## Объявление типов

Составной тип представляет структуру табличной строки или записи; по сути это просто список имён полей и соответствующих типов данных. PostgreSQL позволяет использовать составные типы во многом так же, как и простые типы. Например, в определении таблицы можно объявить столбец составного типа.

### Объявление составных типов

Ниже приведены два простых примера определения составных типов:

CREATE TYPE complex AS (

r double precision,

i double precision

);

CREATE TYPE inventory\_item AS (

name text,

supplier\_id integer,

price numeric

);

Синтаксис очень похож на CREATE TABLE, за исключением того, что он допускает только названия полей и их типы. Какие-либо ограничения (такие как NOT NULL) в 11 версии не поддерживаются. Заметьте, что ключевое слово AS здесь имеет значение; без него система будет считать, что подразумевается другой тип команды CREATE TYPE, и выдаст неожиданную синтаксическую ошибку.

Определив такие типы, мы можем использовать их в таблицах:

CREATE TABLE on\_hand (

item inventory\_item,

count integer

);

INSERT INTO on\_hand VALUES (ROW('fuzzy dice', 42, 1.99), 1000);

или функциях:

CREATE FUNCTION price\_extension(inventory\_item, integer) RETURNS numeric

AS 'SELECT $1.price \* $2' LANGUAGE SQL;

SELECT price\_extension(item, 10) FROM on\_hand;

Всякий раз, когда создаётся таблица, вместе с ней автоматически создаётся составной тип. Этот тип представляет тип строки таблицы, и его именем становится имя таблицы. Например, при выполнении команды:

CREATE TABLE inventory\_item (

name text,

supplier\_id integer REFERENCES suppliers,

price numeric CHECK (price > 0)

);

в качестве побочного эффекта будет создан составной тип inventory\_item, в точности соответствующий тому, что был показан выше, и использовать его можно так же. Однако заметьте, что в текущей реализации есть один недостаток: так как с составным типом не могут быть связаны ограничения, то описанные в определении таблицы ограничения не применяются к значениям составного типа вне таблицы. (Чтобы обойти этот недостаток, создайте домен поверх составного типа и добавьте желаемые ограничения в виде ограничений CHECK для данного домена.)

### Конструирование составных значений

Чтобы записать значение составного типа в виде текстовой константы, его поля нужно заключить в круглые скобки и разделить их запятыми. Значение любого поля можно заключить в кавычки, а если оно содержит запятые или скобки, это делать обязательно. (Подробнее об этом говорится ниже.) Таким образом, в общем виде константа составного типа записывается так:

'( значение1 , значение2 , ... )'

Например, эта запись:

'("fuzzy dice",42,1.99)'

будет допустимой для описанного выше типа inventory\_item. Чтобы присвоить NULL одному из полей, в соответствующем месте в списке нужно оставить пустое место. Например, эта константа задаёт значение NULL для третьего поля:

'("fuzzy dice",42,)'

Если же вместо NULL требуется вставить пустую строку, нужно записать пару кавычек:

'("",42,)'

Здесь в первом поле окажется пустая строка, а в третьем — NULL.

Значения составных типов также можно конструировать, используя синтаксис выражения ROW. В большинстве случаев это значительно проще, чем записывать значения в строке, так как при этом не нужно беспокоиться о вложенности кавычек. Перепишем пример выше:

ROW('fuzzy dice', 42, 1.99)

ROW('', 42, NULL)

Ключевое слово ROW на самом деле может быть необязательным, если в выражении определяются несколько полей, так что эту запись можно упростить до:

('fuzzy dice', 42, 1.99)

('', 42, NULL)

### Обращение к составным типам

Чтобы обратиться к полю столбца составного типа, после имени столбца нужно добавить точку и имя поля, подобно тому, как указывается столбец после имени таблицы. На самом деле, эти обращения неотличимы, так что часто бывает необходимо использовать скобки, чтобы команда была разобрана правильно. Например, можно попытаться выбрать поле столбца из тестовой таблицы on\_hand таким образом:

SELECT item.name FROM on\_hand WHERE item.price > 9.99;

Но это не будет работать, так как согласно правилам SQL имя item здесь воспринимается как имя таблицы, а не столбца в таблице on\_hand. Поэтому этот запрос нужно переписать так:

SELECT (item).name FROM on\_hand WHERE (item).price > 9.99;

либо указать также и имя таблицы (например, в запросе с многими таблицами), примерно так:

SELECT (on\_hand.item).name FROM on\_hand WHERE (on\_hand.item).price > 9.99;

В результате объект в скобках будет правильно интерпретирован как ссылка на столбец item, из которого выбирается поле.

При выборке поля из значения составного типа также возможны подобные синтаксические казусы. Например, чтобы выбрать одно поле из результата функции, возвращающей составное значение, потребуется написать что-то подобное:

SELECT (my\_func(...)).field FROM ...

Без дополнительных скобок в этом запросе произойдёт синтаксическая ошибка.

### Изменение составных типов

Ниже приведены примеры правильных команд добавления и изменения значений составных столбцов. Первые команды иллюстрируют добавление или изменение всего столбца:

INSERT INTO mytab (complex\_col) VALUES((1.1,2.2));

UPDATE mytab SET complex\_col = ROW(1.1,2.2) WHERE ...;

В первом примере опущено ключевое слово ROW, а во втором оно есть; присутствовать или отсутствовать оно может в обоих случаях.

Мы можем изменить также отдельное поле составного столбца:

UPDATE mytab SET complex\_col.r = (complex\_col).r + 1 WHERE ...;

Заметьте, что при этом не нужно (и на самом деле даже нельзя) заключать в скобки имя столбца, следующее сразу за предложением SET, но в ссылке на тот же столбец в выражении, находящемся по правую сторону знака равенства, скобки обязательны.

И мы также можем указать поля в качестве цели команды INSERT:

INSERT INTO mytab (complex\_col.r, complex\_col.i) VALUES(1.1, 2.2);

Если при этом мы не укажем значения для всех полей столбца, оставшиеся поля будут заполнены значениями NULL.

### Использование составных типов в запросах

С составными типами в запросах связаны особые правила синтаксиса и поведение. Эти правила образуют полезные конструкции, но они могут быть неочевидными, если не понимать стоящую за ними логику.

В PostgreSQL ссылка на имя таблицы (или её псевдоним) в запросе по сути является ссылкой на составное значение текущей строки в этой таблице. Например, имея таблицу inventory\_item, показанную выше, мы можем написать:

SELECT c FROM inventory\_item c;

Этот запрос выдаёт один столбец с составным значением, и его результат может быть таким:

c

------------------------

("fuzzy dice",42,1.99)

(1 row)

Заметьте, однако, что простые имена сопоставляются сначала с именами столбцов, и только потом с именами таблиц, так что такой результат получается только потому, что в таблицах запроса не оказалось столбца с именем c.

Обычную запись полного имени столбца вида имя\_таблицы.имя\_столбца можно понимать как применение выбора поля к составному значению текущей строки таблицы. (Из соображений эффективности на самом деле это реализовано по-другому.)

Когда мы пишем

SELECT c.\* FROM inventory\_item c;

то, согласно стандарту SQL, мы должны получить содержимое таблицы, развёрнутое в отдельные столбцы:

name | supplier\_id | price

------------+-------------+-------

fuzzy dice | 42 | 1.99

(1 row)

как с запросом

SELECT c.name, c.supplier\_id, c.price FROM inventory\_item c;

PostgreSQL применяет такое развёртывание для любых выражений с составными значениями, хотя как показано выше, необходимо заключить в скобки значение, к которому применяется .\*, если только это не простое имя таблицы. Например, если myfunc() — функция, возвращающая составной тип со столбцами a, b и c, то эти два запроса выдадут одинаковый результат:

SELECT (myfunc(x)).\* FROM some\_table;

SELECT (myfunc(x)).a, (myfunc(x)).b, (myfunc(x)).c FROM some\_table;