

Задачи администрирования Локализация



Авторские права

© Postgres Professional, 2016–2025

Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов, Игорь Гнатюк

Фото: Олег Бартунов (монастырь Пху и пик Бхрикути, Непал)

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

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

Провайдеры локалей

Настройка сервера

Работа с датами, числами, денежными единицами

Настройка клиента

Сообщения 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 выполняется 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/16/locale>

Локали

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

установлены в операционной системе

Категории локали и соответствующие им параметры

LC_COLLATE	—	правила сортировки символов
LC_CTYPE	—	классификация символов
LC_MESSAGES	<i>lc_messages</i>	язык сообщений
LC_MONETARY	<i>lc_monetary</i>	формат денежных единиц
LC_NUMERIC	<i>lc_numeric</i>	формат чисел
LC_TIME	<i>lc_time</i>	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_регион.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 нужно комбинировать поведение разных локалей. Например, вместе с форматами даты и времени, принятыми в России, использовать английские сообщения сервера. PostgreSQL поддерживает отдельную установку категорий локали через одноименные параметры конфигурации.

Для категорий локали LC_COLLATE и LC_CTYPE соответствующие неизменяемые параметры сервера удалены начиная с версии PostgreSQL 16; теперь за информацией о локали базы данных следует обращаться к таблице системного каталога pg_database.

Стандартная библиотека libc

соответствие POSIX

Внешняя библиотека ICU

переносимое решение

широкие дополнительные возможности

PostgreSQL реализует поддержку различных провайдеров локалей. Провайдеры — это библиотеки, предоставляющие данные локалей и реализующие сортировку. Стандартный провайдер libc использует системную библиотеку C и предоставляет ее локали. Эти локали используются большинством утилит операционной системы.

Также поддерживается провайдер ICU, использующий одноименную внешнюю библиотеку. Имена локалей в ICU строятся по правилам языковых меток IETF BCP 47. Такие метки имеют вид «язык-регион» или просто «язык» (например ru-RU). При использовании локалей ICU PostgreSQL проверяет имена и приводит их к каноническому виду в соответствии с правилами. Параметр *icu_validation_level* позволяет настроить уровень сообщений при проверках локалей.

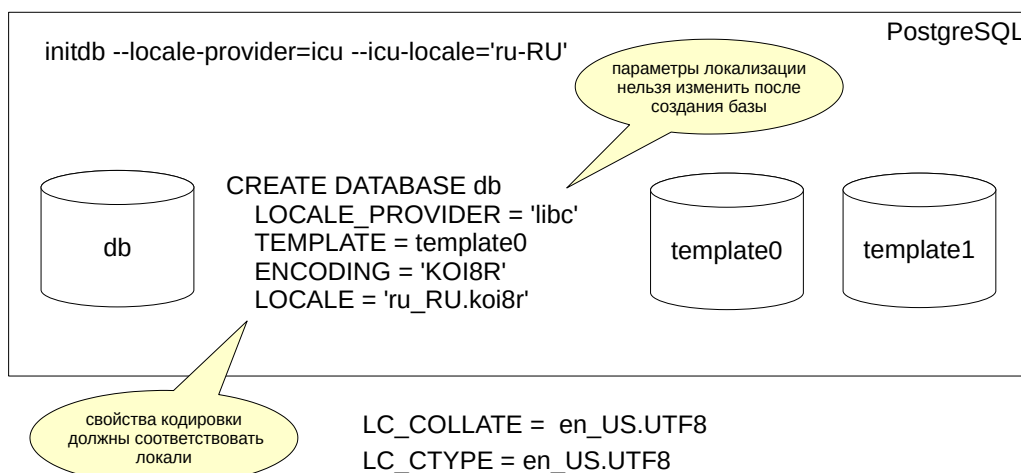
<https://postgrespro.ru/docs/postgresql/16/locale#ICU-LANGUAGE-TAG>

<https://www.rfc-editor.org/rfc/bcp/bcp47.txt>

Информация о выбранном провайдере и локали базы сохраняется в столбцах *pg_datlocprovider* и *datculocale* таблицы *pg_database*. По ряду причин настройки libc нужны во всех базах данных (даже использующих провайдер ICU), поэтому в столбцах *datcollate* и *datatype* всегда будет информация о локали libc.

Команды и утилиты для установки параметров локали позволяют явно выбрать и провайдера. По умолчанию используется провайдер libc.

<https://postgrespro.ru/docs/postgresql/16/locale#LOCALE-PROVIDERS>



При инициализации кластера провайдер и локаль шаблонных баз задаются с помощью ключей утилиты `initdb`. По умолчанию используется провайдер `libc` и локаль из окружения ОС.

При создании базы данных она наследует настройки локализации шаблона. Если требуется создать базу с другими параметрами локализации, нужно использовать шаблон `template0`, в котором гарантированно нет никаких данных, зависящих от локали. При этом можно сменить провайдера, кодировку символов, а также категории локали `LC_CTYPE` и `LC_COLLATE`. Значения для категорий обычно совпадают, поэтому их удобно задавать одним параметром (ключом): для `libc` — `LOCALE`, а для `ICU` — `ICU_LOCALE`.

При этом кодировка базы данных должна быть совместима с параметрами локали `LC_CTYPE` и `LC_COLLATE` этой базы данных. Локали C и POSIX совместимы с любым набором символов, а для остальных локалей провайдера `libc` есть только один набор символов, который будет работать правильно. (Исключение: в среде Windows кодировка UTF-8 может использоваться с любой локалью.) Локали провайдера `ICU` можно использовать с большинством кодировок.

Локали и категории

В операционной системе сервера уже установлены локали с поддержкой русского языка:

```
student$ locale -a | grep ru
```

```
ru_RU.koi8r
ru_RU.utf8
russian
ru_UA.utf8
```

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.UTF-8"
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 и провайдер libc. Чтобы задать другие значения, можно использовать ключи `--locale-provider`, `--locale` (эквивалентен LANG), а также ключи для категорий, например так:

```
student$ sudo pg_createcluster 16 new_cluster -- \
--locale-provider=icu \
--locale=ru_RU.UTF-8 \
--lc-messages=en_US.UTF-8
```

Четыре параметра 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_messages	en_US.UTF-8	superuser	/etc/postgresql/16/main/postgresql.conf
lc_monetary	ru_RU.UTF-8	user	/etc/postgresql/16/main/postgresql.conf
lc_numeric	ru_RU.UTF-8	user	/etc/postgresql/16/main/postgresql.conf
lc_time	ru_RU.UTF-8	user	/etc/postgresql/16/main/postgresql.conf

(4 rows)

Значения этих 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. Для категорий локали `LC_COLLATE` и `LC_CTYPE` соответствующих серверных параметров нет.

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 значения `LC_COLLATE`, `LC_CTYPE` а также провайдера локали, установленные при инициализации кластера. Новые базы данных по умолчанию копируют параметры используемого шаблона:

```
=> CREATE DATABASE admin_localization_utf8;

CREATE DATABASE

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

Expanded display is on.

List of databases

```
-[ RECORD 1 ]-----+-----
Name           | admin_localization_utf8
Owner          | student
Encoding       | UTF8
Locale Provider | libc
Collate        | en_US.UTF-8
Ctype          | en_US.UTF-8
ICU Locale     |
ICU Rules      |
Access privileges |
-[ RECORD 2 ]-----+-----
Name           | template1
Owner          | postgres
Encoding       | UTF8
Locale Provider | libc
Collate        | en_US.UTF-8
Ctype          | en_US.UTF-8
ICU Locale     |
ICU Rules      |
Access privileges | =c/postgres          +
                  | postgres=CTc/postgres
```

Expanded display is off.

Даты и LC_TIME

Параметр конфигурации, отвечающий за соответствующую категорию локали:

```
=> SHOW lc_time;
```

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

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
-----
Понедельник, 23 июня 2025
(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
-----
Monday, 23 June 2025
(1 row)
```

Вернем параметр к исходному значению:

```
=> RESET lc_time;
```

```
RESET
```

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 | Mon 23 Jun 23:49:47.021097 2025 MSK
(1 row)
```

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

```
SET
  datestyle | now
-----+-----
SQL, DMY | 23/06/2025 23:49:47.095668 MSK
(1 row)
```

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

```
SET
  datestyle | now
-----+-----
SQL, MDY | 06/23/2025 23:49:47.14828 MSK
(1 row)
```

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

```
RESET
  datestyle | now
-----+-----
ISO, DMY | 2025-06-23 23:49:47.201193+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:

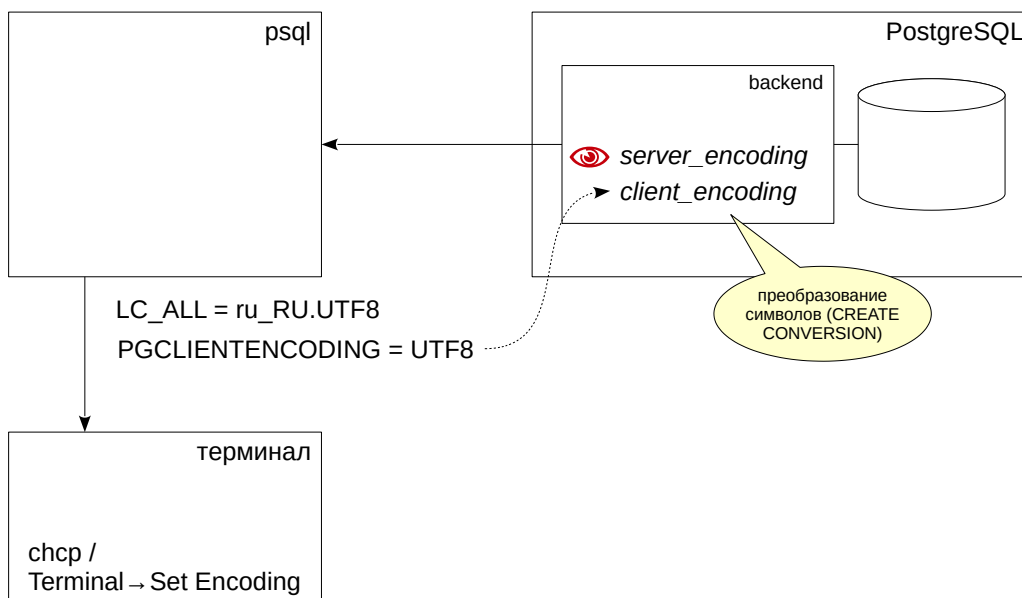
```
=> SET lc_monetary = 'en_US.UTF8';
```

SET

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

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

Настройка клиента



8

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 между кодировкой БД (*server_encoding*) и кодировкой клиента (*client_encoding*). Для большинства пар кодировок в PostgreSQL преднастроены необходимые процедуры преобразования символов: они находятся в таблице системного каталога *pg_conversion*. Можно определить дополнительную процедуру перекодировки (*CREATE CONVERSION*), однако на практике это вряд ли потребуется.

Работа клиента и сервера в разных кодировках

Создадим базу данных с кодировкой KOI8R, а клиент по-прежнему будет использовать UTF8. В качестве провайдера локалей укажем ICU. Чтобы база данных успешно создавалась, нужно использовать шаблон template0: это пустая БД, индексы в которой гарантированно не пострадают в результате изменения кодировки:

```
=> CREATE DATABASE admin_localization_koi8r
      LOCALE_PROVIDER = 'icu'
      ENCODING = 'koi8r'
      ICU_LOCALE = 'ru-RU' LC_CTYPE='ru_RU.koi8r' LC_COLLATE='ru_RU.koi8r'
      TEMPLATE = template0;
```

CREATE DATABASE

В нашем случае (при создании базы с провайдером ICU) нам понадобилось указать параметры LC_CTYPE и LC_COLLATE для соответствия устанавливаемой кодировке.

```
=> \x \l admin_localization_koi8r \x
```

Expanded display is on.

List of databases

```
-[ RECORD 1 ]-----+-----
Name          | admin_localization_koi8r
Owner         | student
Encoding      | KOI8R
Locale Provider | icu
Collate       | ru_RU.koi8r
Ctype        | ru_RU.koi8r
ICU Locale    | ru-RU
ICU Rules     |
Access privileges |
```

Expanded display is off.

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*

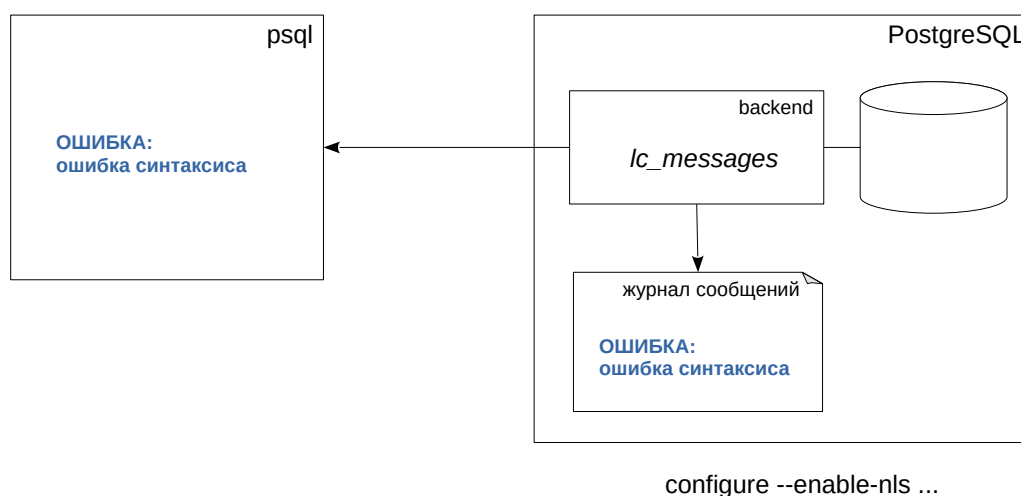
List of conversions				
Schema	Name	Source	Destination	Default?
pg_catalog	iso_8859_5_to_koi8_r	ISO_8859_5	KOI8R	yes
pg_catalog	koi8_r_to_iso_8859_5	KOI8R	ISO_8859_5	yes
pg_catalog	koi8_r_to_mic	KOI8R	MULE_INTERNAL	yes
pg_catalog	koi8_r_to_utf8	KOI8R	UTF8	yes
pg_catalog	koi8_r_to_windows_1251	KOI8R	WIN1251	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



10

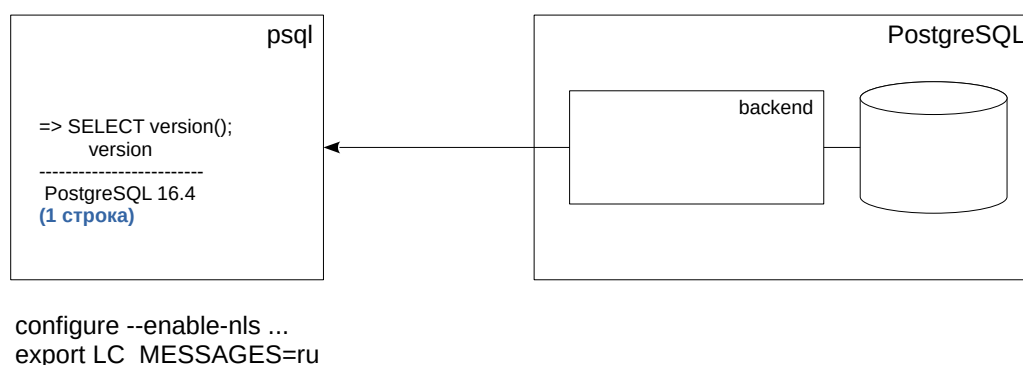
Сообщения сервера (и утилит) PostgreSQL переведены на несколько языков, в том числе и на русский.

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

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/16/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 (обычно это так). 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(regexp_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)
```

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

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

Для записи в журнал будем использовать ошибочную команду:

```
=> select1;
```

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

```
student$ tail -n 2 /var/log/postgresql/postgresql-16-main.log
```

```
2025-06-23 23:49:53.021 MSK [33603] student@admin_localization_utf8 ERROR: syntax error
at or near "select1" at character 1
2025-06-23 23:49:53.021 MSK [33603] 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;
```

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

```
student$ tail -n 2 /var/log/postgresql/postgresql-16-main.log
```

```
2025-06-23 23:49:53.157 MSK [33603] student@admin_localization_utf8 ERROR: syntax error
at or near "select1" at character 1
2025-06-23 23:49:53.157 MSK [33603] student@admin_localization_utf8 STATEMENT: 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
```

```
=> \! 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
```

```
Did not find any relations.
```

```
=> 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

Использование правил

- кластер
- база данных
- столбец таблицы
- выражение

Стандартные правила сортировки

- default, C, POSIX
- unicode, ucs_basic

Чтобы в одной базе данных можно было по-разному сортировать и сравнивать текстовые данные, в PostgreSQL существуют специальные объекты — правила сортировки (collation).

Правила сортировки позволяют использовать порядок сортировки, отличный от установленного по умолчанию для базы данных. Их можно использовать для столбцов таблиц, в определении доменов и просто в выражениях, где сортируются или сравниваются символьные строки. При создании нового правила сортировки указываются провайдер и классификация символов.

Начальный список правил сортировки формируется при инициализации кластера: для всех имеющихся в ОС локалей загружаются их правила сортировки. В дальнейшем при добавлении локалей в ОС можно дозагрузить правила.

Стандартные правила сортировки C и POSIX работают одинаково и создаются для всех кодировок сервера. Для этих правил буквами считаются латинские символы от A до Z, а остальные сортируются в соответствии со своими кодами в данной кодировке. Правило default использует значения LC_COLLATE, LC_CTYPE и провайдера, заданных при создании базы данных. Также доступны правила сортировки unicode и ucs_basic, определенные в стандарте SQL.

Все имеющиеся правила сортировки можно увидеть в таблице системного каталога pg_collation.

<https://postgrespro.ru/docs/postgresql/16/collation>

Порядок символов

зависит от реализации в операционной системе
отслеживается версия библиотеки

Используется по умолчанию

для кластера и базы данных

Правила сортировки провайдера libc в PostgreSQL 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
отслеживается версия библиотеки

Дополнительные возможности

управление порядком сортировки групп символов
дополнительные (особые) правила сортировки

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. Для настройки этих правил можно включить в языковую метку соответствующие параметры.

Дополнительные правила сортировки задаются особым синтаксисом, их примеры приведены в демонстрации.

<https://postgrespro.ru/docs/postgresql/16/collation#ICU-CUSTOM-COLLATIONS>

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

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	853
libc	47
default	1

(3 rows)

Правила сортировки для русского языка, полученные из ОС:

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

Schema	Name	Provider	Collate	Ctype	ICU Locale	ICU Rules
Deterministic?	Description					
pg_catalog	ru-BY-x-icu	icu			ru-BY	
yes	Russian (Belarus)					
pg_catalog	ru-KG-x-icu	icu			ru-KG	
yes	Russian (Kyrgyzstan)					
pg_catalog	ru-KZ-x-icu	icu			ru-KZ	
yes	Russian (Kazakhstan)					
pg_catalog	ru-MD-x-icu	icu			ru-MD	
yes	Russian (Moldova)					
pg_catalog	ru-RU-x-icu	icu			ru-RU	
yes	Russian (Russia)					
pg_catalog	ru-UA-x-icu	icu			ru-UA	
yes	Russian (Ukraine)					
pg_catalog	ru-x-icu	icu			ru	
yes	Russian					
pg_catalog	ru_RU	libc	ru_RU.utf8	ru_RU.utf8		
yes						
pg_catalog	ru_RU.utf8	libc	ru_RU.utf8	ru_RU.utf8		
yes						
pg_catalog	ru_UA	libc	ru_UA.utf8	ru_UA.utf8		
yes						
pg_catalog	ru_UA.utf8	libc	ru_UA.utf8	ru_UA.utf8		
yes						

(11 rows)

Обратите внимание, что отображаются только правила сортировки, применимые к кодировке текущей базы данных, поэтому для разных баз одного экземпляра PostgreSQL результат команды может отличаться.

Правила сортировки для провайдера ICU создаются только в том случае, если сервер собран с параметром `--with-icu`. Для таких правил сортировки формируются имена, включающие метку языка и указание суффикса «-x-icu» для отличия от локалей libc.

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

config
--with-icu

(1 row)

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'),('6'),('b'),('a'),('b'),('c'),('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,b
ru-x-icu | 1,2,3,a,b,b,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 путем определения новых правил с параметрами, заданными в виде части языковой метки.

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'),('b'),('a'),('b'),('c'),('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,b,b
ru-x-icu     | 1,2,3,a,b,b,a,b,c
latn_cyrl    | 1,2,3,a,b,c,a,b,b
latn_cyrl_digit | a,b,c,a,b,b,1,2,3
```

Если вышеуказанных параметров правил сортировки недостаточно, порядок элементов можно изменить с помощью особых правил сортировки:

```
=> CREATE COLLATION colors (
  provider = icu, locale = 'ru', rules = '& K < 0 < Ж < З < Г < С < Ф'
);
```

CREATE COLLATION

```
=> WITH colors(name) AS (
  VALUES ('Каждый'), ('Фазан'), ('Знать'), ('Желает'), ('Где'), ('Охотник'), ('Сидит')
)
SELECT * FROM colors ORDER BY name COLLATE colors;
```

```
name
-----
Каждый
Охотник
Желает
Знать
Где
Сидит
Фазан
(7 rows)
```

И еще один пример. Обратите внимание на синтаксис правила:

```

=> CREATE COLLATION verdicts (
    provider = icu, locale = 'ru', rules = $$ & 'к' < 'н' < ',' < 'п' $$
);

CREATE COLLATION

=> SELECT string_agg(w, ' ' ORDER BY w COLLATE verdicts) FROM
    (VALUES ('казнить'), (','), ('нельзя'), ('помиловать')) AS v(w);

    string_agg
-----
казнить нельзя , помиловать
(1 row)

```

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 row)
```

Но поиск по шаблону для недетерминированного правила невозможен:

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

```
ERROR: nondeterministic collations are not supported for 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

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, в таблицах `pg_collation` и `pg_database`. При каждом использовании правила сохраненный номер версии сверяется с текущим номером, и при их несоответствии запросы будут выдавать предупреждения.

Если обнаружено расхождение, следует пересоздать индексы, использующие измененные правила сортировки, а также проверить все места, где используются эти правила.

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,
       c.collprovider
FROM pg_collation c
WHERE c.collname LIKE 'ru%';
```

collname	version	actual_version	collprovider
ru_RU.koi8r	2.39	2.39	c
ru_RU.utf8	2.39	2.39	c
russian	2.39	2.39	c
ru-UA.utf8	2.39	2.39	c
ru_RU	2.39	2.39	c
ru_RU	2.39	2.39	c
ru-UA	2.39	2.39	c
ru-x-icu	153.121.44.8	153.121.44.8	i
ru-BY-x-icu	153.121.44.8	153.121.44.8	i
ru-KG-x-icu	153.121.44.8	153.121.44.8	i
ru-KZ-x-icu	153.121.44.8	153.121.44.8	i
ru-MD-x-icu	153.121.44.8	153.121.44.8	i
ru-RU-x-icu	153.121.44.8	153.121.44.8	i
ru-UA-x-icu	153.121.44.8	153.121.44.8	i

(14 rows)

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 rows)

Такая же проверка выполняется и для правил сортировки на уровне баз данных. Функция pg_database_collation_actual_version показывает актуальную версию правила сортировки в операционной системе:

```
=> SELECT d.datname,
       d.datcollversion AS version,
       pg_database_collation_actual_version(d.oid) AS actual_version,
       datlocprovider
FROM pg_database d;
```

datname	version	actual_version	datlocprovider
postgres	2.39	2.39	c
student	2.39	2.39	c
template1	2.39	2.39	c
template0		2.39	c
admin_localization_utf8	2.39	2.39	c
admin_localization_koi8r	153.121.44.8	153.121.44.8	i

(6 rows)

Если версия правила, заданного для базы данных, не совпадает с версией библиотеки в ОС, при подключении к базе данных выдается предупреждение. В этом случае рекомендуется перестроить индексы, использующие правило default. После этого можно обновить номер версии, хранящийся в pg_database, командой ALTER DATABASE ... REFRESH COLLATION VERSION.

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?

22

1. Для создания БД в кодировке KOI8R:

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

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

Допустимые форматные маски даты:

<https://postgrespro.ru/docs/postgresql/16/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
```

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

Expanded display is on.

List of databases

```
-[ RECORD 1 ]-----+-----
Name          | admin_localization_koi8r
Owner         | student
Encoding      | KOI8R
Locale Provider | libc
Collate       | ru_RU.koi8r
Ctype        | ru_RU.koi8r
ICU Locale    |
ICU Rules     |
Access privileges |
-[ RECORD 2 ]-----+-----
Name          | admin_localization_utf8
Owner         | student
Encoding      | UTF8
Locale Provider | libc
Collate       | en_US.UTF-8
Ctype        | en_US.UTF-8
ICU Locale    |
ICU Rules     |
Access privileges |
```

Expanded display is off.

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
-----+-----
Вторник: 2 | Вторник: 3
(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
-----+-----
Tuesday: 2 | Tuesday: 3
(1 row)

```