Ликбез по SQL

Балдин Е.М.

15 октября 2013 г.

Цель этой статьи *лик*видация *без*грамотности (*ликбез*) по теме SQL. SQL (произносится как «эс-кью-эл») — язык программирования, используемый для создания, модификации и управления данных. Этот текст ни в коей мере не является исчерпывающей документацией по этому вопросу.

1. Язык определения данных

Группа операторов SQL, ответственных за описание и модификацию структуры базы, представляют из себя специализированный язык определения данных (Data Definition Language).

Язык определения данных включает в себя следующие операторы:

`CREATE` — позволяет создать новую базу данных, таблицу (`TABLE`), индекс (`INDEX`), представление (`VIEW`) или какую-либо другую сущность,

`DROP` — позволяет удалить то, что было создано с помощью `CREATE`,

`ALTER` — позволяет изменить параметры уже существующего объекта.

Из перечисленных наиболее популярна команда для создания таблиц:

```
CREATE TABLE «имя таблицы» (
  «имя столбца 1» «тип данных»
       [DEFAULT «значение по умолчанию»] [«ограничения на столбец»],
 [«имя столбца 2» ...,]
 [...,]
 [«ограничения на таблицу»]
)
```

Ключевое слово `DEFAULT` предваряет значение, которое присваивается, если при вставке данных отсутствует инициализация.

Ограничения, накладываемые на столбец, могут характеризоваться ключевыми словами `NOT NULL` — инициализация при вставке обязательна, `UNIQUE` — значение должно быть уникальным, `PRIMARY KEY` — значение объявляется первичным ключом[1], `CHECK(условное выражение)` — проверка значения и `REFERENCES` — ссылка на допустимый диапазон значений, представленных во внешней таблице (внешний ключ `FOREIGN KEY`). Ограничения, накладываемые на таблицу в целом, дублируют описанную функциональность, позволяя при составлении требований использовать имена несколько столбцов. Например, первичный ключ вполне может быть составным.

Как `CREATE TABLE` позволяет создать таблицу, так `DROP TABLE` позволяет её уничтожить. Синтаксис команды уничтожения гораздо проще команды созидания. Надо только добавить «имя таблицы». Очень опасная для данных команда.

Довольно редко, но бывает необходимо изменить уже существующую таблицу:

```
ALTER TABLE «имя таблицы» «действие»
```

В качестве «действия» можно добавить столбец или ограничение `ADD`, удалить столбец или ограничение `DROP`, изменить параметра столбца `ALTER`, а так же «повесить»/«выключить» триггер `ENABLE/DISABLE TRIGGER`.

2. Язык манипулирования данными

Группа операторов SQL, ответственных за добавление, удаление и модификацию данных, представляют из себя специализированный язык манипулирования данными (Data Manipulation Language).

Язык манипулирования данными включает в себя следующие операторы:

`INSERT` — позволяет добавить одну или несколько строк (rows) данных в уже существующую таблицу,

`UPDATE` — позволяет изменить уже существующие данные,

`DELETE` — позволяет удалить одну и более строк данных из таблицы,

`TRUNCATE` — позволяет очистить одну или несколько таблиц от данных (очень опасная команда).

Для вставки данных следует воспользоваться командой:

```
INSERT INTO «имя таблицы»
          [(«список столбцов»)] VALUES («список значений»)
```

Столбцы и значения в соответствующих списках разделяются запятыми. При отсутствии списка столбцов значения присваиваются в соответствии с порядком именования столбцов при создании таблицы.

[1] В таблице может быть ровно один первичный ключ. Значения объявленное первичным ключом должно быть определенным и уникальным (`NOT NULL`+`UNIQUE`)

В качестве значения команде INSERT можно передать NULL. Это эквивалентно тому, что соответствующее поле не инициализируется при вставке.

Для модификации данных следует использовать команду:

```
UPDATE «имя таблицы»
    SET «столбец»=«значение» [,...]
    [ WHERE «условное выражение» ]
```

Условные выражения могут объединяться по «и» (AND) или по «или» (OR). Логика в SQL трёхзначная. Кроме ожидаемых значений для условных выражений, таких как «истина» (TRUE) и «ложь» (FALSE), допустимо значение «не определено» (UNKNOWN).

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Таблица 1. Таблица истинности для оператора OR

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

Таблица 2. Таблица истинности для оператора AND

Для удаления данных используются команды:

```
DELETE FROM «имя таблицы»
    [ WHERE «условное выражение» ]
```

или

```
TRUNCATE TABLE  «имя таблицы» [, ...]
```

Как это не печально, но удалять гораздо проще, чем добавлять данные. Не следует злоупотреблять этими операторами.

3. Select

Самый популярный оператор в SQL — это SELECT. Получение данных фактически всегда организуется с его помощью. Он всего один, поэтому нет никакого языка получения данных — он сам себе язык. Причём язык декларативный, потому как при использовании SELECT описываются свойства искомых данных, а не информация о том как эти данные получить.

Урезанные правила для SELECT представлены ниже:

```
SELECT «список искомых данных»
    [ FROM «список источников получения данных» ]
    [ WHERE «условное выражение» ]
    [ GROUP BY «выражение» [, ...]]
    [ HAVING «условное выражение» [, ...] ]
    [ ORDER BY «выражение» [, ...] ]
    [ LIMIT «число» ]
    [ OFFSET «число» ]
```

Если в качестве «списка искомых данных» передать «*» (звёздочку), то выводятся все поля из «списка источников получения данных». Результат, выдаваемый SELECT, можно использовать как источник получения данных наравне именами таблиц. Вложенный SELECT при использовании должен заключаться в круглые скобки. С помощью WHERE описываются свойства, которые хочется видеть среди полученных данных.

Инструкция GROUP BY позволяет сгруппировать результаты по указанному признаку. Инструкция HAVING выполняет примерно те же функции, что и инструкция WHERE, но работает уже после применения GROUP BY. Поэтому в случае HAVING можно использовать агрегатные функции.

Сортировка данных обеспечивается с помощь инструкции ORDER BY. «Выражение» для сортировки представляет из себя функцию от имён столбцов (просто имя столбца тоже выражение) и метода сортировки. Через запятую можно указать ещё одно выражение, которое принимается во внимание если предыдущее выражение для сортируемых строчек выдаёт одинаковые значения. Сортировка может производиться по возрастанию (ASC), по убыванию (DESC) и по заданному пользователем алгоритму (USING оператор сравнения[2]).

Для ограничения на число получаемых в ответ на запрос строк можно использовать инструкцию LIMIT, которая гарантирует, что число выведенных строк не будет превышать указанное в качестве параметра число. Инструкция OFFSET указывает SELECT сколько строк из уже отобранных пропустить, прежде чем начать вывод.

4. Управление доступом к данным

Группа операторов SQL, ответственных за дифференциацию пользователей путём предоставления и отмены привилегий, представляют из себя специализированный язык управления доступом к данным (Data Control Language).

Привилегии на работу с таблицами пользователям базы данных предоставляются с помощью команды:

[2] ASC эквивалентен конструкции USING <, DESC — USING >.

```
GRANT «список привилегий»
    ON TABLE  «имя таблицы» [, ...]
    TO «пользователь» [, ...]
```

Типы привилегий которые можно предоставить пользователям:

`SELECT` — разрешение на выполнение запроса на получения данных,

`INSERT` — разрешение на добавление данных в таблицу,

`UPDATE` — разрешение на изменение данных,

`DELETE` — разрешение на удаление данных,

`REFERENCES` — разрешение на создание ограничения по внешнему ключу,

`TRIGGER` — разрешение на создание триггера,

`ALL` — можно делать абсолютно всё.

Эти привилегии можно дать только в случае, если таблица уже существует. На создание таблицы тоже надо иметь право:

```
GRANT CREATE
    ON DATABASE  «имя базы данных» [, ...]
    TO «пользователь» [, ...]
```

Для отмены уже имеющихся привилегий используется команда:

```
REVOKE «список привилегий»
    ON «имя таблицы» [, ...]
    FROM «пользователь» [, ...]
```

5. Ссылочная целостность

Если все значения столбца A присутствуют в столбце B, то столбец A *ссылается* на столбец B. Если столбец B является потенциальным *ключом* (соответствует ограничению PRIMARY KEY или UNIQUE), то любое значение столбца A соотносится ровно одному значению столбца B. В этом случае столбец A является *внешним* ключом (foreign key) по отношению к столбцу B, который в свою очередь называется *родительским* ключом.

Реляционная база данных не должна содержать значений внешних ключей, не имеющих соответствия, иными словами должна поддерживаться *ссылочная целостность* (referential integrity).

Ссылочная целостность поддерживается PostgreSQL посредством механизма внешних ключей. Для создания внешнего ключа необходима таблица с потенциальным ключом:

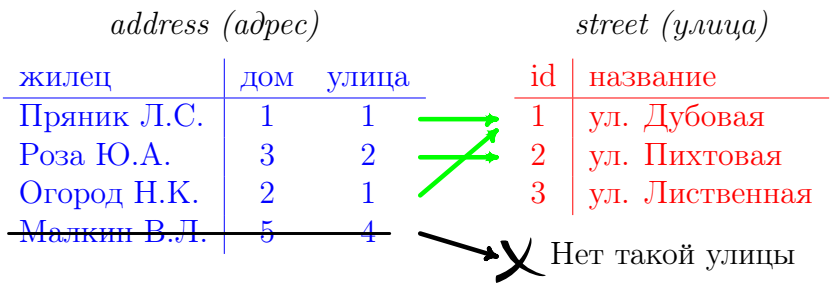

Рис. 1. Столбец «улица» таблицы address является внешним ключом, ссылающимся на столбец «id» таблицы street. Ссылочная целостность не выполнена для Малкина В.Л., так как в таблице *street* нет улицы с идентификатором 4.

```
wikidb=> CREATE TABLE street ( -- список улиц
wikidb(>     id integer PRIMARY KEY, -- ключ
wikidb(>     name text -- название улицы
wikidb(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
         implicit index "street_pkey" for table "street"
wikidb=> INSERT INTO street VALUES (1,'ул. Дубовая');
wikidb=> INSERT INTO street VALUES (2,'ул. Пихтовая');
wikidb=> INSERT INTO street VALUES (3,'ул. Лиственная');
```

Теперь можно создать таблицу с внешним ключом:

```
wikidb=> CREATE TABLE address ( -- адресный список
wikidb(>     person text, -- имя человека
wikidb(>     house text, -- номер дома
wikidb(>     -- внешний ключ
wikidb(>     street_id integer REFERENCES street (id)
wikidb(> );
wikidb=> INSERT INTO address VALUES ('Пряник Л.С.','1',1);
wikidb=> INSERT INTO address VALUES ('Роза Ю.А.','3',2);
wikidb=> INSERT INTO address VALUES ('Огород Н.К.','2',1);
```

Нельзя добавить данные во внешний ключ отсутствующие в родительском ключе:

```
wikidb=> INSERT INTO address VALUES ('Малкин В.Л.','5',4);
ERROR:  insert or update on table "address" violates
        foreign key constraint "address_street_id_fkey"
DETAIL:  Key (street_id)=(4) is not present in table "street".
```

Нельзя из родительского ключа при настройках по умолчанию удалить данные на которые пока ссылаются:

```
wikidb=> DELETE FROM street WHERE id=1;
ERROR:  update or delete on "street" violates foreign key
        constraint "address_street_id_fkey" on "address"
DETAIL:  Key (id)=(1) is still referenced from table "address".
```

Вот такая она — ссылочная целостность.

6. Транзакции

Транзакция — это единый блок операций, который нельзя разорвать. Либо совершается весь блок, либо всё отменяется. PostgreSQL в условиях параллельного доступа распространяет информацию об операциях только по завершению транзакции. Транзакция начинается с оператора `BEGIN` и заканчивается оператором `COMMIT` (подтверждение транзакции) или `ROLLBACK` (отмена транзакции). Возможен режим, когда каждый запрос сам себе транзакция, например, такой режим по умолчанию используется в `psql`. Для отмены этого режима достаточно набрать `BEGIN;`. Неудобством при использовании транзакций является то, что в случае ошибки какого-то из запросов приходится отменять всю транзакцию. Для устранения этого недостатка в 8ой версии PostgreSQL были добавлены точки сохранения (saveponts).

```
test=> -- Начинаем транзакцию
test=> BEGIN;
test=> -- Здесь идёт блок операторов, который удачно завершается

test=> -- Ставим метку
test=> SAVEPOINT savepoint_one;
test=> -- Здесь идёт блок операторов, в котором произошла ошибка

test=> -- Откатываемся до установленной метки,
test=> -- а не отменяем всю транзакцию
test=> ROLLBACK TO savepoint_one;
test=> -- Повторяем последний блок

test=> -- Завершаем транзакцию
test=> COMMIT;
test=> -- Всё, теперь изменения доступны всем
```