GeekBrains

Работа со структурой базы, CTE, Оценка занимаемого места

Урок 2

На этом уроке

Ограничения целостности (Constraints)

Создание структуры базы

Data Definition Language (DDL) и Data Manipulation Language (DML)

Работа с insert

WITH Queries (Common Table Expressions)

Оценка занимаемого места

Ограничения целостности (Constraints)

Гарантируют сохранность формата данных в базе.

Ограничения целостности (Constraints)

`CHECK` — позволяет задать любое выражение, которое вернёт boolean. Например `(salary >= 0)`

`NOT NULL` — гарантирует не пустое значение в столбце

`UNIQUE` — указывает, что значения в столбце не могут дублироваться между строками таблицы

`PRIMARY KEY` — заданного столбца (столбцов) должны быть достаточны для уникальной идентификации записи в таблице (`UNIQUE + NOT NULL`)

`FOREIGN KEY` — устанавливает связь между столбцами в текущей и заданной таблице

```sql
create table employees (
    -- Тип данных bigint с ограничением generated as identity позволяет
    -- создать столбец с уникальным идентификатором для каждой строки.
    -- Уникальность обеспечивается монотонно возрастающей последовательностью чисел.
    -- @see: https://www.postgresql.org/docs/current/sql-createtable.html
    id bigint generated by default as identity,
    salary integer not null,
    manager_id integer not null,

    -- constraint primary key устанавливает ограничение целостности, указывающее,
    -- что значения в заданном столбце должны быть уникальные и не пустые (not null).
    -- Установка этого ограничения автоматически создаёт BTree-индекс для заданного столбца.
    -- @see: https://www.postgresql.org/docs/current/ddl-constraints.html#DDL-CONSTRAINTS-PRIMARY-KEYS
    constraint employees_id_pkey primary key (id),

    -- constraint check устанавливает ограничение на столбец salary,
    -- требующее, чтобы зарплата была неотрицательным числом.
    -- @see: https://www.postgresql.org/docs/current/ddl-constraints.html#DDL-CONSTRAINTS-CHECK-CONSTRAINTS
    constraint employees_salary_not_negative_check check (salary >= 0),

    -- constraint foreign key устанавливает связь между столбцом manager_id и строкой в таблице employee.
    -- в данном случае связь является рекурсивной, так как ссылается на свою же таблицу.
    -- Ограничение foreign key требует наличия ограничения unique на целевом (destination) поле.
    -- on delete restrict указывает, что при удалении менеджера, у которого есть подчиненные, произойдет ошибка.
    constraint employees_fk_manager_id foreign key (manager_id) references employees (id) on delete restrict
);
```

Ограничения целостности (Constraints)

Если именовать ограничения явно, то при появлении ошибки легче понять, в чём проблема

```
1 mydb=> insert into employees (salary, manager_id) values (20000, 42);
2 ERROR:  insert or update on table "employees" violates foreign key constraint
  "employees_fk_manager_id"
3 DETAIL:  Key (manager_id)=(42) is not present in table "employees".
```

Ограничения целостности (Constraints)

Ограничения можно добавлять и к созданным ранее таблицам

```
1 alter table employees add constraint employees_salary_not_negative_check check (salary >= 0);
```

Добавление ограничения блокирует выполнение операций на большой таблице

Ограничения целостности (Constraints)

Есть возможность менять схему таблиц без долгих блокировок, если устанавливать ограничение в несколько этапов

```
1 alter table employees add constraint employees_salary_not_negative_check check (salary >= 0) not
  valid;
2 -- чистим данные
3 alter table employees validate constraint employees_salary_not_negative_check;
```

Создание структуры базы

```
docker run \
    --rm -it \
    -p 5432:5432 \
    --name postgres \
    -e POSTGRES_PASSWORD=password \
    -e PGDATA=/var/lib/postgresql/data \
    -v $(pwd)/mntdata:/var/lib/postgresql/data \
    -v $(pwd)/workdir:/workdir \
    -w /workdir \
    postgres:13.1
```

```
~/projects/pgbrains$ tree
.
└── workdir
    └── schema.sql
```

```
mydb=> \i schema.sql
CREATE TABLE
..
CREATE TABLE
```

DDL и DML

Data Definition Language (DDL)

Операции, добавляющие, изменяющие или удаляющие объекты в базе или изменяющие структуру таблицы.

```
alter table, create table
```

Data Manipulation Language (DML)

Операции для добавления, изменения, удаления строк в таблице

```
insert, update
```

Insert

`insert` может выполнять функции `update`, используя опцию `on conflict`

```
1 insert into departments (id, parent_id, name) values (1, 1, 'root')
2     on conflict (id) do update set
3         parent_id = excluded.parent_id,
4         name = excluded.name;
```

Insert

`insert` может вернуть информацию о вставленной строке, используя оператор `returning`

```
1 insert into departments (parent_id, name) values (42, 'leaf') returning id;
```

WITH Queries (Common Table Expressions)

Оператор `WITH` позволяет скомпоновать несколько запросов в одно выражение.

Используется для передачи результата одного запроса в другой и улучшения читаемости кода.

WITH Queries (Common Table Expressions)

```
with w_deleted as (
    delete from employees
    where
        salary > 1000
    returning manager_id
)
select * from employees where id in (
    select distinct(manager_id) from w_deleted
);
```

WITH Queries (Common Table Expressions)

Оператор WITH RECURSIVE позволяет строить рекурсивные запросы, т. е. запросы, которые могут ссылаться на свой собственный вывод.

Импорт данных

В рабочей директории, в файле `data.sql` готовим данные для наполнения базы

Используя команду `\i data.sql`, выполняем все выражения из файла

Примечание: чтобы наполнить таблицу случайными данными, можно использовать утилиту mock-data: `mock -a 127.0.0.1 -d mydb -u myuser -w secret tables -t employees,departments,positions -r 1000000`

Оценка занимаемого места: файловая система

Посмотрев объем директории $PGDATA (`du -sh $PGDATA/*`), можно оценить, сколько места занимают все данные, хранимые и обрабатываемые сервером PostgreSQL.

```
$ du -sh $PGDATA/*
23M /var/lib/postgresql/data/base
544K    /var/lib/postgresql/data/global
..
17M /var/lib/postgresql/data/pg_wal
..
```

Оценка занимаемого места: табличные данные

psql-команда `\d+` выводит расширенный список всех таблиц, в котором в столбце `Size` будет показано, сколько каждая таблица занимает физического пространства.

```
mydb=> \d+
                            List of relations
 Schema | Name        | Type  | Owner  | Persistence |  Size      | Description
--------+-------------+-------+--------+-------------+------------+-------------
 public | departments | table | myuser | permanent   | 4252 bytes |
(1 row)
```

Оценка занимаемого места: табличные данные

Используя представление `pg_class`, можно найти самые тяжёлые таблицы в базе

```
mydb=> select oid::regclass::text as tablename, pg_total_relation_size(oid) as bytes from pg_class
 where relkind = 'r' order by bytes desc limit 5;
    tablename    |  bytes
-----------------+---------
 pg_depend       | 1130496
 pg_proc         | 1048576
 pg_rewrite      |  704512
 pg_attribute    |  671744
 pg_description |  573440
(5 rows)
```

Оценка занимаемого места: количество строк

Узнать точное количество строк в таблице можно, используя `select count(1)`

```
mydb=> select count(1) as cnt from departments;
 cnt
-----
   5
(1 row)
```

Оценка занимаемого места: количество строк

На больших таблицах (больше миллиона строк) нельзя делать запрос на подсчёт точного количества строк, так как происходит полное сканирование данных.

Чтобы узнать количество строк в большой таблице, обращайтесь к статистике, которую собирает Postgres для оптимизации запросов

```
mydb=> select reltuples::bigint as estimate from pg_class where oid =
'public.departments'::regclass;
 estimate
----------
         5
(1 row)
```

Практическое задание