Лекция 7

Определение столбцов

Определение столбцов

```
имя_столбца тип_данных
[COMPRESSION метод_сжатия]
[COLLATE правило_сортировки] [ограничение_столбца [... ]]
```

- **Тип данных** – определяет набор допустимых значений столбца
  - может включать определение массива с этим типом
- **COMPRESSION** – задаёт метод сжатия для столбца
  - Сжатие поддерживается только для типов данных переменной длины и применяется только для столбцов с режимом хранения **main** или **extended**
  - Поддерживаются следующие методы сжатия: pglz и lz4 (lz4 поддерживается, только если PostgreSQL был собран с ключом --with-lz4)
- **COLLATE** – определяет правило сортировки для столбца
  - Если оно отсутствует, используется правило сортировки по умолчанию, установленное для типа данных столбца
- **Ограничение_столбца** – задают ограничения (проверки), которым должны удовлетворять добавляемые или изменяемые значения

Типы данных

- Символьные:
  - varchar(n) , char(n) , text
- Числовые:
  - smallint (int2), integer (int4), bigint (int8), money (8 байт), numeric(scale, precision) и decimal(scale, precision)
  - real, double precision и float(p)
  - serial (int4), bigserial(int8) и smallserial(int2)
- Дата и время:
  - date, time и time with time zone
  - timestamp, timestamp with time zone (timestamptz)
- Бинарные:
  - bytea
- Логические
  - boolean

Дата и время

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
`timestamp [ (p) ] [ without time zone ]`	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
`timestamp [ (p) ] with time zone`	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
`date`	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
`time [ (p) ] [ without time zone ]`	8 байт	время суток (без даты)	00:00:00	24:00:00	1 микросекунда / 14 цифр
`time [ (p) ] with time zone`	12 байт	только время суток (с часовым поясом)	00:00:00+1459	24:00:00-1459	1 микросекунда / 14 цифр
`interval [ поля ] [ (p) ]`	16 байт	временной интервал	-178000000 лет	178000000 лет	1 микросекунда / 14 цифр

Специальные типы данных

- UUID
- Геометрические типы
- Типы, описывающие сетевые адреса
- XML
- Типы JSON
- Массивы
- Составные типы
- Диапазонные типы
- Пользовательские и доменные типы

https://postgrespro.ru/docs/postgresql/14/datatype

Генерируемые столбцы

- Поддерживается 3 типа генерируемых столбцов:
  - Вычисляемый столбец на основе выражения
  - Столбец идентификации
  - Столбец SERIAL

https://postgrespro.ru/docs/postgresql/14/ddl-generated-columns

Вычисляемый столбец на основе выражения

```
GENERATED ALWAYS AS ( генерирующее_выражение ) STORED
```

- Значения в столбце вычисляется с помощью **генерирующего выражения**, которое:
  - может обращаться к другим столбцам данной таблицы
  - не может обращаться к другим генерируемым столбцам
  - должно включать только постоянные функции и операторы
- **STORED** – значения вычисляются при записи (добавлении или изменении) и сохраняются на диске
  - виртуальные столбцы (вычисляются при чтении) не реализованы!

```
CREATE TABLE people (
    ...,
    height_cm numeric,
    height_in numeric GENERATED ALWAYS AS (height_cm / 2.54) STORED
);
```

Столбец идентификации

```
GENERATED {ALWAYS|BY DEFAULT} AS IDENTITY [(параметры_последовательности)]
```

- Автоматически создается **неявная последовательность**, из которой столбец будет автоматически получать значения
  - Столбцу неявно назначается свойство **NOT NULL**
- **ALWAYS**:
  - При вставке (INSERT) пользовательское значение используется, только если указано **OVERRIDING SYSTEM VALUE**
  - При обновлении (UPDATE) любое отличное от **DEFAULT** значение будет отвергнуто
- **BY DEFAULT**:
  - При вставке (INSERT) пользовательское значение имеет приоритет
  - При обновлении (UPDATE) столбец может быть изменён обычным образом
- Предложение **параметры_последовательности** позволяет переопределить свойства последовательности

```
CREATE TABLE distributors (
     did    integer     PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
     name   varchar(40) NOT NULL CHECK (name <> '')
);
```

Столбец SERIAL

- В PostgreSQL имеется особый тип для создания столбцов идентификации — **SERIAL**
  - В основе этого типа лежит тип **INTEGER**
  - Столбцу неявно назначается свойство **NOT NULL**
  - Автоматически создается **явная последовательность**
  - Автоматически определяется ограничение **DEFAULT**, которое получает значение с помощью функции `nextval('имя_последовательности')`
- Для каждого поля этого типа создаётся таблица, которая содержит, помимо прочего, текущее число из последовательности
  - При извлечении очередного номера из последовательности текущее число увеличивается
- Получить имя созданной последовательности

```
SELECT pg_get_serial_sequence('table_name','column_name');
```

Столбцы, содержащие массивы

- Для определения столбца массива необходимо после указания типа данных добавить пару квадратных скобок. Скобки указывают на то, что в этот столбец можно без ограничений вставлять более одного значения описываемого типа данных:
  - **single_array type[]** – Одномерный массив значений
  - **multi_array type[][]** -- Многомерный массив значений

```
CREATE TABLE favorite_books
        (employee_id integer,
         books text[]);
```

```
CREATE TABLE favorite_authors
        (employee_id integer,
         authors_and_titles text[][]);
```

- Преимущества сохранения значений в массиве, по сравнению с сохранением в одной текстовой строке:
  - каждое значение хранится физически отдельно от другого значения в столбце массива;
  - система знает, где начинается и заканчивается каждое значение массива, что позволяет выбирать значения по их индексу, а не вручную анализировать их из длинной текстовой строки

Изменение и удаление таблиц

Изменение таблиц

```
ALTER TABLE table_name action;
{RENAME TO  |
ADD COLUMN …| RENAME COLUMN …  | ALTER COLUMN … | DROP COLUMN … [CASCADE]}
```

- Поддерживаемые операции:
  - **Rename To** – переименование таблицы
  - **Add column** – добавление столбца
  - **Rename Column** – переименование столбца
  - **Drop column** – удаление столбца
  - **Alter Column** – изменение характеристик столбца:
    - изменение типа данных
    - переименование столбца
    - задание/изменение значения по умолчанию
    - добавление ограничений целостности*

https://postgrespro.ru/docs/postgresql/14/ddl-alter

Изменение таблиц

- Переименование таблицы

```
ALTER TABLE "Production"."Products"
RENAME TO "Production"."Product";
```

- Добавление столбца

```
ALTER TABLE "Production"."Product"
ADD COLUMN description text,
ADD COLUMN comments text NOT NULL;
```

- Переименование столбца

```
ALTER TABLE "Production"."Product"
RENAME COLUMN productid TO product_number;
```

- Удаление столбца

```
ALTER TABLE "Production"."Product"
DROP COLUMN description;
```

- Изменение столбца

```
ALTER TABLE "Production"."Product"
ALTER COLUMN  productname TYPE varchar(50);
ALTER TABLE "Production"."Product"
ALTER COLUMN unitprice SET DEFAULT 7.77;
```

https://postgrespro.ru/docs/postgresql/14/ddl-alter

ВАЖНО

- Изменение типа данных столбца может вызвать ошибку, в случае если:
  - Это может привести к усечению существующих данных
  - Преобразование имеющихся данных к указанному типу не поддерживается
  - Существует ссылающийся на данный столбец внешний ключ (FK) в другой таблице
- Удаление столбца, являющегося первичным ключом (PK) не поддерживается при наличие соответствующего внешнего ключа (FK)
  - В случае, если столбец действительно должен быть удален необходимо использовать параметр **CASCADE** - **DROP ... CASCADE**

```
ERROR: cannot drop column product_id of table pk_t because other objects depend on it
 Подробности: constraint fk_t_id_fkey on table fk_t depends on column product_id of table pk_t
 Подсказка: Use DROP ... CASCADE to drop the dependent objects too.
```

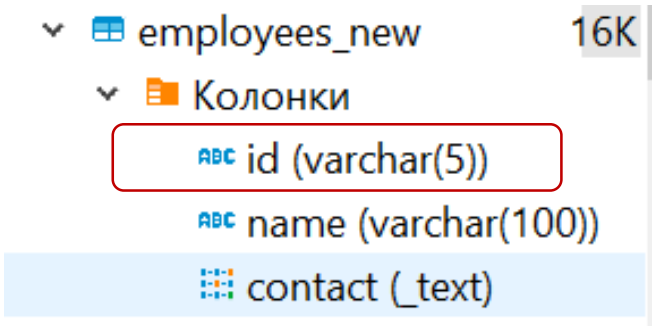
https://postgrespro.ru/docs/postgresql/14/ddl-alter

Изменение типа данных столбца

- В том случае, когда исходный тип данных столбца изменяется на другой тип данных в пределах одной группы, например, оба типа — числовые, то проблем обычно не возникает
- Если исходный и целевой типы данных относятся к разным группам, тогда потребуется использование фразы **USING** команды **ALTER TABLE**

```
ALTER TABLE seats
 ALTER COLUMN fare_conditions SET DATA TYPE integer
 USING ( CASE fare_conditions
                WHEN 'Economy'  THEN 1
                WHEN 'Business' THEN 2
                ELSE 3
         END );
```

Пример

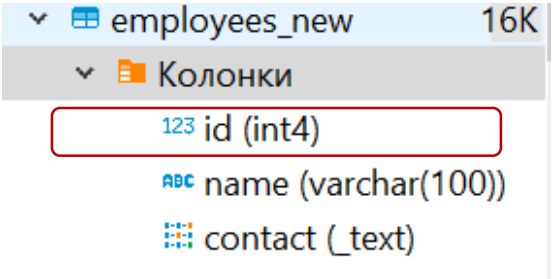


	id	name	contact
1	1	Alice John	{(408)-743-9045,(408)-567-7834}
2	2	Kate Joel	{(408)-783-5731}
3	3	James Bush	{(408)-745-8965,(408)-567-78234}

```
ALTER TABLE public.employees_new
ALTER COLUMN id SET DATA TYPE int;
```

SQL Error [42804]: ERROR: column "id" cannot be cast automatically to type integer
Подсказка: You might need to specify "USING id::integer".

```
ALTER TABLE public.employees_new
ALTER COLUMN id  SET DATA type int USING id::int;
```



Удаление таблиц

```
DROP TABLE [IF EXISTS] table_name [CASCADE| RESTRICT];
```

- Параметр **CASCADE** позволяет удалить таблицу и зависимые от нее объекты
- Параметр **RESTRICT** отклоняет удаление, если от таблицы зависит какой-либо объект
  - Параметр **RESTRICT** используется по умолчанию, если вы не укажете его явно в операторе DROP TABLE
- Для удаления сразу нескольких таблиц необходимо перечислить их в операторе **DROP TABLE** через запятую

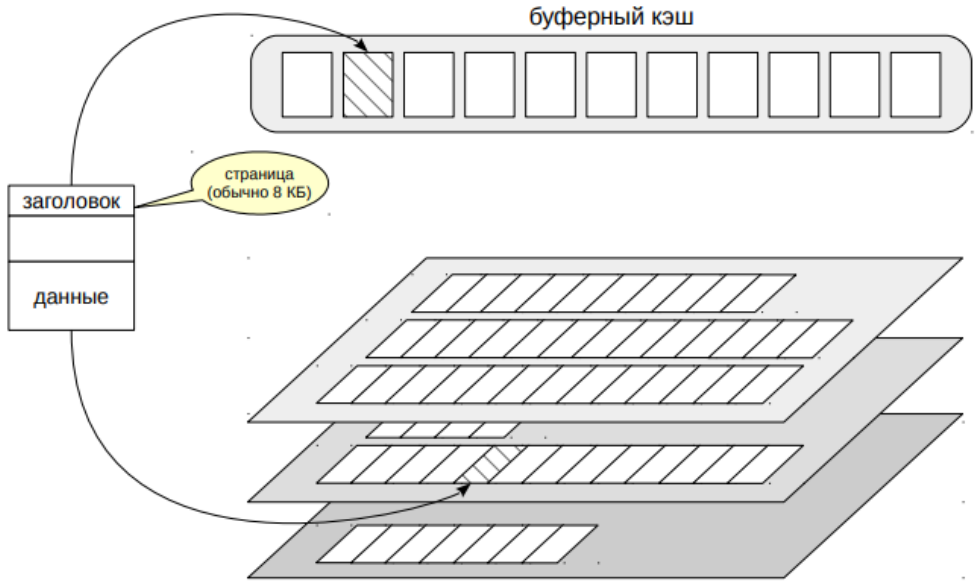
https://postgrespro.ru/docs/postgresql/14/ddl-alter

Хранение данных большого объема

Структура хранения данных

- Данные каждой таблицы сохраняются в отдельном файле
  - Каждый файл занимает не больше 1 GB (может быть изменено при сборке) и имеет свой уникальный идентификатор - **relfilenode**
  - Если таблица больше 1 GB, то ее данные сохраняются в нескольких файлах

```
SELECT relfilenode, relname
 FROM pg_class
WHERE relname='имя_таблицы';
```



- Файлы состоят из 8 КВ страниц, которые в случае необходимости помещаются в буферный кэш
  - Версия строки должна целиком помещаться на одну страницу

Хранение данных большого объема

- Использование PostgreSQL фиксированного размера страниц не позволяет кортежам занимать несколько страниц
- Поэтому для сохранения очень больших значений полей (varchar/text/bytea/numeric), может быть использована одна из следующих стратегий:
  - сжать данные столбца
  - вынести данные соответствующего столбца в отдельную служебную таблицу
  - объединить оба способа
- В PostgreSQL используется технология **TOAST— The Oversized Attributes Storage Technique**
  - Фактически, для каждой таблицы с «потенциально большими» полями автоматически создается парная TOAST-таблица с «нарезкой» каждой «большой» записи сегментами по 2KB (и к ней специальный индекс)
  - Если приходится записывать строку с «большим» значением, то реальная запись произойдет не только в основную таблицу, но и в TOAST
  - Данные из TOAST-таблицы читаются только при обращении к «длинному» атрибуту

Способы сохранения больших значений

- Все столбцы имеют связанную с ними стратегию хранения:
  - **Plain (p)** – не допускает ни сжатия, ни внешнего хранения (numbers, char)
  - **Extended (x)** – допускает как сжатие, так и внешнее хранение. **Это значение по умолчанию для большинства типов данных, поддерживающих TOAST**
    - Сначала предпринимается попытка сжатия, если строка все еще слишком велика создаётся TOAST таблица
  - **Main (m)** – допускает сжатие, а внешнее хранение применяется как крайняя мера, когда нет другого способа уменьшить строку (numeric)
  - **External (e)** – допускает отдельное хранение, но не сжатие (сразу создает TOAST-таблицу). Ускоряет работу с данными в текстовых (text) и байтовых (bytes) столбцах

Просмотр и изменение стратегии хранения

- Получение информации о текущей стратегии хранения

```
SELECT attname, atttypid::regtype,
                  CASE attstorage WHEN 'p' THEN 'plain'
                                  WHEN 'e' THEN 'external'
                                  WHEN 'm' THEN 'main'
                                  WHEN 'x' THEN 'extended'
                  END AS strategy
             FROM pg_attribute
            WHERE attrelid = 'имя_таблицы'::regclass AND attnum > 0;
```

- Изменение стратегии хранения

```
ALTER TABLE имя_таблицы ALTER COLUMN имя_столбца
SET STORAGE {MAIN | EXTENDED | EXTERNAL | PLAIN};
```

Пример получения информации о Toast-таблице

```
CREATE TABLE t(id serial PRIMARY KEY, n numeric);
```

```
select attname, atttypid::regtype,
    case attstorage when 'p' then 'plain'
        when 'e' then 'external'
        when 'm' then 'main'
        when 'x' then 'extended'
    end AS strategy
from pg_attribute
where attrelid = 't'::regclass and attnum > 0;
```

	attname	atttypid	strategy
1	id	integer	plain
2	n	numeric	main

```
SELECT relname, relfilenode FROM pg_class
WHERE OID = (
             SELECT reltoastrelid
             FROM pg_class
             WHERE relname='t'
);
```

Имя Toast-таблицы

	relname	relfilenode
1	pg_toast_95138	95 142

Просмотр содержимого Toast-таблицы

```
select *
from pg_toast.pg_toast_95138;
```

chunk_id	chunk_seq	chunk_data

```
INSERT INTO t(n)
SELECT 123456789::numeric ^ 12345::numeric;
```

```
select *
from pg_toast.pg_toast_95138;
```

	chunk_id	chunk_seq	chunk_data
1	95 148	0	a8 H â K " %} $%Ó M V U ... [1996]
2	95 148	1	ï « ¸ Ð x Æ L ä å z ° ó k ... [1996]
3	95 148	2	æ " T"Z a + %& %!u!' % £ Í%... [1996]
4	95 148	3	þ H z ò$< Ê µ %É Z ¨&r ... [1996]
5	95 148	4	Ã Â > ¸&å ç ½ $÷ C&$ ø!Ý (... [1996]
6	95 148	5	@$N 0 _ Û ¡ i Ì w Ú ! ò ... [1996]
7	95 148	6	i$Æ § (ã à X T m ä ç F ... [1996]
8	95 148	7	ï d B S + é _$ý m $x å ... [1996]

```
select chunk_id, count(*)
from pg_toast.pg_toast_95138
group by chunk_id;
```

	chunk_id	count
1	95 148	26
2	95 149	26

Системные данные таблицы

- PostgreSQL определяет ряд системных столбцов во всех таблицах, которые обычно невидимы для пользователя
  - они не будут отображаться в запросах, если они не запрошены явно
- Эти столбцы содержат метаданные о содержимом строк таблицы
  - Многие из них содержат данные, которые могут помочь различать кортежи (отдельные состояния строки) при работе с блоками транзакций
  - Любая строка таблицы, в дополнение к пользовательским столбцам, будет иметь значения в каждом из системных столбцов

```
SELECT tableoid , xmin, cmin,ctid, empid, lastname, firstname
FROM "HR"."Employees";
```

	tableoid	xmin	cmin	ctid	empid	lastname	firstname
1	51 179	5494	1	(0,4)	2	Funk	Don
2	51 179	5494	2	(0,5)	3	Lew	Judy
3	51 179	5494	3	(0,6)	4	Peled	Yael
4	51 179	5494	7	(0,10)	8	Cameron	Maria

Системные столбцы

Системный столбец	Описание
tableoid (table object identifier)	oid таблицы, содержащей строку. Имя и oid таблицы связаны системной таблицей pg_class.
xmin (transaction minimum)	Идентификатор транзакции, вставившей запись
cmin (command minimum)	Идентификатор команды, начинающийся с 0, связанный с транзакцией вставки записи
xmax (transaction maximum)	Идентификатор транзакции, удалившей запись. Если запись видна (не была удалена), он устанавливается равным нулю.
cmax (command maximum)	Идентификатор команды, связанный с транзакцией удаления записи. Как и xmax, если запись видна, он устанавливается равным нулю.
ctid (tuple identifier)	Идентификатор, описывающий физическое расположение записи в базе данных. Ctid представлен парой чисел: номер блока и индекс записи в этом блоке.