CREATE TABLE производная\_таблица определение

INHERITS ( базовая\_таблица [, ...] )

**Ограничения в таблицах**

Ограничение (constraint) представляет собой особый атрибут таблицы, который устанавливает критерии допустимости для содержимого ее полей. Соблюдение этих правил помогает предотвратить заполнение базы ошибочными или неподходящими данными.

Формат установки ограничений следующий:

[ CONSTRAINT ограничение ]

{ NOT NULL | UNIQUE | PRIMARY KEY | DEFAULT значение | CHECK ( условие ) |

REFERENCES таблица [ ( поле ) ]

[ MATCH FULL | MATCH PARTIAL ]

[ ON DELETE операция ]

[ ON UPDATE операция ]

[ DEFERRABLE | NOT DEFERRABLE ]

[ INITIALLY DEFERRED | INITIALLY IMMEDIATE ] }

Существует шесть типов ограничений полей, задаваемых при помощи специальных ключевых слов:

• NOT NULL. Поле не может содержать псевдозначение NULL. Ограничение NOT NULL эквивалентно ограничению CHECK (поле NOT NULL).

• UNIQUE. Поле не может содержать повторяющиеся значения. Следует учитывать, что ограничение UNIQUE допускает многократное вхождение псевдозначений NULL, поскольку формально NULL не совпадает ни с каким другим значением.

• PRIMARY KEY. Автоматически устанавливает ограничения UNIQUE и NOT NULL, а для заданного поля создается индекс. В таблице может устанавливаться только одно ограничение первичного ключа.

• DEFAULT значение. Пропущенные значения поля заменяются заданной величиной. Значение по умолчанию должно относиться к типу данных, соответствующему типу поля. В PostgreSQL 7.1.х значение по умолчанию не может задаваться при помощи подзапроса.

• CHECK условие. Команда INSERT или UPDATE для записи завершается успешно лишь при выполнении заданного условия (выражения, возвращающего логический результат). При установке ограничения поля в секции CHECK может использоваться только поле, для которого устанавливается ограничение.

• REFERENCES. Это ограничение состоит из нескольких секций, которые перечислены ниже.

о REFERENCES таблица [ ( поле ) ]. Входные значения ограничиваемого поля сравниваются со значениями другого поля в заданной таблице. Если совпадения отсутствуют, команда INSERT или UPDATE завершается неудачей. Если параметр поле не указан, проверка выполняется по первичному ключу. Ограничение REFERENCES похоже на ограничение таблицы FOREIGN KEY, описанное в следующем пункте этого подраздела. Действительно, между этими ограничениями есть много общего.

о ON DELETE операция. При выполнении команды DELETE для заданной таблицы с ограничиваемым полем выполняется одна из следующих операций: NО ACTION (если удаление приводит к нарушению целостности ссылок, происходит ошибка; используется по умолчанию, если операция не указана), RESTRICT (аналогично NО ACTION), CASCADE (удаление всех записей, содержащих ссылки на удаляемую запись), SET NULL (поля, содержащие ссылки на удаляемую запись, заменяются псевдозначениями NULL), SET DEFAULT (полям, содержащим ссылки на удаляемую запись, присваивается значение по умолчанию).

о ON UPDATE операция. При выполнении команды UPDATE для заданной таблицы выполняется одна из операций, описанных выше. По умолчанию используется значение NО ACTION. Если выбрана операция CASCADE, все записи, содержащие ссылки на обновляемую запись, обновляются новым значением (вместо удаления, как в случае с ON DELETE CASCADE).

В ограничениях таблиц, в отличие от ограничений полей, могут участвовать сразу несколько полей таблицы. Синтаксис ограничения таблицы:

[ CONSTRAINT ограничение ]

{ UNIQUE ( поле [....]) |

PRIMARY KEY ( поле [. ...] ) |

CHECK ( условие ) |

FOREIGN KEY ( поле [. ... ] )

REFERENCES таблица [ ( поле [. ] ) ]

[ MATCH FULL | MATCH PARTIAL ]

[ ON DELETE операция ]

[ ON UPDATE операция ]

[ DEFERRABLE | NOT DEFERRABLE ]

[ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

**Массивы**

Поля данных PostgreSQL вместо отдельных величин могут содержать конструкции, называемые массивами. Массив сам по себе не является самостоятельным типом данных, а лишь расширяет любой другой тип данных PostgreSQL.

Чтобы создать простейшее поле-массив, включите в команду CREATE TABLE пару квадратных скобок после имени поля. Квадратные скобки показывают, что вместо одного значения в поле может храниться массив указанного типа. Дополнительные квадратные скобки определяют многомерные массивы, то есть «массивы массивов».

В PostgreSQL предусмотрен специальный синтаксис вставки нескольких значений в одно поле. Этот синтаксис основан на определении массивов-констант.

Обобщенная форма массива-константы выглядит так:

'{ "текст" [, ...] }' -- массив строк

'{ число [, ...]}' -- числовой массив

Чтобы узнать количество значений, хранящихся в массиве, следует воспользоваться функцией array\_dims().

**Последовательности**

Последовательностью (sequence) в PostgreSQL называется объект базы данных, который фактически представляет собой автоматически увеличивающееся число. В других СУБД последовательности часто называются счетчиками. Последовательность определяется текущим числовым значением и набором характеристик, определяющих алгоритм автоматического увеличения (или уменьшения) используемых данных.

Наряду с текущим значением в определение последовательности также включается минимальное значение, максимальное значение и приращение. Обычно приращение равно 1, но оно также может быть любым целым числом.

На практике последовательности не рассчитаны на прямой доступ из программы. Работа с ними осуществляется через специальные функции PostgreSQL, предназначенные для увеличения, присваивания или получения текущего значения последовательности.

Последовательности создаются командой SQL CREATE SEQUENCE с положительным или отрицательным приращением. В этом определении единственный обязательный параметр последовательность определяет имя создаваемой последовательности. Значения последовательности представляются типом integer, поэтому максимальное и минимальное

значения должны лежать в интервале от 2 147 483 647 до -2 147 483 647. Синтаксис команды CREATE SEQUENCE:

CREATE SEQUENCE последовательность

[ INCREMENT приращение ]

[ MINVALUE минимум ]

[ MAXVALUE максимум ]

[ START начало ]

[ CACHE кэш ]

[ CYCLE ]

•INCREMENT приращение. Числовое изменение текущего значения последовательности. Отрицательное приращение создает убывающую последовательность. По умолчанию приращение равно 1.

•MINVALUE минимум. Минимальное допустимое значение последовательности.

•MAXVALUE максимум. Максимальное допустимое значение последовательности.

•START начало. Начальное значение последовательности, которым является любое целое число в интервале между минимальным и максимальным значениями.

•CACHE кэш. Возможность предварительного вычисления и хранения значений последовательности в памяти. Кэширование ускоряет доступ к часто используемым последовательностям. Минимальное значение, заданное по умолчанию, равно 1; увеличение объема кэша приводит к увеличению числа кэшируемых значений.

•CYCLE. При достижении нижнего или верхнего порога последовательность продолжает генерировать новые значения. В этом случае она переходит к минимальному значению (для возрастающих последовательностей) или к максимальному значению (для убывающих последовательностей).

К последовательности можно обратиться командой SELECT, как к таблице или представлению (хотя такая возможность используется относительно редко). При составлении запроса к последовательности в списке выборки вместо полей указываются атрибуты последовательности.

Операции с последовательностями:

•nextval ('последовательность' )– увеличивает текущее значение последовательности и возвращает новое;

•currval (' последовательность') – возвращает значение, полученное при последнем вызове nextval. Если в текущем сеансе nextval не вызывалась, то currval не сможет вернуть значение;

•setval ('последовательность', n) – присваивает текущее значение последовательности, следующий вызов nextval вернет значение n+приращение;

•setval (' последовательность', n, b) – также присваивает текущее значение последовательности. Если третий параметр (b) равен false, то следующий вызов nextval вернет n, иначе – n+преращение.

**Задание на лабораторную работу**

1. Создание таблицы:

CREATE TABLE auto

(

inv integer NOT NULL,

mark text,

year\_m text,

CONSTRAINT auto\_pkey PRIMARY KEY (inv)

);

INSERT INTO auto(inv, mark, year\_m) VALUES (1, 'tesla', '2016');

INSERT INTO auto(inv, mark, year\_m) VALUES (2, 'opel', '2001');

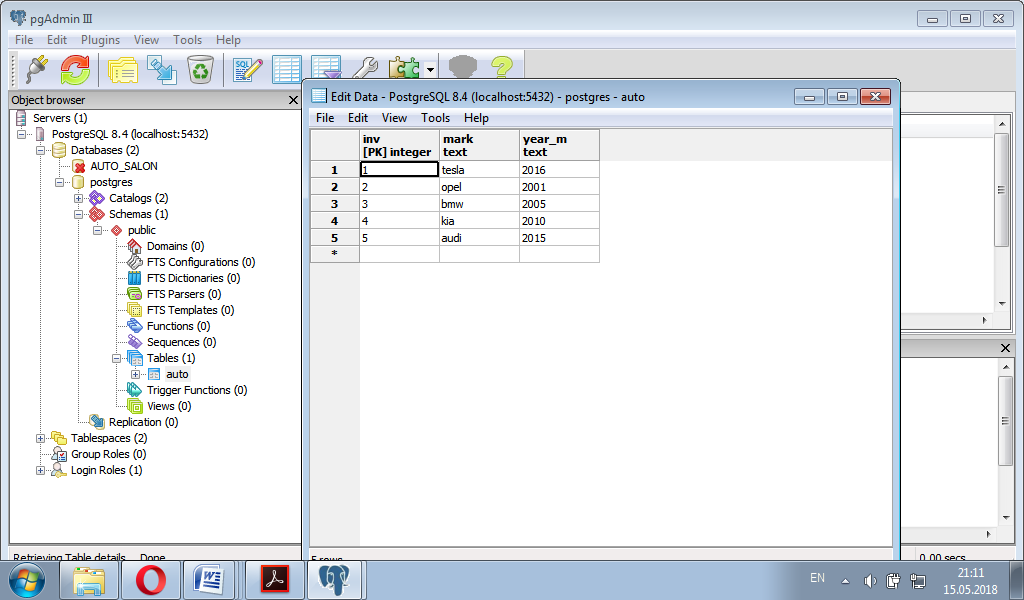


Рис. 1. Создание таблицы auto

2. Создание объектно-реляционных связей:

CREATE TABLE auto\_color

(

-- Inherited from table auto\_color: inv integer NOT NULL,

-- Inherited from table auto\_color: mark text,

-- Inherited from table auto\_color: year\_m text,

color text,

CONSTRAINT auto\_color\_pkey PRIMARY KEY (inv));

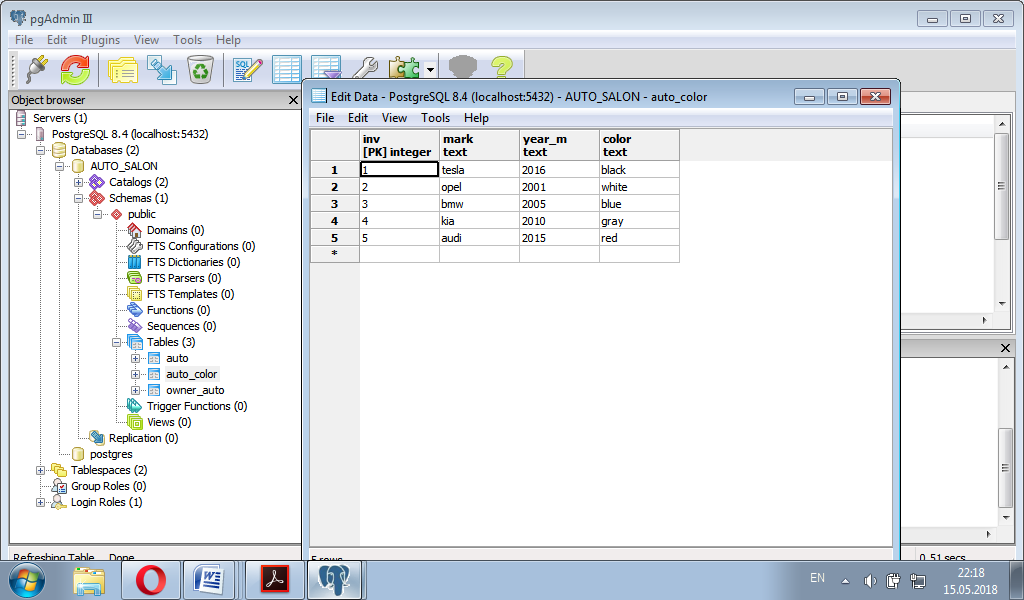


Рис. 2. Создание объектно-реляционных связей

3. Использование массивов:

CREATE TABLE new\_auto

(

id integer,

auto\_mark text[]

);

INSERT INTO new\_auto(id, auto\_mark) VALUES (1, '{audi, tesla, bmw}');

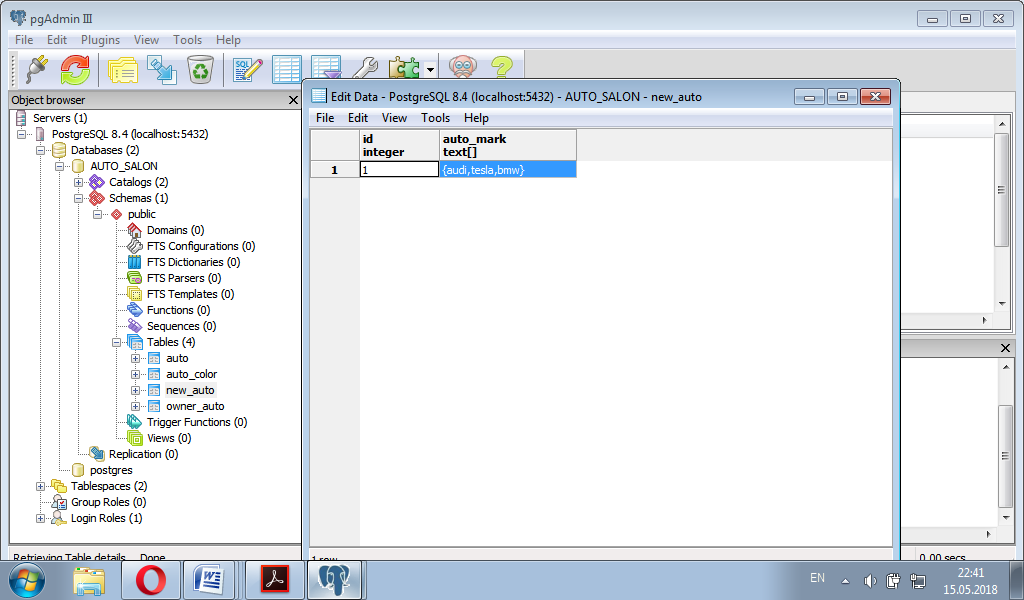


Рис. 3. Использование массивов

4. Использование последовательностей:

CREATE TABLE sequs

(

inv integer NOT NULL,

mark text

);

CREATE SEQUENCE inv START 1;

SELECT nextval('inv') FROM sequs;

INSERT INTO sequs VALUES (nextval('inv'), 'audi');

INSERT INTO sequs VALUES (nextval('inv'), 'opel');

INSERT INTO sequs VALUES (nextval('inv'), 'tesla');

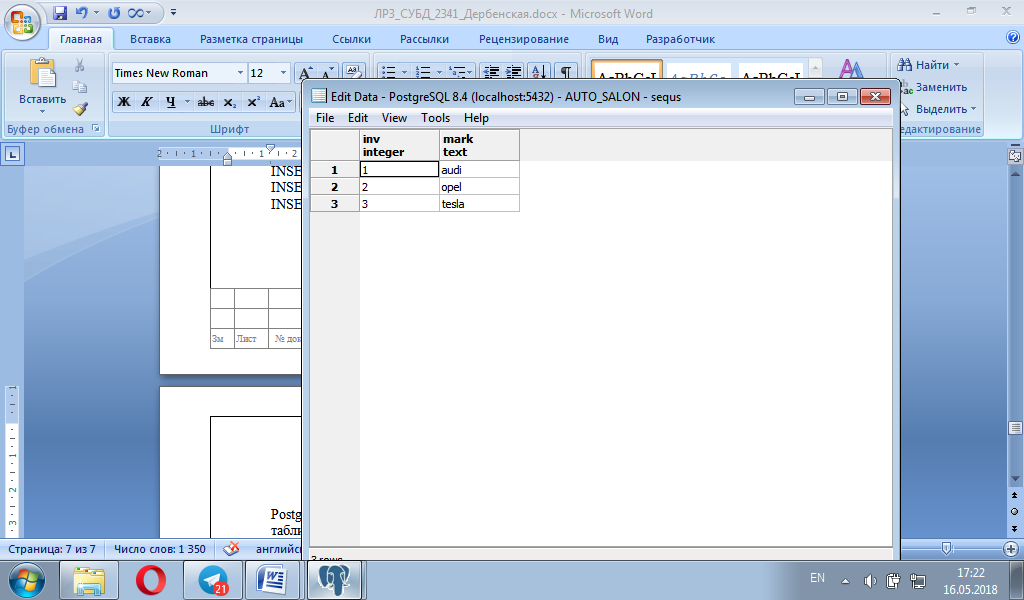


Рис. 4. Использование последовательностей

**Вывод.** Во время выполнения лабораторной работы я научилась работе с PostgreSQL, созданию объектно-реляционных связей, использованию ограничений в таблицах, массивов, последовательностей, backup и restore БД.