

# Задачи администрирования Локализация



## **Авторские права**

© Postgres Professional, 2016–2022.

Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов

## **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

## **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:  
[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

## **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Назначение локализации

Локали и категории

Работа с датами, числами, денежными единицами

Настройка сервера и клиента

Настройка сообщений сервера и клиентских утилит

Правила сортировки

## Поддержка кодировок

выбор кодировки символов для различных языков  
перекодировка символов между клиентом и сервером

## Функционал, зависящий от локализации

сортировка и сравнение символов  
функции `upper`, `lower`, `initcap`  
поиск по шаблону  
функции `to_char` с датами, числами, денежными единицами  
язык сообщений сервера и утилит

Возможности локализации в PostgreSQL позволяют хранить текст в различных кодировках. Для русского языка поддерживаются все основные кодировки символов, включая UTF8, WIN1251, KOI8R, ISO\_8859\_5.

Клиентское приложение может работать в кодировке, отличной от кодировки сервера. В таком случае настраивается преобразование символов между клиентом и сервером.

Кроме поддержки различных кодировок локализация влияет на работу следующего функционала сервера:

- сортировка символов, например в предложениях `ORDER BY` или в операциях сравнения (`>`, `<`);
- преобразование букв в верхний и нижний регистр в функциях `upper`, `lower`, `initcap`;
- поиск по шаблону в регулярных выражениях, операторах `LIKE`, `SIMILAR TO`, включая поиск без учета регистра символов;
- форматирование дат, чисел и денежных единиц в функциях `to_char`;
- выбор языка сообщений сервера и утилит.

<https://postgrespro.ru/docs/postgresql/13/locale>

## Локали

определяют язык, территорию и кодировку символов,  
например `ru_RU.UTF8`

установлены в операционной системе

## Категории локали

<code>lc_ctype</code>	классификация символов
<code>lc_collate</code>	правила сортировки символов
<code>lc_messages</code>	язык сообщений
<code>lc_monetary</code>	формат денежных единиц
<code>lc_numeric</code>	формат чисел
<code>lc_time</code>	формат даты и времени

PostgreSQL использует возможности локализации, предоставляемые операционной системой. Поэтому в ОС следует предварительно настроить локали, которые потребуются для работы СУБД.

Обычно локали в ОС задаются в формате «язык\_территория.кодировка». Например, `ru_RU.UTF8` определяет локаль с русским языком (ru), на котором говорят в России (RU), и кодировкой UTF8. В Windows используются развернутые имена: `Russian_Russia.1251`.

Язык и территория определяют такие национальные особенности, как порядок символов, формат даты, разделитель десятичных разрядов и т. п.

Иногда бывает необходимо комбинировать поведение некоторых функций из разных локалей. Например, вместе с правилами сортировки русского языка использовать английские сообщения сервера. PostgreSQL поддерживает раздельную установку категорий локали через одноименные параметры конфигурации.

## Локали и категории

В операционной системе сервера должны быть установлены локали с поддержкой русского языка:

```
student$ locale -a | grep ru

ru_RU.koi8r
ru_RU.utf8
russian
```

Параметры локализации сеанса ОС определяются переменными окружения:

```
student$ locale

LANG=en_US.UTF-8
LANGUAGE=en_US
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC=ru_RU.UTF-8
LC_TIME=ru_RU.UTF-8
LC_COLLATE="en_US.UTF-8"
LC_MONETARY=ru_RU.UTF-8
LC_MESSAGES=en_US.UTF8
LC_PAPER=ru_RU.UTF-8
LC_NAME=ru_RU.UTF-8
LC_ADDRESS=ru_RU.UTF-8
LC_TELEPHONE=ru_RU.UTF-8
LC_MEASUREMENT=ru_RU.UTF-8
LC_IDENTIFICATION=ru_RU.UTF-8
LC_ALL=
```

Помимо переменных, соответствующих категориям локали, здесь также присутствуют переменные, имеющие особое значение:

- LC\_ALL — если установлена, то используется для всех категорий локали, даже если они заданы;
- LANG — используется для тех категорий локали, для которых значение не задано;
- LANGUAGE — если установлена, то используется вместо LC\_MESSAGES (об этом далее).

---

При установке PostgreSQL из пакета в виртуальной машине курса для инициализации кластера main были использованы параметры сеанса ОС. Если нужно задать другие значения, можно использовать ключ `--locale` (эквивалентен LANG), а также ключи для каждой категории, например так:

```
student$ pg_createcluster 13 new_cluster --locale=ru_RU.UTF-8 --lc-messages=en_US.UTF-8
```

Шесть параметров PostgreSQL соответствуют одноименным категориям локали. Видно, что их меньше чем в ОС:

```
=> SELECT name, setting, context, sourcefile
FROM pg_settings
WHERE name LIKE 'lc%';
```

name	setting	context	sourcefile
lc_collate	en_US.UTF-8	internal	
lc_ctype	en_US.UTF-8	internal	
lc_messages	en_US.UTF-8	superuser	/etc/postgresql/13/main/postgresql.conf
lc_monetary	ru_RU.UTF-8	user	/etc/postgresql/13/main/postgresql.conf
lc_numeric	ru_RU.UTF-8	user	/etc/postgresql/13/main/postgresql.conf
lc_time	ru_RU.UTF-8	user	/etc/postgresql/13/main/postgresql.conf

(6 rows)

---

Параметры `lc_ctype` и `lc_collate` задаются при инициализации кластера и изменить их невозможно. Значения остальных четырех параметров записываются в конфигурационный файл. При этом параметры `lc_monetary`, `lc_numeric` и `lc_time` может изменить любой пользователь (`context=user`), а `lc_messages` — только суперпользователь.

---

Каждый обслуживающий процесс при запуске сбрасывает значение LC\_ALL и устанавливает переменные окружения ОС для категорий локали в соответствии с конфигурационными параметрами "lc%".

При изменении параметра по ходу сеанса обслуживающий процесс устанавливает это же значение для соответствующей переменной окружения.

Стандартные базы `postgres`, `template0` и `template1` используют параметры `lc_collate` и `lc_ctype` кластера, кодировка баз также определяется этими параметрами.

Новые базы данных по умолчанию копируют параметры используемого шаблона:

```
=> CREATE DATABASE admin_localization_utf8;

CREATE DATABASE
```

```
=> \list admin_localization_utf8|template1
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
admin_localization_utf8	student	UTF8	en_US.UTF-8	en_US.UTF-8	
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + postgres=CTc/postgres

(2 rows)

## Даты и LC\_TIME

Параметр конфигурации, отвечающий за соответствующую категорию локали:

```
=> SET lc_time = 'ru_RU.UTF8';
```

SET

Для использования названий месяцев и дней недели на русском языке в форматной маске функции to\_char используется префикс TM:

```
=> SELECT to_char(current_date, 'TMDay, DD TMmonth YYYY');
```

```
to_char
-----
Четверг, 19 января 2023
(1 row)
```

То же самое для американского английского:

```
=> SET lc_time = 'en_US.UTF8';
```

SET

```
=> SELECT to_char(current_date, 'TMDay, DD TMMonth YYYY');
```

```
to_char
-----
Thursday, 19 January 2023
(1 row)
```

Функции to\_date/to\_char позволяют в явном виде указать формат даты и времени при вводе/выводе значений. На неявное преобразование даты в текст и наоборот влияет параметр datestyle. Вот несколько примеров:

```
=> SET datestyle = 'Postgres'; SELECT current_setting('datestyle') AS datestyle, now();
```

SET

```
datestyle | now
-----+-----
Postgres, DMY | Thu 19 Jan 14:34:39.341323 2023 MSK
(1 row)
```

```
=> SET datestyle = 'SQL, DMY'; SELECT current_setting('datestyle') AS datestyle, now();
```

SET

```
datestyle | now
-----+-----
SQL, DMY | 19/01/2023 14:34:39.393098 MSK
(1 row)
```

```
=> SET datestyle = 'SQL, MDY'; SELECT current_setting('datestyle') AS datestyle, now();
```

SET

```
datestyle | now
-----+-----
SQL, MDY | 01/19/2023 14:34:39.44334 MSK
(1 row)
```

```
=> RESET datestyle; SELECT current_setting('datestyle') AS datestyle, now();
```

RESET

```
datestyle | now
-----+-----
ISO, DMY | 2023-01-19 14:34:39.492903+03
(1 row)
```

## Числа и LC\_NUMERIC

Параметр конфигурации, отвечающий за соответствующую категорию локали:

```
=> SET lc_numeric = 'ru_RU.UTF8';
```

SET

Разделители групп разрядов (G), а также целой и дробной части (D), принятые в России:

```
=> SELECT to_char('12345'::numeric, '999G999D00');
```

```
to_char
-----
12 345,00
(1 row)
```

То же для США:

```
=> SET lc_numeric = 'en_US.UTF8';
```

SET

```
=> SELECT to_char('12345'::numeric, '999G999D00');
```

```
to_char
-----
12,345.00
(1 row)
```

---

## Денежные единицы и LC\_MONETARY

Параметр конфигурации, отвечающий за соответствующую категорию локали:

```
=> SET lc_monetary = 'ru_RU.UTF8';
```

SET

Денежный тип данных MONEY добавляет к сумме код валюты:

```
=> SELECT '12345'::money;
```

```
money
-----
12 345,00 ₺
(1 row)
```

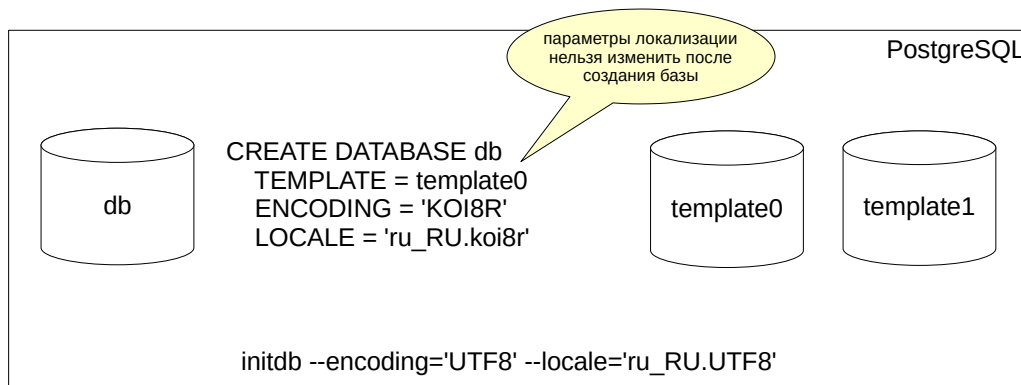
Но нужно учитывать, что в таблице с таким типом можно хранить только одну валюту, и та будет меняться вместе с LC\_MONETARY:

```
=> SET lc_monetary = 'en_US.UTF8';
```

SET

```
=> SELECT '12345'::money;
```

```
money
-----
$12,345.00
(1 row)
```



```
LC_COLLATE = en_US.UTF8
LC_CTYPE = en_US.UTF8
```

Установка локали выполняется при инициализации кластера баз данных. Утилита `initdb` использует локаль из окружения ОС, либо локаль можно указать явно через соответствующие параметры.

При создании новой базы данных из шаблона `template0` можно указать параметры локализации, отличные от тех, что использовались при инициализации кластера баз данных.

К основным параметрам локализации базы данных относятся: кодировка символов (`encoding`), а также категории локали `lc_ctype` и `lc_collate`, которые удобно задавать одним параметром (ключом) `locale`, потому что их значения обычно совпадают.

На эти категории локали накладываются ограничения:

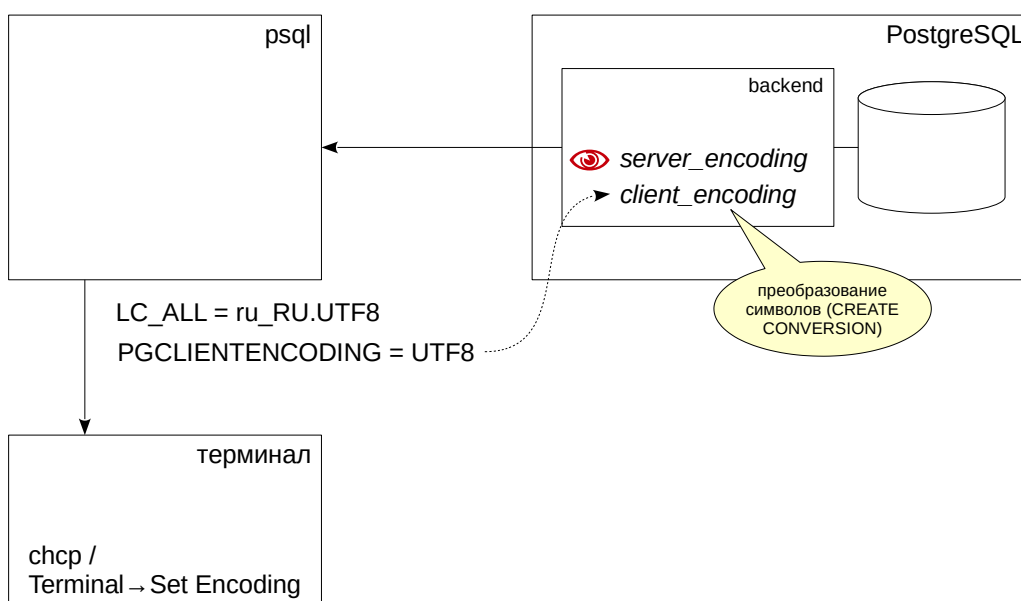
- кодировки символов у `locale` (`lc_ctype` и `lc_collate`) должны совпадать с кодировкой базы данных.
- после создания базы данных `locale` (`lc_ctype` и `lc_collate`) изменить нельзя.

Второе ограничение объясняется тем, что изменение правил сортировки и классификации символов может нарушить работу существующих индексов. Кроме того, может измениться поведение операций сравнения текстовых строк в ограничениях `CHECK` и табличных триггерах, что сделает уже сохраненные данные некорректными.

Именно поэтому, если нужно создать базу с отличающимися параметрами локализации, необходимо использовать шаблон `template0`, в котором гарантированно нет никаких данных.



# Настройка клиента



7

Для настройки локализации клиентского приложения нужно сделать следующее.

- Проверить, что настройки сервера корректны. Как минимум, что используется правильная кодировка БД (неизменяемый параметр *server\_encoding* покажет кодировку, заданную при создании БД).
- Проверить, что в клиентской ОС установлены нужные локали, и настроить категории локали (переменные среды LC\_\*) в сеансе ОС.
- Настроить кодировку, с которой работает приложение, и проверить настройки устройства вывода.  
Например, psql в Windows обычно использует кодировку Win-1251, поэтому в терминале (cmd.exe) необходимо выполнить команду `chcp 1251` и установить шрифт true type (например, Lucida Console).
- После подключения к БД проверить параметр *client\_encoding*. Этот параметр отвечает за перекодировку символов между клиентом и сервером. При необходимости установить в значение кодировки приложения. Значение *client\_encoding* можно задать и в переменной среды PGCLIENTENCODING.

Обслуживающий процесс автоматически перекодирует символы из кодировки БД (*server\_encoding*) в кодировку клиента (*client\_encoding*). Для большинства кодировок в PostgreSQL преднастроены необходимые процедуры преобразования символов: они находятся в таблице системного каталога `pg_conversion`. Возможно создание дополнительных пользовательских процедур (`CREATE CONVERSION`).

## Работа клиента и сервера в разных кодировках

Создадим базу данных с кодировкой KOI8R, а клиент по-прежнему будет использовать UTF8. Чтобы база данных успешно создавалась, нужно учесть следующее:

- в ОС должна быть установлена локаль с нужной кодировкой символов;
- нужно использовать шаблон template0: это пустая БД, индексы в которой гарантированно не пострадают в результате изменения кодировки.

Значения LC\_COLLATE и LC\_CTYPE, как правило, должны быть согласованы, поэтому обычно их устанавливают вместе одним параметром LOCALE:

```
=> CREATE DATABASE admin_localization_koi8r
      ENCODING = 'koi8r'
      LOCALE = 'ru_RU.koi8r'
      TEMPLATE = template0;
```

CREATE DATABASE

```
=> \l admin_localization_koi8r
```

```

                                List of databases
      Name      | Owner  | Encoding | Collate  | Ctype    | Access privileges
-----+-----+-----+-----+-----+-----
admin_localization_koi8r | student | KOI8R    | ru_RU.koi8r | ru_RU.koi8r |
(1 row)
```

Подключимся к admin\_localization\_koi8r и убедимся, что работает перекодировка символов.

```
=> \c admin_localization_koi8r
```

You are now connected to database "admin\_localization\_koi8r" as user "student".

Кодировки клиента и сервера:

```
=> SELECT name, setting, context
FROM pg_settings
WHERE name LIKE '%encoding';
```

```

      name      | setting | context
-----+-----+-----
client_encoding | KOI8R   | user
server_encoding | KOI8R   | internal
(2 rows)
```

Значение параметра client\_encoding устанавливает клиент, в данном случае psql. В интерактивном режиме значение устанавливается по кодировке клиента (переменной LC\_CTYPE в ОС). Но при запуске из скрипта, как сейчас в демонстрации, параметр устанавливается в значение кодировки базы данных.

Явно зададим такое же значение, как и у клиентской локали:

```
=> \! locale | grep LC_CTYPE
```

```
LC_CTYPE="en_US.UTF-8"
```

```
=> SET client_encoding = 'UTF-8';
```

SET

Теперь символы будут правильно конвертироваться из UTF8 в KOI8R и обратно:

```
=> CREATE TABLE tab AS SELECT 'Привет, мир! '::text AS col;
```

SELECT 1

```
=> SELECT * FROM tab;
```

```

      col
-----
Привет, мир!
(1 row)
```

Список доступных перекодировок для koi8:

```
=> \dcS *koi8*
```

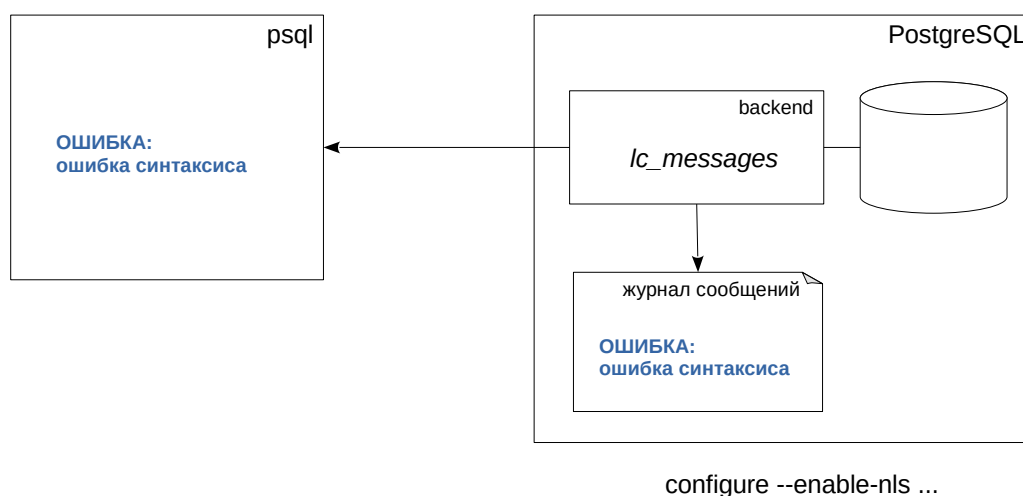
List of conversions				
Schema	Name	Source	Destination	Default?
pg_catalog	iso_8859_5_to_koi8_r	ISO_8859_5	KOI8R	yes
pg_catalog	koi8_r_to_iso_8859_5	KOI8R	ISO_8859_5	yes
pg_catalog	koi8_r_to_mic	KOI8R	MULE_INTERNAL	yes
pg_catalog	koi8_r_to_utf8	KOI8R	UTF8	yes
pg_catalog	koi8_r_to_windows_1251	KOI8R	WIN1251	yes
pg_catalog	koi8_r_to_windows_866	KOI8R	WIN866	yes
pg_catalog	koi8_u_to_utf8	KOI8U	UTF8	yes
pg_catalog	mic_to_koi8_r	MULE_INTERNAL	KOI8R	yes
pg_catalog	utf8_to_koi8_r	UTF8	KOI8R	yes
pg_catalog	utf8_to_koi8_u	UTF8	KOI8U	yes
pg_catalog	windows_1251_to_koi8_r	WIN1251	KOI8R	yes
pg_catalog	windows_866_to_koi8_r	WIN866	KOI8R	yes

(12 rows)

=> \c admin\_localization\_utf8

You are now connected to database "admin\_localization\_utf8" as user "student".

# Сообщения сервера



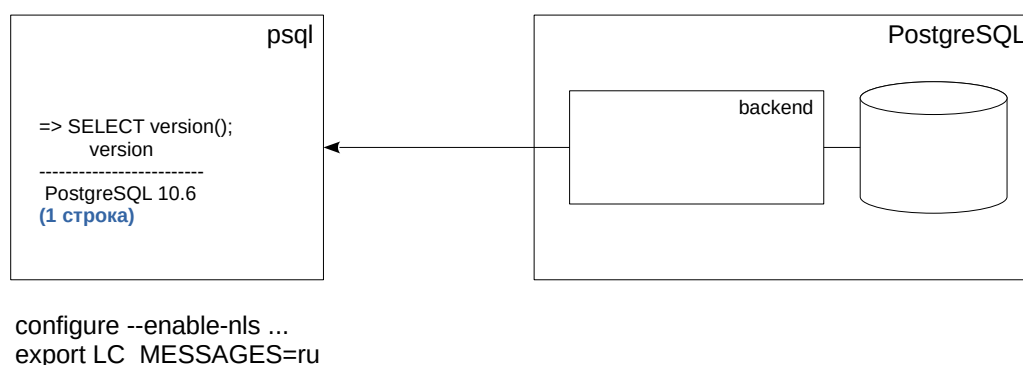
Сообщения сервера (и утилит) PostgreSQL переведены на несколько языков. В том числе и на русский.

Для того чтобы сообщения сервера выводились на русском языке, нужно, чтобы сервер PostgreSQL был собран с поддержкой национальных языков (`configure --enable-nls`).

Параметр конфигурации `lc_messages` управляет языком сообщений сервера.

Сообщения сервера не только отправляются клиенту, но и записываются в журнал сервера. При выборе языка, отличного от английского, следует убедиться, что инструменты работы с журналом сервера понимают этот язык. Например `pgBadger` может работать только с английскими сообщениями.

Подробнее о переводе сообщений сервера для переводчиков и разработчиков: <https://postgrespro.ru/docs/postgresql/13/nls>



Утилиты PostgreSQL (psql, pg\_dump и пр.) также поддерживают национальные языки.

Для того, чтобы сообщения утилит выводились на языке, отличном от английского, нужно, чтобы клиент PostgreSQL был собран с поддержкой NLS. Язык вывода устанавливается на клиенте переменной среды LC\_MESSAGES (параметр сервера *lc\_messages* влияет только на сообщения самого сервера, но не клиента).

Большинство ОС (включая Windows) используют следующий порядок просмотра переменных среды для языка сообщений: LANGUAGE, LC\_ALL, LC\_MESSAGES, LANG.

## Сообщения сервера и клиента

Сообщения сервера можно выводить на разных языках. Предварительно нужно убедиться, что PostgreSQL собран с поддержкой национальных языков:

```
=> SELECT unnest(regex_match(setting, '--enable-nls')) config
FROM pg_config()
WHERE name = 'CONFIGURE';

      config
-----
--enable-nls
(1 row)
```

Список доступных языков и файлы сообщений здесь:

```
=> SELECT setting
FROM pg_config()
WHERE name = 'LOCALEDIR';

      setting
-----
/usr/share/locale
(1 row)
```

```
postgres$ ls -C /usr/share/locale
```

aa	ca@valencia	fa_IR	is	mhr	pl	ti
ab	ce	ff	it	mi	pms	tig
ace	ch	fi	it_CARES	mjw	ps	tk
ach	chr	fil	iu	mk	pt	tl
af	ckb	fo	ja	ml	pt_BR	tr
ak	co	fr	jam	mn	ro	trv
am	crh	fr_CA	jv	mnw	ru	tt
an	cs	frp	ka	mo	rw	tt@iqtelif
ar	csb	fur	kab	mr	sa	tzm
arn	cv	fy	ki	ms	sc	ug
ary	cy	ga	kk	mt	sd	uk
as	da	gd	kl	my	sdh	ur
ast	de	gez	km	na	se	ur_PK
ay	de_CH	gl	kmr	nah	shn	uz
az	dv	gn	kn	nb	si	uz@Latn
ba	dz	gu	ko	nb_NO	sk	ve
bar	ee	gv	kok	nds	sl	vec
be	el	ha	ku	ne	so	vi
be@latin	en	haw	kv	nl	son	wa
bem	en_AU	he	kw	nn	sq	wae
be@tarask	en@boldquot	hi	ky	nso	sr	wal
bg	en_CA	hr	la	nv	sr@latin	wo
bi	en_GB	ht	lb	ny	sr@Latn	xh
bn	en@quot	hu	lg	oc	sv	yo
bn_BD	en@shaw	hy	ln	om	sw	zh_CN
bn_IN	eo	hy_AM	lo	or	szl	zh_Hans
bo	es	hye	locale.alias	os	ta	zh_Hant
br	et	ia	lt	pa	ta_LK	zh_HK
bs	eu	id	lv	pam	te	zh_LATN@pinyin
byn	fa	ie	mai	pap	tg	zh_TW
ca	fa_AF	io	mg	pi	th	zu

Для записи в журнал будем использовать ошибочную команду:

```
=> select1;

ERROR:  syntax error at or near "select1"
LINE 1: select1;
      ^
```

```
postgres$ tail -n 2 /var/log/postgresql/postgresql-13-main.log
```

```
2023-01-19 14:34:41.034 MSK [144237] student@admin_localization_utf8 ERROR:  syntax error at or near "select1" at character 1
2023-01-19 14:34:41.034 MSK [144237] student@admin_localization_utf8 STATEMENT:  select1;
```

В журнал сервера записывается такое же сообщение об ошибке и сама ошибочная команда.

Для вывода сообщений на русском языке надо изменить параметр `lc_messages`.

```
=> SET lc_messages TO 'ru_RU.UTF8';
```

```
SET
```

```
=> select1;
```

```
ОШИБКА:  ошибка синтаксиса (примерное положение: "select1")
LINE 1: select1;
      ^
```

```
postgres$ tail -n 2 /var/log/postgresql/postgresql-13-main.log
```

```
2023-01-19 14:34:41.153 MSK [144237] student@admin_localization_utf8 ОШИБКА:  ошибка синтаксиса (примерное положение: "select1") (символ 1)
2023-01-19 14:34:41.153 MSK [144237] student@admin_localization_utf8 ОПЕРАТОР:  select1;
```

Теперь сообщение выводится клиенту и в журнал на русском языке.

Посмотрим на сообщения, которые выводит клиент `psql`. Они остались на английском языке:

```
=> \dt
```

Did not find any relations.

Язык сообщений клиента, собранного с поддержкой национальных языков, определяется переменными ОС LC\_MESSAGES и LANGUAGE, причем LANGUAGE имеет приоритет:

```
=> \! echo $LC_MESSAGES
```

en\_US.UTF8

```
=> \! echo $LANGUAGE
```

en\_US

Для полной русификации установим русский язык и для сообщений psql.

```
=> \q
```

```
student$ export LANGUAGE=ru_RU
```

```
student$ psql -d admin_localization_utf8
```

```
=> \dt
```

Отношения не найдены.

```
=> SET lc_messages TO 'ru_RU.UTF8';
```

SET

Теперь все сообщения выводятся на русском языке.

## Объекты базы данных

- разные правила сортировки в одной БД
- начальное наполнение при создании БД
- указываются для столбцов таблиц и в выражениях

## Провайдеры

- libc
- ICU

## Специальные правила сортировки

- «C» и «POSIX»

Чтобы в одной базе данных можно было по-разному сортировать и сравнивать текстовые данные, в PostgreSQL существуют специальные объекты — правила сортировки (collation). Они позволяют использовать порядок сортировки, отличный от установленного по умолчанию для базы данных.

При создании правила сортировки указываются внешняя библиотека, реализующая сортировку (провайдер), и классификация символов. Начиная с PostgreSQL 10 можно выбирать между библиотеками ICU и libc; до этого всегда использовалась libc.

Начальный список объектов — правил сортировки — формируется при инициализации кластера. Для всех имеющихся в ОС локалей в базу данных загружаются правила сортировки. В дальнейшем при добавлении локалей в ОС можно дозагрузить правила.

Правила сортировки можно использовать при создании текстовых столбцов таблиц, при определении доменов и просто в выражениях, где сортируются или сравниваются текстовые строки.

Специальные правила сортировки «C» и «POSIX» работают одинаково и создаются для всех кодировок сервера. Для этих правил буквами будут считаться только латинские символы от A до Z, все остальные знаки будут сортироваться в соответствии со своими кодами в данной кодировке.

Посмотреть имеющиеся правила сортировки можно в таблице системного каталога pg\_collation.

<https://postgrespro.ru/docs/postgresql/13/collation>



## Порядок символов

зависит от реализации в операционной системе

## Изменения в библиотеке

отслеживается номер версии

## Использование

по умолчанию для базы данных или кластера

явное указание при определении столбцов и в выражениях

Правила сортировки провайдера libc в базе данных работают точно так же, как и в операционной системе. Однако библиотека libc может быть реализована по-разному в разных операционных системах. Как следствие, сортировка для одних и тех же локалей может отличаться в зависимости от ОС сервера базы данных. Если приложение должно поддерживать работу (включая логическую репликацию) СУБД на разных ОС, следует убедиться, что сортировка везде работает корректно.

Более серьезной потенциальной проблемой является установка новой версии библиотеки libc в ОС. Такое может произойти, например, при переходе на новый сервер с новой версией ОС. Изменение в новой версии libc используемых в базе данных правил сортировки может привести к некорректной работе индексов и другим проблемам.

## Порядок символов

зависит от версии библиотеки, но не от операционной системы

## Использование

явное указание при определении столбцов и в выражениях

нельзя использовать по умолчанию для базы данных или кластера

## Дополнительные возможности

управление порядком сортировки групп символов

Для использования провайдера ICU в определении правил сортировки необходимо, чтобы PostgreSQL был собран с поддержкой этой библиотеки. Библиотека ICU и реализованные в ней правила сортировки работают одинаково на всех операционных системах.

Библиотека ICU допускает видоизменение правил сортировки без смены языка и территории. Можно указать разный порядок сортировки для отдельных групп символов. Например, символы какой группы должны идти раньше: кириллица, латиница или цифры; буквы в верхнем регистре или в нижнем.

<https://icu.unicode.org/>

При инициализации кластера баз данных или при создании новой БД можно указать только локали библиотеки `libc`. Использовать в таком качестве библиотеку ICU можно, начиная с версии PostgreSQL 15.

## Правила сортировки

При инициализации кластера в системном каталоге каждой базы данных создаются специальные объекты — правила сортировки. Они создаются на основе информации об установленных в ОС локалях.

Начальное наполнение базы правилами сортировки:

```
=> SELECT CASE collprovider
      WHEN 'd' THEN 'default'
      WHEN 'c' THEN 'libc'
      WHEN 'i' THEN 'icu'
    END,
    count(*)
FROM pg_collation
GROUP BY collprovider;
```

case	count
icu	787
libc	45
default	1

(3 строки)

Правила сортировки для русского языка, полученные из ОС:

```
=> \d0S+ ru*
```

Список правил сортировки						
Схема	Имя	LC_COLLATE	LC_CTYPE	Провайдер	Детерминированное?	Описание
pg_catalog	ru-BY-x-icu	ru-BY	ru-BY	icu	да	Russian (Belarus)
pg_catalog	ru-KG-x-icu	ru-KG	ru-KG	icu	да	Russian (Kyrgyzstan)
pg_catalog	ru-KZ-x-icu	ru-KZ	ru-KZ	icu	да	Russian (Kazakhstan)
pg_catalog	ru-MD-x-icu	ru-MD	ru-MD	icu	да	Russian (Moldova)
pg_catalog	ru-RU-x-icu	ru-RU	ru-RU	icu	да	Russian (Russia)
pg_catalog	ru-UA-x-icu	ru-UA	ru-UA	icu	да	Russian (Ukraine)
pg_catalog	ru-x-icu	ru	ru	icu	да	Russian
pg_catalog	ru_RU	ru_RU.utf8	ru_RU.utf8	libc	да	
pg_catalog	ru_RU.utf8	ru_RU.utf8	ru_RU.utf8	libc	да	

(9 строк)

Правила сортировки для провайдера ICU создаются только в том случае, если сервер собран с параметром `--with-icu`:

```
=> SELECT unnest(regexp_match(setting, '--with-icu')) config
FROM pg_config()
WHERE name = 'CONFIGURE';

config
-----
--with-icu
(1 строка)
```

## Правила сортировки ICU

Сортировка для правила `ru-x-icu` отличается от используемой по умолчанию сортировки для правила `ru_RU`. Кириллица идет раньше латиницы:

```
=> WITH t(c) AS (
      VALUES ('a'), ('б'), ('в'), ('а'), ('б'), ('с'), ('1'), ('2'), ('3')
    )
SELECT string_agg(t.c, '' ORDER BY t.c) AS "default",
       string_agg(t.c, '' ORDER BY t.c COLLATE "ru-x-icu") AS "ru-x-icu"
FROM t \gx

-[ RECORD 1 ]-----
default | 1,2,3,a,b,c,a,b,v
ru-x-icu | 1,2,3,a,b,v,a,b,c
```

Ключевое слово `COLLATE` после выражения (в `ORDER BY`) явно задает правило сортировки. По умолчанию используется правило с именем `default`, соответствующее параметрам локали базы данных.

Возможности библиотеки ICU позволяют по-разному настраивать сортировку отдельных групп символов.

Создадим новое правило сортировки, где латинские символы идут раньше символов кириллицы:

```
=> CREATE COLLATION latn_cyrl
      (provider = icu, locale = 'ru-RU-u-kr-latn-cyrl');
```

CREATE COLLATION

Еще одно, где в дополнение к предыдущему буквы идут раньше цифр:

```
=> CREATE COLLATION latn_cyrl_digit
      (provider = icu, locale = 'ru-RU-u-kr-latn-cyrl-digit');
```

CREATE COLLATION

Проверяем сортировку:

```
=> WITH t(c) AS (
      VALUES('a'),('6'),('в'),('а'),('б'),('с'),('1'),('2'),('3')
    )
SELECT string_agg(t.c,', ' ORDER BY t.c) AS "default",
       string_agg(t.c,', ' ORDER BY t.c COLLATE "ru-x-icu") AS "ru-x-icu",
       string_agg(t.c,', ' ORDER BY t.c COLLATE "latn_cyrl") AS "latn_cyrl",
       string_agg(t.c,', ' ORDER BY t.c COLLATE "latn_cyrl_digit") AS "latn_cyrl_digit"
FROM t \gx
```

```
-[ RECORD 1 ]---+-----
default      | 1,2,3,a,b,c,a,б,в
ru-x-icu     | 1,2,3,a,б,в,a,b,c
latn_cyrl    | 1,2,3,a,b,c,a,б,в
latn_cyrl_digit | a,b,c,a,б,в,1,2,3
```

## Детерминированные правила сортировки

равенство строк при побайтовом совпадении

## Недетерминированные правила сортировки

строки, состоящие из разных байтов, могут быть равными

дополнительная гибкость

производительность ниже

не поддерживают LIKE и другие операции

Правило сортировки в зависимости от типа используемого сравнения может быть детерминированным или недетерминированным.

Детерминированные сравнения считают равными строки, состоящие из одних и тех же байтов. При недетерминированном сравнении даже не совпадающие побайтово строки могут считаться равными.

Например, в Unicode буква «ё» может быть представлена одним символом Cyrillic Small Letter Io (U+0451), а может быть составлена из буквы «е» (U+0435) и диакритического знака «диерезис» (U+0308).

Все стандартные правила — детерминированные. Пользовательские правила по умолчанию тоже будут детерминированными.

Недетерминированные правила дают дополнительную гибкость: можно, например, задать ограничение уникальности без учета регистра или диакритических символов. Но производительность детерминированных правил обычно выше. К тому же недетерминированные правила не поддерживают некоторые операции, такие как поиск по шаблону.

## Недетерминированные правила сортировки

Определим недетерминированное правило для регистронезависимой сортировки:

```
=> CREATE COLLATION ignore_case  
      (provider = icu, locale = 'und-u-ks-level2', deterministic = false);
```

CREATE COLLATION

Теперь строки в разных регистрах могут быть равными:

```
=> SELECT  
      'postgres' = 'POSTGRES' AS "default",  
      'postgres' = 'POSTGRES' COLLATE "ignore_case" AS "ignore_case"  
;  
  
 default | ignore_case  
-----+-----  
 f       | t  
(1 строка)
```

Но поиск по шаблону для недетерминированного правила невозможен:

```
=> SELECT 'PostgreSQL' LIKE 'post%' COLLATE "ignore_case";
```

ОШИБКА: недетерминированные правила сортировки не поддерживаются для LIKE

Изменение библиотеки-провайдера приводит к проблемам

неправильная сортировка значений в индексах  
сравнения строк в ограничениях целостности  
сравнения строк в запросах  
и т. д.

Номер версии библиотеки хранится в системном каталоге

При изменении установленной версии — предупреждение

`libc` не гарантирует стабильность и платформонезависимость версии

18

Если при изменении библиотеки-провайдера сравнение начинает работать иначе, это может повлиять на используемые в базе данных правила сортировки и вызвать проблемы:

- неправильную работу запросов из-за нарушения сортировки значений в индексах;
- некорректные данные из-за изменившегося поведения при сравнении строк в ограничениях целостности CHECK и триггерах;
- некорректные результаты из-за изменившегося поведения сравнения строк в запросах, процедурном коде и политиках защиты строк.

Поэтому при использовании правил сортировки PostgreSQL будет предупреждать о возможных проблемах, связанных с изменением библиотек.

При создании правила текущий номер версии библиотеки сохраняется в системном каталоге. При каждом использовании правила сохраненный номер версии сверяется с текущим номером, и при обнаружении расхождения запросы будут выдавать предупреждения о несоответствии версий.

Если такая ситуация случилась, следует пересоздать индексы, использующие измененные правила сортировки, а также проверить все места, где используются правила сортировки.

Нужно помнить, что, в отличие от ICU, провайдер `libc` не гарантирует одинаковую работу сортировки даже при совпадении версий библиотеки.

## Изменения в библиотеках-провайдерах

При создании правила сортировки в системном каталоге сохраняется версия библиотеки, переданная операционной системой. Если в ОС изменится версия библиотеки, каждый раз при использовании правила будет выдаваться предупреждение о том, что версии не совпадают.

Проверить соответствие версий можно запросом:

```
=> SELECT c.collname,
       c.collversion AS version,
       pg_collation_actual_version(c.oid) AS actual_version
FROM pg_collation c
WHERE c.collname LIKE 'ru%';
```

collname	version	actual_version
ru-BY-x-icu	153.112.40	153.112.40
ru-KG-x-icu	153.112.40	153.112.40
ru-KZ-x-icu	153.112.40	153.112.40
ru-MD-x-icu	153.112.40	153.112.40
ru-RU-x-icu	153.112.40	153.112.40
ru-UA-x-icu	153.112.40	153.112.40
ru-x-icu	153.112.40	153.112.40
ru_RU	2.35	2.35
ru_RU	2.35	2.35
ru_RU.koi8r	2.35	2.35
ru_RU.utf8	2.35	2.35
russian	2.35	2.35

(12 строк)

Изменение правила сортировки может привести к некорректной работе индексов. Если подобное произошло, рекомендуется пересоздать индексы с новой версией правила сортировки. После этого можно обновить версию в системном каталоге командой ALTER COLLATION ... REFRESH VERSION и предупреждения перестанут выдаваться.

Запрос для получения списка объектов, которые зависят от требующих обновления правил сортировки:

```
=> SELECT pg_describe_object(refclassid, refobjid, refobjsubid) AS "Collation",
       pg_describe_object(classid, objid, objsubid) AS "Object"
FROM pg_depend d
JOIN pg_collation c ON refclassid = 'pg_collation'::regclass AND refobjid = c.oid
WHERE c.collversion <> pg_collation_actual_version(c.oid)
ORDER BY 1, 2;
```

Collation	Object
-----------	--------

(0 строк)



Перед инициализацией кластера нужные для СУБД локали должны быть установлены в ОС

Клиент и сервер могут работать в различных кодировках с автоматическим преобразованием символов

Сообщения сервера и утилит переведены на множество языков, включая русский

Правила сортировки используют внешние библиотеки, изменения в которых могут привести к повреждению данных и некорректным результатам

1. Перенос данных между базами в разных кодировках.  
Создайте базу данных с кодировкой KOI8R.  
Создайте таблицу и добавьте в нее строки, содержащие символы кириллицы.  
Сделайте копию базы данных утилитой `pg_dump`.  
Восстановите таблицу из копии в базу с кодировкой UTF8.
2. Получите номер сегодняшнего дня недели.  
Меняется ли номер дня недели в зависимости от настроек локализации?

21

1. Для создания БД в кодировке KOI8R:

- убедитесь, что в ОС установлена нужная локаль;
- в команде `CREATE DATABASE` используйте шаблон `template0` параметры `ENCODING` и `LOCALE`.

2. Для получения номера дня недели используйте функцию `to_char`.

Допустимые форматные маски даты:

<https://postgrespro.ru/docs/postgresql/13/functions-formatting#FUNCTIONS-FORMATTING-DATETIME-TABLE>

## 1. Кодировки базы данных

Проверим, что в ОС есть локали с кодировкой koi8:

```
=> \! locale -a | grep koi8
```

```
ru_RU.koi8r
```

Создаем базы данных с кодировками KOI8R и UTF8:

```
=> CREATE DATABASE admin_localization_koi8r
    TEMPLATE template0
    ENCODING 'koi8r'
    LOCALE 'ru_RU.koi8r';
```

```
CREATE DATABASE
```

```
=> CREATE DATABASE admin_localization_utf8;
```

```
CREATE DATABASE
```

```
=> \l admin_localization_*
```

```

                                List of databases
  Name          | Owner   | Encoding | Collate  | Ctype    | Access privileges
-----+-----+-----+-----+-----+-----
admin_localization_koi8r | student | KOI8R    | ru_RU.koi8r | ru_RU.koi8r |
admin_localization_utf8  | student | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(2 rows)
```

Подключаемся к базе с кодировкой KOI8R:

```
=> \c admin_localization_koi8r
```

You are now connected to database "admin\_localization\_koi8r" as user "student".

```
=> SET client_encoding = 'UTF8';
```

```
SET
```

Убедимся, что клиент и сервер используют разные кодировки:

```
=> SELECT name, setting
FROM pg_settings
WHERE name LIKE '%encoding';
```

```

  name          | setting
-----+-----
client_encoding | UTF8
server_encoding | KOI8R
(2 rows)
```

Создаем таблицу, содержащую строки с кириллицей:

```
=> CREATE TABLE tab AS
    SELECT 'Привет, мир!' AS col;
```

```
SELECT 1
```

```
=> SELECT * FROM tab;
```

```

  col
-----
Привет, мир!
(1 row)
```

```
=> \q
```

Получаем логическую копию:

```
student$ pg_dump -d admin_localization_koi8r -Fc -f koi8r.dump
```

Содержимое копии выгружается в кодировке базы данных (KOI8R), а в начале файла есть команда установки параметра client\_encoding в то же значение KOI8R.

Восстанавливаем таблицу в базе данных admin\_localization\_utf8:

```
student$ pg_restore koi8r.dump -d admin_localization_utf8 -t tab
```

Благодаря установке `client_encoding` при восстановлении символы автоматически перекодируются. Проверим, что кириллица корректно перенесена:

```
student$ psql -d admin_localization_utf8
```

```
=> SELECT * FROM tab;
```

```
      col
-----
Привет, мир!
(1 row)
```

## 2. Номер сегодняшнего дня недели

Текущие настройки локализации даты и времени:

```
=> SHOW lc_time;
```

```
      lc_time
-----
ru_RU.UTF-8
(1 row)
```

Для получения номера дня недели есть две форматные маски:

- ID — неделя начинается с понедельника;
- D — неделя начинается с воскресенья.

```
=> SELECT to_char(current_date, 'TMDay: ID') AS "ID",
         to_char(current_date, 'TMDay: D') AS "D" ;
```

```
      ID      |      D
-----+-----
Четверг: 4   | Четверг: 5
(1 row)
```

Номер дня недели не зависит от настроек локализации, в частности, от параметра `lc_time`:

```
=> SET lc_time TO 'en_US.utf8';
```

```
SET
```

```
=> SELECT to_char(current_date, 'TMDay: ID') AS "ID",
         to_char(current_date, 'TMDay: D') AS "D" ;
```

```
      ID      |      D
-----+-----
Thursday: 4   | Thursday: 5
(1 row)
```