СИСТЕМА
УПРАВЛЕНИЯ
БАЗАМИ
ДАННЫХ

ЛИНТЕР®

ЛИНТЕР БАСТИОН

ЛИНТЕР СТАНДАРТ

# Командный интерфейс

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

РЕЛЭКС

## Товарные знаки

РЕЛЭКС™, ЛИНТЕР® являются товарными знаками, принадлежащими АО НПП «Реляционные экспертные системы» (далее по тексту – компания РЕЛЭКС). Прочие названия и обозначения продуктов в документе являются товарными знаками их производителей, продавцов или разработчиков.

## Интеллектуальная собственность

Правообладателем продуктов ЛИНТЕР® является компания РЕЛЭКС (1990-2026). Все права защищены.

Данный документ является результатом интеллектуальной деятельности, права на который принадлежат компании РЕЛЭКС.

Все материалы данного документа, а также его части/разделы могут свободно размещаться на любых сетевых ресурсах при условии указания на них источника документа и активных ссылок на сайты компании РЕЛЭКС: relex.ru и linter.ru.

При использовании любого материала из данного документа несетевым/печатным изданием обязательно указание в этом издании источника материала и ссылок на сайты компании РЕЛЭКС: relex.ru и linter.ru.

Цитирование информации из данного документа в средствах массовой информации допускается при обязательном упоминании первоисточника информации и компании РЕЛЭКС.

Любое использование в коммерческих целях информации из данного документа, включая (но не ограничиваясь этим) воспроизведение, передачу, преобразование, сохранение в системе поиска информации, перевод на другой (в том числе компьютерный) язык в какой-либо форме, какими-либо средствами, электронными, механическими, магнитными, оптическими, химическими, ручными или иными, запрещено без предварительного письменного разрешения компании РЕЛЭКС.

## О документе

Материал, содержащийся в данном документе, прошел доскональную проверку, но компания РЕЛЭКС не гарантирует, что документ не содержит ошибок и пропусков, поэтому оставляет за собой право в любое время вносить в документ исправления и изменения, пересматривать и обновлять содержащуюся в нем информацию.

## Контактные данные

394006, Россия, г. Воронеж, ул. Бахметьева, 2Б.

Тел./факс: (473) 2-711-711, 2-778-333.

e-mail: info@linter.ru.

## Техническая поддержка

С целью повышения качества программного продукта ЛИНТЕР и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки пользовательских рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам в раздел Поддержка на сайте ЛИНТЕР.

# Содержание

**Предисловие** ..... 3
Назначение документа ..... 3
Для кого предназначен документ ..... 3
Необходимые предварительные знания ..... 3
Дополнительные документы ..... 3
**Назначение** ..... 4
**Условия выполнения** ..... 5
**Выполнение программы** ..... 6
Переменные окружения ..... 6
Запуск ..... 6
Ключи ..... 7
**Обработка SQL-операторов** ..... 11
Ввод ..... 11
Результаты ..... 13
Редактирование ..... 15
SQL-операторы с параметрами ..... 15
Общие правила ..... 15
Формат ввода параметров ..... 18
**Команды** ..... 20
! ..... 20
_ ..... 22
ALTER PROC ..... 23
BLOB ..... 24
BROWSE ..... 24
CODEPAGE ..... 27
COUNT ..... 28
CREATE PROC ..... 29
CREATE TRIG ..... 30
DBINFO ..... 31
DISFILL ..... 32
DISHEAD ..... 33
DISPLAY ..... 34
DISREP ..... 36
DISSEP ..... 37
DISTAIL ..... 38
ECHO ..... 39
EDIT ..... 41
EHEAD ..... 43
EXEC ..... 43
EXIT|QUIT ..... 44
FORMAT ..... 45
HEADER ..... 46
HELP ..... 47
HISTORY ..... 49
IGNORE ..... 50
LIST ..... 51
OPTIMISTIC ..... 52
OUTFILE ..... 52
PAGE ..... 54
PESSIMISTIC ..... 56
PRECOUNT ..... 56
PRIORITY ..... 57
RESULT ..... 58

SH ..... 59
SHOW ..... 60
SLEEP ..... 62
STRUCTURE TABLE ..... 63
TIME ..... 64
UNLOAD ..... 66
USERNAME ..... 67
Работа с BLOB-данными ..... 68
Командный режим ..... 68
Табличный режим ..... 72
Автодополнение ввода ..... 73
**Сообщения программы** ..... 74
Информационные сообщения ..... 74
Диагностические сообщения ..... 74
Пользовательские ошибки ..... 74
Системные ошибки ..... 74
Коды завершения ..... 75

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

# Предисловие

## Назначение документа

Документ содержит описание возможностей программы `inl`, реализующей командный интерфейс пользователя с СУБД ЛИНТЕР. Командный интерфейс обеспечивается для всех программных платформ, на которых функционирует СУБД ЛИНТЕР.

Приводится описание управляющих, информационных команд и команд настройки интерфейса.

Документ предназначен для СУБД ЛИНТЕР БАСТИОН 6.0 сборка 20.8, далее по тексту СУБД ЛИНТЕР.

## Для кого предназначен документ

Документ предназначен для программистов и профессиональных пользователей СУБД ЛИНТЕР.

## Необходимые предварительные знания

Для работы с командным интерфейсом необходимо:

- знать основы реляционных баз данных;
- владеть языком баз данных SQL для СУБД ЛИНТЕР;
- уметь работать в соответствующей операционной системе на уровне простого пользователя.

## Дополнительные документы

- Библиотеки специальных типов данных
- Сетевые средства
- Справочник кодов завершения

# Назначение

Командный интерфейс может использоваться:

1) для проверки (отладки) синтаксиса и семантики SQL-запросов в процессе разработки клиентских приложений;
2) для получения и анализа временных характеристик выполняемых SQL-запросов;
3) для выполнения SQL-скриптов в интерактивном или пакетном режиме;
4) для форматирования и выдачи на консоль (или на печать) результатов поисковых SQL-запросов.

# Условия выполнения

Для выполнения `inl` необходимы следующие условия:

1) СУБД ЛИНТЕР должна быть активна (кроме запуска `inl` с ключом `-h`);
2) в момент запуска `inl` СУБД ЛИНТЕР должна иметь, в общем случае, три свободных канала;
3) 250 Мбайт свободной оперативной памяти;
4) пользователь, от имени которого запускается `inl`, должен быть зарегистрирован в БД, к которой осуществляется доступ.

# Выполнение программы

## Переменные окружения

Утилита `inl` распознает следующие переменные окружения:

- `LINTER_INLDEFCONNMODE` – режим обработки транзакций по умолчанию.

### Спецификация

```
LINTER_INLDEFCONNMODE=optimistic | pessimistic |exclusive
```

**Примечание**
Значение должно задаваться в нижнем регистре. Режим optimistic устарел (использовать не рекомендуется).

Если переменная окружения не задана, то действует `autocommit`;

- `LINTER_INLDATEOUT` – формат для вывода даты по умолчанию. Если задан, то для вывода даты используется указанный формат, иначе – формат по умолчанию. Спецификацию формата см. в документе «Библиотеки специальных типов данных», функция `TICKTOSTRF`;
- `LINTER_EDIT=<спецификация файла>`

  Определяет местонахождение и имя редактора скриптов, используемого утилитой `inl`. Если переменная LINTER_EDIT не определена, то по умолчанию в ОС типа Linux, ЗОСРВ Нейтрино используется редактор vi, а в ОС типа Windows – редактор Notepad.

## Запуск

Запуск программы осуществляется стандартными средствами запуска задач, имеющимися в каждой операционной системе. Исполняемый файл утилиты – `inl`.

### Командная строка

```
inl [-u <имя пользователя>[/[<пароль пользователя>]]]
[-n <имя сервера>]
[-f <спецификация файла>]
[-t]
[-p <спецификация файла>]
[-{ci|c} <кодировка интерфейса>]
[-s]
[-q]
[-{h|?}]
[-nohist]
[-es]
[-nf][-nc][-ns][-nl][-nh][-ia]
[-i <код завершения>]
[-sm <имя схемы>]
[-version]
[-briefversion]
```


© Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
[-dis]
[-onpromt]
```

По умолчанию для соединения с БД используется кодировка OEM.

# Ключи

Для передачи программе параметров используется набор ключей, позволяющих однозначно интерпретировать вид параметра. Все ключи имеют уникальное мнемоническое обозначение. Ключи в командной строке можно располагать в любой последовательности (за одним исключением: если используются оба ключа `-h` и `-c`, то ключ `-c` должен следовать первым, чтобы сообщения, выводимые ключом `-h`, отображались в указанной кодировке).

### Примечания

1. Команды и ключи допускается вводить как малыми, так и большими буквами.
2. При вводе значений ключей (имена, пароли, наименования таблиц и т.п.) малые и большие буквы различаются.
3. Если задан ключ, не относящийся к команде, то ошибка не фиксируется, а ключ программой не обрабатывается (игнорируется).
4. Признаком ключа является знак минус «-», альтернативный признак ключа «/» (обратная косая черта) допустим только в ОС типа Windows.
5. Для получения справочной информации о ключах программы необходимо задать в командной строке ключ `-?` или `-h`.
6. Имена файлов в командах и/или ключах должны задаваться полностью, расширения по умолчанию не допускаются.

Ниже перечислены используемые утилитой ключи с описанием их применения (таблица 1).

Таблица 1. Ключи утилиты inl

| Ключ | Описание |
|---|---|
| `-u` | Идентификация пользователя с аргументами `<имя пользователя>` и `<пароль пользователя>`. Если пароль не указан, он считается состоящим из одних пробелов |
| `-n` | Идентификация имени ЛИНТЕР-сервера. Имеет смысл только при работе в сети. Для подключения к удаленному ЛИНТЕР-серверу необходимо настроить сетевые средства СУБД ЛИНТЕР в соответствии с документом «Сетевые средства», раздел «Файл сетевой конфигурации». Имя <сервера> из ключа должно соответствовать имени одному из удаленных ЛИНТЕР-серверов, указанных в файле сетевой конфигурации СУБД ЛИНТЕР `nodetab` |
| `-f` | Определение имени файла, содержащего SQL-запрос (серию или пакет SQL-запросов), подлежащий выполнению |
| `-p` | При работе в пакетном режиме задает имя текстового файла с паролями пользователей.<br>Формат строк файла:<br>`<имя пользователя>/[<пароль>]` |

| Ключ | Описание |
|---|---|
| | Если <пароль> не задан, он будет запрошен интерактивно с консоли.<br>Особенности использования ключа `-p` одновременно с ключом `-u <имя пользователя>[/[<пароль пользователя>]]`<br>1) если после имени пользователя в ключе `-u` знак «/» присутствует, то подразумевается, что пароль задан (он будет пустым, если после знака «/» нет символов пароля), и в этом случае ключ `-p` игнорируется;<br>2) если после имени пользователя в ключе `-u` знака «/» нет, то пароль не задан, и тогда при наличии ключа `-p` пароль берется из файла.<br>Параметры файла паролей:<br>• максимальное количество строк: 100 (лишние строки игнорируются);<br>• максимальная длина строки: 100 байтов;<br>• кодировка текста: ASCII.<br>Ключ `-p` может задаваться не только в паре с ключом `-f`, но и отдельно, в частности, при вводе в командной строке имени пользователя без пароля.<br>Пример.<br>`inl -u SYSTEM/ -p pwd.txt`<br>Утилита `inl` выдаст сообщение о неверном пароле, если пароль у пользователя SYSTEM не пустой.<br>Если в файле `pwd.txt` задан пароль для пользователя SYSTEM – он будет взят из файла. Если нет, то будет запрошен интерактивно |
| `-t` | Отключение выдачи временной статистики (см. команду TIME) |
| `-s` | Запрещает выдачу на экран диагностического сообщения:<br>• кода завершения 2202 при удалении объекта из БД (то есть в БД нет удаляемого объекта);<br>• кода завершения 2310 (неизвестная глобальная переменная).<br>Игнорируются ошибки удаления `CHARACTER SET` и `TRANSLATION`.<br>На все другие диагностические сообщения, связанные с операцией удаления объекта (например, синтаксическая ошибка в SQL-запросе, отсутствие привилегий и т.п.) действие ключа не распространяется.<br>Также запрещает выдачу на экран диагностического сообщения с уведомлением об удалении несуществующих объектов |
| `-q` | Запрет выдачи информации об авторских правах |
| `-ia` | Заставляет игнорировать любой код завершения SQL-запроса и продолжать работу программы |
| `-i` | Включает игнорирование выдачи диагностического сообщения для указанного <кода завершения> (см. команду IGNORE). Ключей `-i`, включающих различные коды завершения, в SQL-скрипте может быть несколько, при этом ключ действует как переключатель: если <код завершения> еще не установлен, то он устанавливается, если уже установлен, то сбрасывается<br>**Примечание**<br>Список установленных по ключу `-i` кодов завершения можно просмотреть по команде LIST |


© Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

| Ключ | Описание |
|---|---|
| -sm | Задает имя схемы. При указании этого ключа сразу после открытия канала выполняется команда SET SCHEMA <имя схемы> (при возникновении ошибки выполнения этой команды канал закрывается) |
| -es | Заставляет выдавать на экран тексты запросов при ошибках, отличных от ошибок синтаксиса/логики |
| -nh | Запрет выдачи заголовка |
| -nc | Запрет выдачи итоговой статистики о выполненных SQL-запросах (число найденных/обработанных записей и длительность обработки запроса – см. команду COUNT) |
| -ns | Запрет выдачи разделителя столбцов (см. команду UNLOAD) |
| -nl | Запрет выдачи разделителя столбцов в начале и конце записей выборки данных (см. команду UNLOAD) |
| -nf | Запрещает дополнять выдаваемое значение до ширины столбца |
| -ci или -c | Задает кодовую страницу для интерфейса утилиты.<br>Если ключ не задан, по умолчанию используется язык операционной системы.<br>Если кодовая страница задана неверно или не установлена в ОС, используется англоязычный интерфейс.<br>Примеры:<br>-ci cp866 (русскоязычный интерфейс)<br>-c ENG (англоязычный интерфейс) |
| -h или -? | Выдает на экран справочную информацию и завершает выполнение программы. Если в командной строке заданы другие ключи, то они игнорируются |
| -nohist | Запрещает протоколирование выполняемых команд с целью последующего отображения по команде HISTORY<br><br>**Примечание**<br>Действует только в среде ОС типа Linux, ЗОСРВ Нейтрино |
| -dis | Задает игнорирование inl-команд BROWSE, EDIT, EXIT, QUIT, PAGE, SH |
| -onpromt | Заставляет выдавать подсказку SQL> при работе в потоковом режиме |
| -version | Полная информация о версии программы |
| -briefversion | Номер версии сборки программы |

## Принятые умолчания

1) Ключи регистронезависимы.
2) Если при запуске опущен ключ -u (вместе с аргументами), то прежде чем приступить к работе, inl запрашивает у пользователя имя и пароль.
3) Если опущен ключ -n, то inl будет работать с сервером по умолчанию.
4) При отсутствии ключа -f программа переходит в интерактивный режим работы.
5) При отсутствии ключа -t выдается время начала и окончания обработки каждого SQL-запроса.
6) Если ключ -c не задан, то берется значение переменной окружения LINTER_CP; если и она не задана, то возьмется национальная кодировка (locale), установленная в ОС.

## Примечания

1. При вводе пароля вводимые символы маскируются.
2. При запуске утилиты `inl` ей можно передавать команду для выполнения, например:

```
echo 'select * from auto;' | inl -u SYSTEM/MANAGER8
```

3. Команды с ключами `-version` и `-briefversion` могут выполняться при неактивном ядре СУБД ЛИНТЕР:

```
inl -version
inl -briefversion
```

### Пример

Для запуска программы:

1) набрать имя программы и нажать клавишу **<Enter>**:

```
inl <Enter>
```

2) появится заставка программы и приглашение для ввода имени пользователя:

```
Интерактивный SQL в.6.0 СУБД ЛИНТЕР в.6.0
Имя пользователя: SYSTEM
```

3) после ввода имени пользователя появится приглашение для ввода пароля пользователя:

```
Пароль пользователя:
```

4) если регистрационные параметры введены правильно, появится подсказка готовности к работе:

```
SQL>
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

# Обработка SQL-операторов

Выполнение SQL-операторов возможно в двух режимах: интерактивном и пакетном.

В интерактивном режиме текст SQL-оператора вводится с клавиатуры и сразу передается на обработку `inl`.

В пакетном режиме текст SQL-оператора предварительно записывается в текстовый файле (SQL-скрипте) с помощью любого текстового редактора имеющегося в операционной системе и поддерживающего тип файла `.txt`, либо с помощью `inl`, затем SQL-скрипт передается на обработку `inl`. SQL-скрипт может содержать любое количество SQL-операторов.

Для предоставления входных SQL-операторов и результатов из выполнения в требуемой кодировке надо:

- либо установить нужную кодировку с помощью переменной среды окружения `LINTER_CP`;
- либо предварительно выполнить SQL-оператор `SET NAMES <кодировка>`.

**Примечание**
`SET NAMES` влияет только на последующее выполнение команд по тому каналу, по которому этот запрос был подан, а установка `LINTER_CP` – на все последующие действия, кроме выполняемых по ранее открытым каналам и тем каналам, для которых впоследствии будет явно выполнен запрос `SET NAMES`.

# Ввод

Ввод SQL-операторов в интерактивном режиме выполняется по следующим правилам:

1) ввод SQL-оператора должен начинаться после приглашения `inl`;

```
SQL>select count(*) from auto;
```

2) если текст SQL-оператора не умещается на одной строке консоли, он может быть продолжен на следующих строках;
3) прекращение ввода текущей строки и переход на следующую строку выполняется по клавише **<Enter>**;
4) переход на следующую строку должен выполняться только на границах лексем SQL-оператора, то есть разрыв ключевых слов, имен таблиц, столбцов, идентификаторов не допускается;
5) продолжение ввода SQL-оператора должно выполняться после приглашения `inl` (выдается порядковый номер строк продолжения):

```
SQL> update DEPT_SUMMARY s
1>set NUM_EMPS = (
2>select count(1)
3>from EMP E
4>where E.DEPTNO = S.DEPTNO);
```

6) пробелы, введенные в начале и/или в конце текста SQL-оператора, игнорируются;
7) строки, состоящие полностью из пробелов и/или знаков табуляции, игнорируются;
8) текст SQL-оператора должен заканчиваться знаком «;» (в конце текущей строки либо на следующей строке продолжения);

9) неотображаемый на консоли комментарий SQL-скрипта задается:

- символом «;» в начале строки, например,

```
; Это неотображаемый комментарий
```

- после символов «//» в конце строки

```
Table AUTO; //Проверить выполнение;
```

> **Примечание**
> Символы «//» являются комментарием sql, а не комментарием `inl`.

- после символов «--» в конце строки

```
SELECT * FROM AUTO; -- Пример SQL-запроса;
```

10) строчный отображаемый на консоли комментарий SQL-скрипта задается двойным символом «–» в начале строки, например:

```
-- Это строчный отображаемый комментарий
```

или с помощью команды «! » утилиты `inl`:

```
! Это отображаемый комментарий
```

11) строчный комментарий:

- можно использовать только в отдельных строках или внутри SQL-запроса, но не после точки с запятой в той же строке, например,

```
SELECT count(*) /* количество записей */ FROM AUTO;
```

- должен заканчиваться знаком «;», если он относится к законченному SQL-оператору;
- не должен заканчиваться знаком «;», если это не последний оператор текста хранимой процедуры.

В противном случае (если знак «;» отсутствует), будут проигнорированы все строки до первой последующей строки, которая заканчивается знаком «;» включительно, либо до конца входного файла;

12) ввод SQL-оператора `CREATE PROCEDURE` ... имеет свои особенности, связанные с тем, что тело хранимой процедуры задается на процедурном языке СУБД ЛИНТЕР, в котором разделение операторов также выполняется по символу ';'. Чтобы избежать коллизий и не начать преждевременное выполнение SQL-оператора при обнаружении во вводимом тексте символа ';', необходимо для всех таких операторов задавать признак продолжения '//' ввода тела хранимой процедуры, как, например, в этом примере:

```
SQL>create procedure Proc1(in p int) result int
1> code
2> return p+3; //
3> end;
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

# Результаты

В общем случае выдается следующая информация:

1) длительность обработки SQL-оператора (время начала и окончания обработки с точностью до минуты) – если установлен режим выдачи статистики (см. команду COUNT);
2) код завершения и его расшифровка, указывающие на причину, по которой ядро СУБД ЛИНТЕР не смогло выполнить SQL-оператор (см. документ «Справочник кодов завершения»), например:

```
INL: состояние выполнения: 1503
Таблица уже существует
```

В частных случаях дополнительно выдается следующая информация:

| SQL-оператор | Дополнительная информация |
|---|---|
| INSERT | Количество добавленных в таблицу записей<br>`INL: добавлено строк : nnn` |
| DELETE | Количество реально удаленных из таблицы записей<br>`INL: удалено строк : nnn` |
| UPDATE | Количество реально измененных в таблице записей<br>`INL: изменено строк : nnn` |
| SELECT | Количество реально выданных записей в виде<br>`INL: выдано строк : nnn` |
| GET EVENT | Текущее состояние указанного события<br>(событие установлено/событие не установлено) |
| TEST TABLE | Результаты тестирования таблицы |

Результаты поисковых запросов, выводимые по умолчанию, приведены в таблице 2.

Таблица 2. Отображение по умолчанию типов данных

| Тип данных | Представление по умолчанию |
|---|---|
| SMALLINT | Отрицательные значения выводятся со знаком '–', положительные – без знака. Для представления положительных чисел со знаком надо использовать функцию to_char(), например:<br>`select to_char(weight, 's99999') from auto;` |
| INT | Аналогично SMALLINT |
| BIGINT | Аналогично SMALLINT |
| CHAR | Ширина выводимого столбца всегда равна размеру столбца в таблице |
| VARCHAR | По умолчанию ширина выводимого столбца всегда равна размеру столбца в таблице. Если количество выбранных символов меньше размера столбца в таблице, то недостающие символы отображаются в виде пробелов справа.<br>Для отмены расширения столбца используется ключ `-nf` |
| BYTE | Ширина выводимого столбца всегда равна размеру столбца в таблице |
| VARBYTE | По умолчанию ширина выводимого столбца всегда равна размеру столбца в таблице. Если количество выбранных байтов меньше размера столбца в таблице, то недостающие байты отображаются в виде пробелов справа. |

| Тип данных | Представление по умолчанию |
|---|---|
| | Для отмены расширения столбца используется ключ `-nf` |
| `REAL` | Данные отображаются в формате [s]99999999.9999 без выравнивания по десятичной точке, незначащие нули справа и слева отбрасываются, например:<br>`\|  123.456\|`<br>`\| 123.4567\|`<br>`\|  -12.06\|`<br>`\|  0.6999\|`<br>`\|  45\|` |
| `DOUBLE` | Данные отображаются в формате [s]99999999999999.999999999 без выравнивания по десятичной точке, незначащие нули справа и слева отбрасываются, например:<br>`\|  123.456\|`<br>`\| 123.456789123\|`<br>`\|  - 23.406\|`<br>`\|  0.69\|`<br>`\|  678885\|` |
| `DECIMAL (NUMERIC)` | Данные отображаются в формате [s]9999999999999999999.9999999999 с выравниванием по десятичной точке, незначащие нули справа (кроме первого после точки) и слева отбрасываются, например:<br>`\|  123.456\|`<br>`\| 23.456789123\|`<br>`\|  12.06\|`<br>`\|  0.6999\|`<br>`\|  643.0\|`<br>`\| -796.00954\|` |
| `DATE` | Данные отображаются в формате дд.мм.гггг:чч:ми:сс:тт, например:<br>`24.08,2002:14:24:39:00` |
| `BOOLEAN` | Данные отображаются в формате T или F:<br>`T – true (0), F – false (1)` |
| `NCHAR` | Представление UNICODE-данных в текущей кодовой странице `inl` |
| `NCHAR VARYING` | Представление UNICODE-данных в текущей кодовой странице `inl`. Если количество выбранных символов меньше размера столбца в таблице, то недостающие символы отображаются в виде пробелов справа.<br>Для отмены расширения столбца используется ключ `-nf` |
| `BLOB` | Системный описатель BLOB-столбца в виде:<br>• размер BLOB-файла;<br>• номер первой порции;<br>• номер последней порции;<br>• число BLOB-файлов; |

| Тип данных | Представление по умолчанию |
|---|---|
| | • дд.мм.гггг дата – дата создания фразового индекса (если он создан) или нулевая дата – в противном случае;<br>• тип BLOB-данных (текст, графика, видео и т.п.), например:<br>`\|0000572006 000002 000143 1 22.09.2002 003\|` |
| `EXTFILE` | Спецификация внешнего файла (символьная строка длиной 512 символов) |

**Примечания**

1. Для просмотра данных типа `CHAR`, `VARCHAR`, `BYTE`, `VARBYTE`, `NCHAR`, `NCHAR VARYING` большой размерности рекомендуется использовать табличный режим (см. команду BROWSE).
2. При выводе символьного значения на терминал двоичные нули, символы табуляции, перевода строк, возврата каретки заменяются пробелами, другие непечатные символы – точками. При выводе в файл непечатные символьные значения записываются без изменений.

# Редактирование

Для редактирования текста SQL-оператора:

1) выполнить команду EDIT сразу после выполнения SQL-оператора, текст которого должен быть изменен;
2) откорректировать текст SQL-оператора;
3) выполнить команду EXEC для выполнения откорректированного SQL-оператора.

Для редактирования SQL-скрипта:

1) выполнить команду EDIT сразу после выполнения SQL-скрипта, текст которого должен быть изменен;
2) откорректировать текст SQL-скрипта;
3) сохранить, при необходимости, текст откорректированного SQL-скрипта в файле;
4) выполнить SQL-скрипт.

# SQL-операторы с параметрами

Утилита `inl` поддерживает обработку параметров для следующих SQL-операторов:

- SELECT;
- INSERT;
- DELETE;
- UPDATE;
- EXECUTE | CALL.

## Общие правила

Обработка SQL-операторов с параметрами выполняется по следующим правилам:

1) допустимыми являются как именованные параметры (:<имя параметра>), так и неименованные (?), например:

```
select make from auto where year=:year;
select make from auto where year>?;
```

2) SQL-оператор может включать одновременно именованные и неименованные параметры, например:

```
select make from auto where year between :beg_year and :end_year
 and color=?;
```

3) подсчет параметров ведется слева направо по тексту SQL-оператора, например:

```
select make from auto where year in (?,:year,?);
```

Порядок подсчета параметров SQL-оператора приведен на рисунке 1.

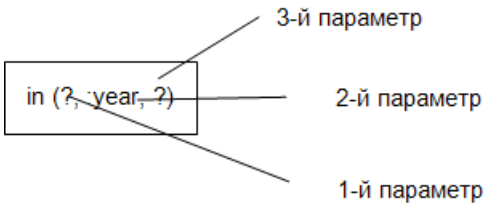



Рисунок 1. Порядок подсчета параметров SQL-оператора

4) общее количество уникальных именованных параметров в одном SQL-операторе – не больше 255;

5) длина значения вводимого параметра не должна превышать 4 Кбайт;

6) максимальная суммарная длина значений всех параметров – 16 Кбайт;

7) при обработке SQL-оператора с параметрами `inl` запрашивает ввод значения для каждого параметра. При этом для именованных параметров в качестве подсказки выдается имя и в скобках тип и, если необходимо, длина параметра, а для неименованных – порядковый номер и тип, например:

```
SQL>SELECT count(model) FROM AUTO WHERE COLOR=?
1>AND YEAR = :YEAR;
Параметр 1 (CHAR(10))>YELLOW
YEAR (INTEGER)>71
|            38|
INL : выдано строк: 1
SQL>select distinct make,year  from auto
1>where ?(char(4))= make;
SQL>insert into auto( personid,make,year,model) values
1>(?,:make,:year,:model);
SQL>delete from auto where year<= ?;
```

8) если в SQL-операторе есть несколько одноименных именованных параметров, то запрос на ввод значения для данного именованного параметра выполняется один раз, например:

```
SQL>select count(model) from auto where weight between
 1>:base_weight-500 and :base_weight+500;
BASE_WEIGHT (INTEGER)>1000
|          19|
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
SQL>exec
BASE_WEIGHT (INTEGER)>1500
|           54|
SQL>
```

9) присвоение параметру NULL-значения выполняется с помощью ввода строки NULL (все заглавные буквы). Для присвоения символьным строкам значения 'NULL' необходимо использовать отличную от строки NULL транслитерацию, например, null, Null;

```
SQL>update auto set color=? where personid=:person;
Параметр1 (CHAR(10))>NULL
PERSON (INTEGER)>34
SQL>exec
Параметр1 (CHAR(10))>null
PERSON (INTEGER)>35
SQL>select color from auto where personid in (34,35);
|                 |
|null             |
```

10) для хранимых процедур запрашивается ввод значений для всех типов параметров, однако введенное значение OUT-параметра игнорируется. Список параметров процедуры является позиционным, поэтому, если в списке параметров присутствует параметр типа OUT, то в SQL-операторе с параметрами он должен быть позиционирован либо как необрабатываемый (пропущенный), знаком «запятая» (в конце списка параметры типа OUT можно не задавать), либо как обычный параметр (но запрос на ввод значения такого параметра выдаваться не будет), например:

| Список параметров процедуры | SQL-оператор с параметрами |
|---|---|
| proc1 (in p1 int,out p2 int,inout p3 char(25),in p4 date); | execute proc1(?,,?,?);<br>execute proc1(?,?,?,?); |
| proc2 (out p1 int,out p2 int,inout p3 char(25),in p4 date); | execute proc2(,,:p3,?);<br>execute proc2(?,?,:p3,?); |
| proc3 (inout p1 int,in p2 int,out p3 char(25),out p4 date); | execute proc3(:p1,:p2); |

11) нельзя использовать параметры для столбцов типа `BLOB` и `EXTFILE`;

12) если SQL-оператор с параметрами представлен в виде SQL-скрипта, то значения параметров должны задаваться в этом же скрипте сразу после текста SQL-оператора, каждое значение на отдельной строке. Для именованных повторяющихся параметров значение должно задаваться один раз, например:

Текст SQL-скрипта:

```
SELECT model, "Подзапрос5".name
FROM AUTO,
(SELECT name, personid FROM person where sex=?) as "Подзапрос5"
where color=? and year=?
and auto.personid="Подзапрос5".personid;
M
YELLOW
70
```

13) значения булевских параметров можно вводить в верхнем регистре (TRUE, T, FALSE, F) и в нижнем (true, t, false, f);

14) в случае если SQL-транслятор не может самостоятельно определить тип параметра из контекста SQL-запроса, пользователь должен явно указывать типы параметров.

```
Правильные конструкции:
select concat (? (char(10)), ? (char(5)), :param (char(10)));

select ? (char(10)) + ? (char(5))|| :param (char(10));

Ошибочная конструкция:
select concat (?, :param);
```

# Формат ввода параметров

В таблице 3 приведены форматы ввода параметров.

Таблица 3. Форматы ввода параметров

| Тип параметра | Формат ввода |
|---|---|
| SMALLINT | [s]999999 |
| INT | [s]999999 |
| BIGINT | [s]999999 |
| CHAR | Символьная строка длиной не более 4000 знаков, символы вводятся в текущей кодовой странице `inl` и не преобразуются (двойные кавычки для отмены преобразования не требуются) |
| VARCHAR | Символьная строка длиной не более 4000 знаков, символы вводятся в текущей кодовой странице `inl` и не преобразуются (двойные кавычки для отмены преобразования не требуются) |
| BYTE | Символьная строка шестнадцатеричных цифр длиной не более 4000 байт. Каждый байт должен быть представлен двумя символами шестнадцатеричных цифр. Между парами шестнадцатеричных цифр допускается любое число пробелов. Внутри пары шестнадцатеричных цифр разделители не допускаются. Общее количество шестнадцатеричных цифр должно быть четным |
| VARBYTE | Символьная строка шестнадцатеричных цифр длиной не более 4000 байт. Правила ввода аналогичны формату byte |
| REAL | [s]99999999.9999<br>Незначащие нули можно не вводить |
| DOUBLE | [s]99999999999999.999999999<br>Незначащие нули можно не вводить |
| DECIMAL (NUMERIC) | [s]9999999999999999.9999999999<br>Незначащие нули можно не вводить |
| DATE | Формат дд.мм.гггг[:чч:[ми:[сс:[тт]]]] |
| BOOLEAN | Символ `T` или `F`:<br>`T` – true, `F` – false |
| NCHAR | Символьная строка длиной не более 4000 цифровых символов, задающая UNICODE-значение в текущей кодовой странице `inl` (не более 2000 UNICODE-символов) |

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

| Тип параметра | Формат ввода |
|---|---|
| NCHAR VARYING | Символьная строка длиной не более 4000 цифровых символов, задающая UNICODE-значение в текущей кодовой странице `inl` (не более 2000 UNICODE-символов) |

## Примеры

1)

```
SQL>SELECT count(model) FROM AUTO WHERE COLOR=?
1>AND YEAR = :YEAR;
Параметр 1 (CHAR(10))>YELLOW
YEAR (INTEGER)>71
|            38|
INL : выдано строк: 1
SQL>exec
Параметр 1 (CHAR(10))>GREEN
YEAR (INTEGER)>70
|            20|
INL : выдано строк: 1
```

2)

```
SQL>SELECT count(model)FROM AUTO WHERE ?=COLOR
1>AND year=:YEAR;
Параметр 1 (CHAR(10))>YELLOW
YEAR (INTEGER)>71
|            38|
```

3)

```
SQL>SELECT * FROM AUTO WHERE COLOR=? AND YEAR = :YEAR;
Параметр 1 (CHAR(10))>ELLOW
YEAR (SMALLINT)>71
```

4)

```
SQL>SELECT * FROM AUTO WHERE HORSEPWR=:A OR DSPLCMNT=:A;
A (SMALLINT)>250
```

# Команды

## !

### Формат

```
! [символьная строка]
```

### Назначение

Вывод на экран терминала или в выходной файл символьной строки.

### Описание

При обработке SQL-скриптов часто возникает необходимость вести протокол выполнения SQL-скрипта, то есть выдавать на экран терминала (или в выходной файл) сообщения о том, какая работа выполняется в текущий момент или уже была выполнена. Это можно сделать с помощью комментариев, которые явно записываются в SQL-скрипт и выводятся по команде `!`. Дополнительно команда `!` может использоваться для форматирования результатов поисковых SQL-запросов, выводимых в файл в виде простых справок (отчетов).

Команда обрабатывается следующим образом:

- текст комментария выводится на экран терминала. Если используется средство операционной системы для перенаправления стандартного вывода – то в указанный выходной файл;
- если команда задается в интерактивном режиме, то текст комментария просто дублируется на экране. Таким способом можно протоколировать интерактивную работу пользователя;
- текст выводится в том виде, в каком он задан, начиная от символа `!` и до конца строки. Только первый встретившийся в строке символ `!` воспринимается как команда, все остальные – как обычные символы;
- команда `!` воздействует только на одну строку SQL-скрипта. Перенос комментария не допускается (необходимо в каждой продолжающейся строке комментария повторять команду `!`);
- использование комментария внутри текстов SQL-запросов запрещается (в общем случае будет фиксироваться синтаксическая ошибка).

### Примеры

1) Пусть содержимое SQL-скрипта `sample.sql` будет следующее:

```
! *** исполняется запрос – que1.sql
_que1.sql
! *** исполняется запрос – que2.sql
_que2.sql
! *** исполняется запрос – que3.sql
_que3.sql
```

Протокол выполнения скрипта `sample.sql` на экране терминала будет выглядеть следующим образом:

```
SQL> _sample.sql
  *** исполняется запрос – que1.sql
  *** исполняется запрос – que2.sql
  *** исполняется запрос – que3.sql
SQL>
```

Если SQL-скрипт `sample.sql` запущен на выполнение как

```
inl -u SYSTEM/MANAGER8 -f sample.sql > prtcol.txt
```

то протокол выполнения будет записываться в файл `prtcol.txt`, создаваемый в том же каталоге, в котором выполняется утилита `inl`.

2) Пример использования команды `!` для форматирования выборки данных поискового запроса. Пусть SQL-скрипт имеет следующий вид:

```
time
!                Справка
! об автомобилях выпуска до 1970 года
select make as "Изготовитель", model as "Модель"
from auto where year <=70;
```

Команда `TIME` добавлена, чтобы исключить из справки временные параметры выполнения SQL-запроса. Тогда справка будет выдана в следующем виде:

```
                  Справка
 об автомобилях выпуска до 1970 года
 Изготовитель          Модель
 –                     –
|ALPINE               |A-310                |
|CHRYSLER             |DODGE CORONET CUSTOM|
|AMERICAN MOTORS      |GREMLIN X            |
|GENERAL MOTORS       |OLDSMOBILE 98        |
…
```

**Примечание**

Комментарии, задаваемые по команде `!`, являются исполняемыми, то есть всегда выводятся на экран или в выходной файл. Для комментирования (документирования) текстов SQL-скриптов можно использовать неисполняемые комментарии. Они задаются следующим образом:

```
<однострочный SQL-оператор> [!]<комментарий>;
```

Например:

```
create role ROLE1  for XXXX; Роль "Начальник отдела";
```

или

```
create role ROLE1  for XXXX; ! Роль "Начальник отдела";
```

причем и SQL-оператор и комментарий к нему должны размещаться на одной строке. Комментарий должен заканчиваться знаком ';'. Знак '!' (признак исполняемого комментария) можно не задавать – он все равно будет проигнорирован.

## _

### Формат

_<спецификация файла>

<Спецификация файла> задает каталог, имя и расширение текстового файла, содержащего выполняемый SQL-скрипт.

Если в <спецификации файла> не указан каталог, то файл ищется в текущем каталоге.

### Назначение

Выполнение SQL-скрипта.

### Описание

Команда _ (символ подчеркивания) заставляет inl выбирать команды для выполнения не с терминала, а из указанного файла. Файл для команды _ может содержать SQL-запросы и/или команды inl.

По достижении конца файла inl автоматически переключится на прием команд с терминала.

Входной файл – это обычный текстовый файл, который можно подготовить любым текстовым редактором, имеющимся в операционной системе. По умолчанию в inl в качестве входного файла установлен терминал пользователя.

Правила подготовки входного файла максимально просты: команды набираются так, как если бы они набирались в диалоговом режиме.

Этот файл может готовиться, естественно, не только текстовым редактором, но и любой программой, в том числе и средствами inl (см. команду EDIT).

### Примеры

1) Заданный SQL-скрипт (systab.sql) ищется в подкаталоге dict корневого каталога (ОС Linux, ЗОСРВ Нейтрино):

```
SQL>_../dict/systab.sql
```

2) Данный пример демонстрирует запуск командного файла (append.sql). Этот файл содержит три запроса на добавление (в таблицы AUTO, FINANCE и PERSON) и контрольный запрос на поиск.

Содержимое файла append.sql:

```
Page
Time
Count
! ****** Insert Into Table AUTO  ******
INSERT INTO AUTO VALUES('VOLVO', '4 DR', 'COUPE', 8, 120, 118,
 2900, 'BLACK', 70, '0262593464330109', 720303, 98224, 10001);
! ****** Ok. ******
! ****** Insert Into Table FINANCE ******
INSERT INTO FINANCE VALUES('AMERICAN EXPRESS', 300, 800, 'ARCO',
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
 1600, 5, 'HARTFORD INSURANCE', 21000, 'RUTGERS', '',
 'SEASHORE PROP.', 'SECURITY PACIFIC NATIONAL L.A.',10001);
! ****** Ok. ******
! ****** Insert Into Table PERSON ******
INSERT INTO Person (FIRSTNAM, PERSONID) VALUES('JENNIFER',10001);
! ****** Ok. ******
! ****** Test Query ******
SELECT COUNT(*) FROM AUTO, FINANCE, PERSON
WHERE Person.PersonId =10001 AND
Auto.PersonId=Person.PersonId AND
Finance.PersonId=Person.PersonId;
```

Для выполнения этого запроса нужно подать команду запуска:

```
SQL>_append.sql
******  Insert Into Table AUTO  ******
******  Ok. ******
******  Insert Into Table FINANCE ******
******  Ok. ******
******  Insert Into Table PERSON  ******
******  Ok. ******
******  Test Query  ******
1
SQL>
```

**Примечание**
Если сразу после запуска файла `append.sql` подать команду EXEC, то еще раз выполнится контрольный запрос на поиск, но без комментария (`****** Test Query ******`).

# ALTER PROC

### Формат

`ALTER PROC <спецификация файла>`

### Назначение

Модификация хранимой процедуры из файла.

**Примечание**
Команда для модификации триггера не предусмотрена. Для выполнения этой операции триггер необходимо удалить, а затем создать заново.

### Описание

Команда `ALTER PROC` заменяет текст существующей в БД хранимой процедуры.

<Спецификация файла> задает местоположение, имя и расширение текстового файла с новым исходным текстом хранимой процедуры.

Для выполнения команды необходимы следующие условия:

1) в БД должны присутствовать системные таблицы $$$PRCD, $$$PROC;
2) модифицируемая процедура должна существовать в БД;
3) текст модифицируемой процедуры в исходном файле должен начинаться с ключевого слова PROCEDURE. Другие ключевые слова (типа CREATE) недопустимы.

# BLOB

См. раздел «Работа с BLOB-данными».

# BROWSE

## Формат

BROWSE

## Назначение

Смена режима отображения информации.

## Описание

Утилита inl поддерживает два способа отображения информации, получаемой в результате выполнения поисковых SQL-операторов: строчный и табличный.

Специфика строчного режима:

- каждая запись выборки данных сразу выдается на экран терминала или записывается в выходной файл (если при запуске inl был сделано перенаправление стандартного вывода);
- если запись выборки данных не умещается на экране терминала, то она продолжается на следующих строках экрана;
- записи выборки данных выдаются на экран с той скоростью, с какой происходит выборка информации из базы данных;
- приостанов выдачи информации (например, для просмотра или анализа) выполняется стандартными для операционной системы средствами управления экраном терминала;
- если выданная запись выборки данных ушла «за экран», то доступ к ней уже невозможен;
- нет возможности просматривать данные типа BLOB и EXTFILE.

## Примеры выдачи выборки данных в строчном режиме

1) запись выборки данных полностью помещается в строке экрана (рисунок 2):

```
SQL>select make, model, color from auto
where personid in (10,20,30);
```

```
 MAKE                 MODEL                  COLOR
 ----                 -----                  -----
!AMERICAN MOTORS      !MATADOR STATION       !WHITE      !
!FORD                 !MERCURY MONTEREY      !BLACK      !
!FORD                 !LTD COUNTRY SQUIRE    !GREEN      !
 INL : выдано строк      : 3
```

Рисунок 2. Короткая запись выборки данных

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

2) запись выборки данных располагается в нескольких строках экрана (рисунок 3)

```
SQL>select * from auto where personid in (10,20,30);
```

```
 MAKE                 MODEL                BODYTYPE        CYLNDERS    HORSEPWR
   DSPLCMNT    WEIGHT       COLOR      YEAR         SERIALNO          CHKDATE     C
HKMILE     PERSONID
 ----                 -----                --------        --------    --------
   --------    ------      -----      ----         --------          -------     -
------     --------
!AMERICAN MOTORS      !MATADOR STATION     !STATION WAGON  !          8!          1
50!        304!        3725!WHITE      !          70!P298565792069809!      720122!
     85778!          10!
!FORD                 !MERCURY MONTEREY    !SEDAN HARDTOP  !          8!          2
08!        430!        4500!BLACK      !          71!K670210003057005!      720202!
     47013!          20!
!FORD                 !LTD COUNTRY SQUIRE  !STATION WAGON  !          8!          1
63!        352!        4505!GREEN      !          71!M787560194860222!      720205!
     28616!          30!
 INL : выдано строк      : 3
```

Рисунок 3. Длинная запись выборки данных

Очевидны отрицательные стороны строчного режима:

- нет возможности вернуться к уже выданной выборке данных;
- отсутствует скроллинг по горизонтали в выборке данных, который представляет собой широкую таблицу, сложно проследить значения столбцов, не входящих (по ширине) в одну строку терминала.

Специфика табличного режима:

- `inl` сначала запоминает в собственном буфере столько записей выборки данных, сколько позволяет доступная оперативная память;
- после заполнения буфера выдает записи выборки данных в удобной экранной форме;
- поддерживается скроллинг по горизонтали/вертикали (не уместившиеся в буфере записи выборки данных подкачиваются по мере необходимости);
- поддерживается возможность манипулирования данными (добавление, удаление, модификация) для обновляемых SELECT-запросов;
- возможность просмотра данных типа `BLOB` и `EXTFILE`;
- форма представления информации удобна для визуализации выборки данных, но работа, в общем случае, выполняется несколько медленнее по сравнению со строчным режимом, особенно при первоначальном приеме выборки данных.

### Примечание

Если в строчном режиме вывод результата перенаправлен с экрана в выходной файл, то длинные записи выборки данных помещаются в файл полностью, без разделения на отдельные строки. В этом случае при просмотре такого файла стандартными средствами операционной системы возможен скроллинг по горизонтали/вертикали и строчный режим по своим визуальным возможностям приближается к табличному.

Табличный режим действует только на те SQL-запросы, которые потенциально возвращают множественную выборку данных, а именно:

- SELECT-запросы;

- хранимые процедуры, возвращающие в качестве выборки данных курсор;
- оператор `TEST TABLE ...`, возвращающий набор диагностических сообщений.

Для всех остальных SQL-операторов всегда используется строчный режим вывода информации (даже если явно установлен табличный режим).

### Пример выдачи выборки данных в табличном режиме

Выдача выборки данных в табличном режиме приведена на рисунке 4.




Рисунок 4. Просмотр результатов в табличном режиме

### Модификация данных в табличном режиме

В табличном режиме можно выполнять обновление данных (добавление, удаление, корректировку), полученных в результате выполнения обновляемого SELECT-запроса. SELECT-запрос считается обновляемым, если выполняются следующие условия:

- данные выбираются только из одного экземпляра таблицы или представления;
- таблица не является системной таблицей БД;
- представление порождено не из системной таблицы;
- результатом выполнения SELECT-запроса являются значения столбцов таблицы (представления), а не вычисляемые выражения агрегатных функций;
- запрос не содержит литералов.

Команда `BROWSE` работает как циклический двоичный переключатель, то есть каждое выполнение `BROWSE` отменяет текущий режим отображения и устанавливает противоположный. Установленный режим сохраняется до конца работы `inl`.

При запуске `inl` по умолчанию устанавливается строчный режим отображения.

Для просмотра текущего режима отображения используется команда SHOW:

- положение `вкл.` – режим браузера (табличный режим);

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

- положение выкл. – терминальный (строчный) режим.

# CODEPAGE

## Формат

```
CODEPAGE <кодовая страница>
```

## Назначение

Установка кодовой таблицы.

## Описание

Результатом выполнения команды CODEPAGE является информирование сервера СУБД ЛИНТЕР о том, в какой кодировке работает программа inl. Это означает, что сервер ожидает от inl символьные данные в заданной кодировке и, соответственно, в этой же кодировке должен возвращать ей результаты SQL-запросов. При этом данные в самой БД могут храниться в любой другой кодировке. Ответственность за правильное кодирование входных данных возлагается на inl. СУБД ЛИНТЕР не проверяет соответствие декларированной кодовой страницы и фактически использованной, поэтому, в случае их несовпадения, в БД могут быть записаны неправильные символьные данные.

Команда CODEPAGE действует до конца текущего сеанса inl или до выполнения новой команды CODEPAGE.

По умолчанию при запуске inl используется кодовая страница 866.

Информация о текущей кодовой странице выдается по команде LIST.

## Примеры

1) Пусть в таблице Tst_Code символьные данные хранятся в кодировке KOI8-R. Для их корректного отображения необходимо установить эту же кодировку.

```
SQL>codepage KOI8-R
SQL>select * from "Tst_Code";
…
```

2) Получить список поддерживаемых СУБД кодовых страниц и установить нужную.

```
SQL>select name from $$$charset;

   |DEFAULT |
   |CP866   |
   |KOI8-R  |
   |CP1251  |
   |CP437   |
   |CP1252  |
   |CP8859-1|
   |CP8859-2|
   …
SQL>codepage KOI8-R
```

# COUNT

## Формат

COUNT

## Назначение

Разрешение/запрет выдачи итоговой статистики.

## Описание

Результаты обработки SQL-запросов, относящихся к обработке собственно данных, inl по умолчанию сопровождает дополнительной статистической информацией, позволяющей оценивать как временные характеристики выполнения запроса, так и его семантическую корректность (количество реально выбранных/обработанных записей), например:

```
INL : удалено строк: 23
```

или

```
INL : выдано строк: 12
```

Если вывод статистической информации нежелателен, то его можно запретить, установив переключатель итоговой статистики в положение выкл. (выключен).

Если этот переключатель установлен в положение вкл. (включен), то выборки данных будут сопровождаться итоговой статистикой.

Управление переключателем итоговой статистики выполняет команда COUNT, которая меняет текущее состояние переключателя на противоположное, то есть вкл./выкл. на выкл./вкл. соответственно.

Команда COUNT воздействует только на SQL-операторы:

- SELECT;
- INSERT;
- DELETE;
- UPDATE;
- EXECUTE PROCEDURE (если возвращается тип данных «курсор»).

Сразу после запуска inl переключатель итоговой статистики по умолчанию установлен в положение вкл.

Для просмотра текущего состояния переключателя статистики используется команда LIST.

## Примеры

1) В процессе отладки SQL-запроса для подсчета найденных записей можно вместо встроенной в SQL функции count использовать команду COUNT.

```
SQL>select count(personid) from auto where make='FORD';
|         118|
INL : выдано строк: 1
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
SQL>select personid from auto where make='FORD';
|           1|
|          20|
|          22|
|          30|
…
|         981|
|         983|
|         997|
INL : выдано строк: 118
```

2) Получить время выполнения SQL-запроса.

```
SQL>list
…
count    : вкл.
…
SQL>update auto set year=year+1900;
INL: начальное время : 17:54:54 конечное  время : 17.54.56
INL: изменено строк: 1000
```

# CREATE PROC

### Формат

```
CREATE PROC <спецификация файла>
```

### Назначение

Создание хранимой процедуры из файла.

### Описание

<Спецификация файла> задает местоположение, имя и расширение текстового файла с исходным текстом хранимой процедуры.

Для выполнения команды необходимы следующие условия:

1) в БД должны присутствовать системные таблицы `$$$PRCD`, `$$$PROC`;
2) имя создаваемой процедуры должно быть уникальным в текущей схеме;
3) текст процедуры в исходном файле должен начинаться с ключевого слова `PROCEDURE`. Другие ключевые слова (типа `CREATE`, `ALTER`) недопустимы.

### Пример

Файл `crt_proc.sql` (исходный текст хранимой процедуры):

```
procedure test_proc ()
declare
   exception noresults for custom 100;
code
   execute direct "create or replace table tab (id1 int, id2 int,
 s char(10));";
```

```
   if errcode() <> 0 then
     signal noresults;
   endif
exceptions
   when others then
     resignal;
end

! Создание процедуры из файла
SQL>create proc crt_proc.sql

! Выполнение созданной процедуры
SQL>execute test_proc();
```

# CREATE TRIG

### Формат

```
CREATE TRIG <спецификация файла>
```

### Назначение

Создание триггера из файла.

### Описание

<Спецификация файла> задает местоположение, имя и расширение текстового файла с исходным текстом триггера.

Для выполнения команды необходимы следующие условия:

1) в БД должна присутствовать системная таблица `$$$TRIG`;
2) имя создаваемого триггера должно быть уникальным в текущей схеме;
3) текст триггера в исходном файле должен начинаться с ключевого слова `TRIGGER`. Другие ключевые слова (типа `CREATE`, `ALTER`) недопустимы.

### Пример

Пусть созданы таблицы:

```
create or replace table test(i int);
create or replace table test_result(ch char(20));
```

Файл `crt_trig.sql` (исходный текст триггера):

```
trigger test_tr before insert on test for each row old as "OLD"
 new as "NEW"
 execute
code
  execute direct "insert into test_result(ch) values('inserted #"
 + itoa(NEW.i) + "');"; //
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
end;

! Создание триггера  из файла
SQL>create trig crt_trig.sql

! Выполнение триггера
SQL> insert into test values(1);
SQL> insert into test values(2);
SQL> select * from test_result;
|inserted #1          |
|inserted #2          |
```

# DBINFO

### Формат

`DBINFO`

### Назначение

Получение справочной информации о параметрах запуска СУБД.

### Описание

Команда DBINFO выдает информацию о параметрах запуска ядра СУБД ЛИНТЕР. Она может использоваться для следующих целей:

1) эмпирический подбор параметров запуска СУБД ЛИНТЕР для оптимизации выполнения как отдельных клиентских приложений (автономная оптимизация), так и совокупности нескольких, одновременно работающих с БД, приложений (комплексная оптимизация). Этот процесс выполняется, как правило, следующим образом:
   - предварительно теоретически, на основе анализа потока SQL-запросов клиентского приложения, оцениваются размеры очередей таблиц, столбцов, памяти ядра и других параметров, влияющих на эффективность работы ядра СУБД;
   - на макете (или на готовом клиентском приложении) практически оценивается эффективность выбранных параметров. Путем варьирования значений параметров и отслеживания их влияния на эффективность обработки подбираются оптимальные значения.
2) задавать правильные режимы работы с БД, например:
   - если при запуске СУБД был отключен режим транзакций (эта информация, в числе прочей, выдается по команде `DBINFO`), то отказ от ошибочно измененной в БД информации будет невозможен;
   - если выполняется отладка клиентского приложения, то иногда полезно вести протокол обращения к БД (файл `LINTER.LOG`), в котором фиксируются все операции с данными, выполненные ядром СУБД. При эксплуатации клиентских приложений ведение протокола обращений приводит только к дополнительным затратам времени на обработку SQL-запросов и поэтому не имеет смысла. Информация о протоколе обращения также выдается по команде `DBINFO`.

### Пример

```
SQL>dbinfo
Информация о базе данных 'DEMO DATABASE'
СУБД ЛИНТЕР версия  :6.0.17
Размер памяти ядра  :522174
Размер очереди каналов  :100
Размер очереди таблиц :190
Размер очереди колонок  :1270
Размер очереди файлов :390
Размер очереди пользователей  :100
Размер памяти сортировки  :130543
Размеры кэшей транслятора SQL
(0 - по умолчанию) польз/таблицы/столбцы/проц/кодировки :0/0/0/0/0
Количество процессов сортировки :1
Предельная длина записи в таблице БД  :4096
Интервал сброса изменений   :0
Интервал проверки соединения  :120
Журнал транзакций :включен
Протокол обращений (LINTER.LOG) :выключен
Синхронный вывод  :выключен
Разный порядок байт клиента и сервера :нет
Режим совместимости по стандартом SQL :нет
Обязательные префиксы для геоданных :нет
Режим 'только чтение' :нет
Квантование по времени  :нет
Увеличенный буфер обмена  :да
Кодировка базы данных :CP1251
Операционная система  :Windows NT
```

В строке «СУБД ЛИНТЕР версия» третье число обозначает номер сборки.

# DISFILL

### Формат

```
DISFILL
```

### Назначение

Запрещает/разрешает заполнение пробелами полей записи выборки данных до максимальной длины столбца.

### Описание

Команда `DISFILL` запрещает/разрешает дополнять пробелами столбец записи выборки данных до заданной при создании таблицы максимальной длины.

Команда работает как переключатель – изменяет текущий режим на противоположный.

По умолчанию при запуске `inl` установлен режим вывода максимальной длины столбца.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

### Пример

```
create table tst(vc varchar(10), c char(5));
insert into tst (vc, c) values('1','1');
insert into tst (vc, c) values('11','11');
insert into tst (vc, c) values('111','111');
-- выдача столбцов выборки данных по максимальной длине (режим по
 умолчанию)
select vc, c from tst;
 VC          C
 --          -
|1          |1    |
|11         |11   |
|111        |111  |
 INL : выдано строк: 3

-- выдача столбцов выборки данных по фактической длине
disfill
select vc, c from tst;
 VC  C
 --  -
|1|1    |
|11|11   |
|111|111  |
 INL : выдано строк: 3
```

# DISHEAD

### Формат

```
DISHEAD
```

### Назначение

Запрещает/разрешает выводить заголовок столбца записи выборки данных.

### Описание

Команда `DISHEAD` запрещает/разрешает выводить заголовок столбца записи выборки данных, установленный по умолчанию или с помощью команды HEADER.

Команда работает как переключатель – изменяет текущий режим вывода заголовка на противоположный.

По умолчанию при запуске `inl` установлен режим вывода заголовка.

### Пример

```
select make, model from auto fetch first 2;
MAKE                 MODEL
 ----                 -----
```

```
|FORD                |MERCURY COMET GT V8 |
|ALPINE              |A-310               |
 INL : выдано строк: 2

header: Изготовитель        Модель
select make, model from auto fetch first 2;
Изготовитель        Модель
|FORD                |MERCURY COMET GT V8 |
|ALPINE              |A-310               |
 INL : выдано строк: 2

dishead
select make, model from auto fetch first 2;
|FORD                |MERCURY COMET GT V8 |
|ALPINE              |A-310               |
 INL : выдано строк: 2
```

# DISPLAY

### Формат

```
DISPLAY
EVENT [<имя схемы>.]<имя события>
| VARIABLE [<имя схемы>.]<имя переменной>
```

Поддерживаются следующие конструкции именования событий и переменных:

- "schema.name";
- "schema"."name";
- Name;
- "name";
- schema.name;
- schema."name";
- "schema".name.

### Назначение

Предоставляет справочную информацию о событиях (глобальных/не глобальных) и глобальных переменных хранимых процедур.

### Описание

Опция `EVENT` предоставляет справочную информацию о событиях. Информация берется из системной псевдотаблицы `$$$EVENTS_INFO`.

<Имя события> может ссылаться как на глобальное (хранимое в БД) событие, так и на обычное (локальное существующее только в текущем сеансе СУБД).

Если <имя схемы> не задано, по умолчанию используется текущая схема.

Все имена допускают символ обобщения %.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

Если имена являются регистрозависимыми, то они должны обрамляться двойными кавычками, например,

```
"SysAdmin"."%"
        "Ито%"
```

Для получения соответствующей справочной информации подать запросы:

- о всех событиях:

```
display event %.%
```

- о всех глобальных переменных хранимых процедур:

```
display variable%.%
```

- о всех событиях только текущей схемы:

```
display event %
```

- о всех глобальных переменных хранимых процедур только текущей схемы:

```
display variable %
```

### Примеры

1)

```
create or replace global event MOD_AUTO as delete, update on auto;

display event m%
SQL>
             Event SYSTEM.MOD_AUTO

 * ID                          : 3
 * Global                      : YES
 * On table                    : AUTO
 * On operation                : UPDATE DELETE
 * Autoreset                   : NO
 * Disabled                    : NO
 * Active                      : NO
 * Procedure                   : NO
 * Wait execution procedure    : NO
 * Private                     : NO
 * Current time event          : NO
 * Execute as current user     : NO
 * Source                      : create or replace global event
 MOD_AUTO as
 delete, update on auto
 * Query charset               : CP866
```

2)

```
display event system.m%
display event sys%.m%
```

```
display event s%.m%
display event s%.%
display event %.mod_auto
display event s%.%
display event %."Удаление в AUTO"
display event %."Удаление%"
display event "Адмн%"."Удаление%"
```

3)

```
create  variable department int=125;
create  variable company char='RELEX';

SQL> display variable %

             Variable SYSTEM.DEPARTMENT

 * ID                          : 1
 * NULL allowed                : NO
 * Type                        : INTEGER
 * Maximum length              : 4
 * Default value length        : 4
 * Default value               : 125

             Variable SYSTEM.COMPANY

 * ID                          : 2
 * NULL allowed                : NO
 * Type                        : CHAR
 * Maximum length              : 20
 * Default value length        : 5
 * Default value               : 'RELEX'
```

# DISREP

### Формат

`DISREP`

### Назначение

Запрещает/разрешает выводить результаты выполнения SQL-запросов и хранимых процедур.

### Описание

Команда `DISREP` аналогична команде COUNT, но ее действие распространяется и на вывод результата хранимых процедур (если возвращается некурсорный тип данных).

Команда работает как переключатель – изменяет текущий режим на противоположный.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

По умолчанию при запуске `inl` установлен режим вывода результатов.

### Пример

```
SQL>create procedure proc(in p int) result int
1>code
2> return p+3;//
3>end;
SQL>execute proc(2);
return value = 5
SQL>disrep
SQL>execute proc(2);
SQL>
```

# DISSEP

### Формат

```
DISSEP
```

### Назначение

Запрещает/разрешает выводить разделители в записи выборки данных.

### Описание

Команда `DISSEP` запрещает/разрешает выводить разделители, установленные по умолчанию или с помощью команды UNLOAD.

Команда работает как переключатель – изменяет текущий режим вывода межстолбцовых разделителей на противоположный.

По умолчанию при запуске `inl` установлен режим вывода всех разделителей.

**Примечание**
Проверить текущий режим вывода разделителей можно с помощью команды LIST. Если установлен запрет на вывод разделителей, то выдается сообщение `Separators disabled`. Если вывод разделителей разрешен, сообщение об этом не выдается.

### Примеры

```
unload:!
list
…
(по умолчанию вывод всех разделителей разрешен, поэтому сообщение
 об этом не выдается)
…
select personid, make, model from auto fetch first 2;

PERSONID    MAKE                 MODEL
 --------    ----                  -----
```

```
!          1!FORD                !MERCURY COMET GT V8 !
!          2!ALPINE              !A-310               !

 INL : выдано строк: 2

-- отмена выдачи всех разделителей
dissep
list
…
separators disabled
…
select personid, make, model  from auto fetch first 2;

PERSONID    MAKE                 MODEL
 --------    ----                 -----
          1FORD                MERCURY COMET GT V8
          2ALPINE              A-310

 INL : выдано строк: 2

-- возобновление выдачи всех разделителей
dissep
select personid, make, model  from auto fetch first 2;

PERSONID    MAKE                 MODEL
 --------    ----                 -----
!          1!FORD                !MERCURY COMET GT V8 !
!          2!ALPINE              !A-310               !

INL : выдано строк: 2
```

# DISTAIL

### Формат

`DISTAIL`

### Назначение

Запрещает/разрешает выводить в записи выборки данных концевые разделители.

### Описание

Команда `DISTAIL` запрещает/разрешает выводить концевые разделители, установленные по умолчанию или с помощью команды UNLOAD.

Команда работает как переключатель – изменяет текущий режим вывода разделителей на противоположный.

По умолчанию при запуске `inl` установлен режим вывода концевых разделителей.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

### Примечания

1. Проверить текущий режим вывода разделителей можно с помощью команды LIST. Если установлен запрет на вывод разделителей, то выдается сообщение `Limiters disabled`. Если вывод концевых разделителей разрешен, сообщение об этом не выдается.
2. Команда `DISTAIL` устанавливает концевые разделители только в том случае, если они не были установлены предыдущей командой DISSEP.

### Пример

```
unload:!
list
select personid, make, model  from auto fetch first 2;

PERSONID    MAKE                 MODEL
 --------    ----                 -----
!          1!FORD                !MERCURY COMET GT V8 !
!          2!ALPINE              !A-310               !
 INL : выдано строк: 2

-- отмена выдачи граничных разделителей
distail
list
select personid, make, model  from auto fetch first 2;
PERSONID    MAKE                 MODEL
 --------    ----                 -----
          1!FORD                !MERCURY COMET GT V8
          2!ALPINE              !A-310
 INL : выдано строк: 2
```

# ECHO

### Формат

```
ECHO {ON | OFF | ERROR}
```

### Назначение

Трассировка SQL-запросов.

### Описание

Команда позволяет осуществлять вывод в стандартный поток вывода всех или только ошибочных запросов выполняемого SQL-скрипта.

При указании опции ON выводятся:

- тексты всех SQL-запросов до синтаксического разбора;
- команды username;
- команды create/alter proc;

- команды `blob insert`, `blob append`, `blob clear`, `blob get`. Как и для других подобных команд, при этом выдается не исходный текст команды, а результат ее разбора. Лишние пробелы, если они были, не выдаются. Имена схем, таблиц и столбцов выдаются без кавычек (даже если кавычки были в исходном тексте). Тип столбца выдается только в случае, если он не 0. Явно заданное шестнадцатеричное значение выдается в верхнем регистре.

При указании опции OFF тексты SQL-запросов не выводятся (режим по умолчанию).

При указании опции ERROR выводятся тексты только тех SQL-запросов, при выполнении которых произошла ошибка. Если для SQL-запроса действует команда игнорирования ошибочного кода завершения (IGNORE), то такие запросы не трассируются.

Аналогично трассируются вызовы хранимых процедур.

## Примеры

1) Файл `tst.sql`:

```
echo off;
! >>> трассировка не выполняется
drop table $$$LEVEL;
echo on;
! >>>выполняется трассировка всех запросов
drop table $$$LEVEL;
echo error;
! >>>выполняется трассировка только ошибочных запросов
drop table $$$LEVEL;
```

Запуск SQL-скрипта:

```
inl -u SYSTEM/MANAGER8 -f tst.sql > trace.sql
```

Результат выполнения скрипта (файл `trace.sql`):

```
>>> трассировка не выполняется
 INL : начальное время : 14:54:07 конечное время :  14:54:08
 INL : состояние выполнения : 83
нельзя явно изменять системную таблицу
>>> выполняется трассировка всех запросов
drop table $$$LEVEL;
 INL : начальное время : 14:54:08 конечное время :  14:54:08
 INL : состояние выполнения : 83
нельзя явно изменять системную таблицу
echo error;
>>> выполняется трассировка только ошибочных запросов
 INL : начальное время : 14:54:08 конечное время :  14:54:08
 INL : состояние выполнения : 83
drop table $$$LEVEL;
нельзя явно изменять системную таблицу
```

2) В среде ОС типа Linux, ЗОСРВ Нейтрино:

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
echo 'select * from auto;' | inl -u SYSTEM/MANAGER8
```

# EDIT

## Формат

```
EDIT [<спецификация файла>]
```

<спецификация файла>::= местонахождение (полный путь, имя и расширение) редактируемого SQL-скрипта.

Если <спецификация файла> не задана, по умолчанию используется файл `inl.buf` из каталога запуска `inl`.

## Назначение

Интерактивное создание/редактирование SQL-скрипта.

## Описание

Команда EDIT используется для интерактивного (то есть не выходя в операционную систему) создания или редактирования SQL-скрипта во внешнем буфере `inl`.

Все SQL-запросы и команды, которые влияют на отображение результатов выполнения SQL-запросов, `inl` запоминает и хранит в своем рабочем буфере размером 64 Кбайт. Таким образом, максимальная длина SQL-запроса, обрабатываемого программой `inl`, равна 64 Кбайт.

Рабочий буфер всегда (кроме самого первого запуска `inl` после установки СУБД ЛИНТЕР) содержит два объекта: последнюю выполненную команду (`outfil`, `header`, `unload` или `!`) и последний (введенный пользователем) или выполненный программой `inl` SQL-запрос.

После выполнения SQL-запроса его текст из буфера не удаляется, поэтому можно:

- повторить его выполнение необходимое число раз (например, с различными значениями параметров параметризованного SQL-запроса);
- отредактировать и выполнить;
- сохранить в отдельном файле для будущего использования.

Повторное выполнение SQL-запроса выполняется по команде EXEC.

Процесс подготовки и выполнения SQL-запроса в интерактивном режиме проходит в следующей последовательности:

1) выполнить команду edit.

```
SQL>edit
```

после чего `inl`:

- удаляет из текущего каталога файл `inl.buf` (если он там есть);
- открывает на запись новый файл `inl.buf` (все в том же текущем каталоге);
- переписывает из внутреннего буфера в файл `inl.buf` последний выполненный (введенный) SQL-запрос и, если есть, последнюю команду форматирования;

- закрывает файл `inl.buf` и вызывает на выполнение стандартный системный редактор текстовых файлов (рисунок 5) (например, в среде ОС Windows это Notepad), который открывает файл `inl.buf`.

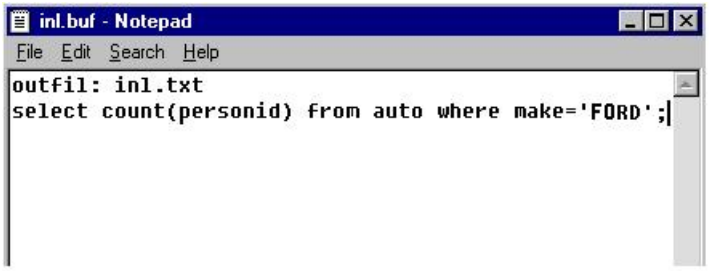



Рисунок 5. Окно текстового редактора

2) ввести (если файл `inl.buf` пуст) или откорректировать (при необходимости) с помощью стандартных средств редактора текст SQL-скрипта;
3) сохранить при необходимости созданный (откорректированный) SQL-скрипт в отдельном текстовом файле;
4) выйти из редактора. Утилита `inl` переписывает содержимое `inl.buf` в свой внутренний буфер. После этого созданный (откорректированный) SQL-скрипт можно выполнить сразу же или позднее по команде EXEC;
5) при активизации редактора вместо открываемого по умолчанию файла `inl.buf` можно с помощью стандартных средств самого редактора открыть любой ранее созданный SQL-скрипт, текст которого будет сохранен при закрытии редактора по умолчанию в `inl.buf`.

### Пример

```
!Запуск редактора для ввода текста SQL-скрипта
SQL>edit
!В окне редактора вводим текст SQL-скрипта и сохраняем его
!в файле по умолчанию inl.buf, закрываем редактор

!Выполняем созданный SQL-скрипт
SQL>exec
!Получаем код завершения «Несуществующая таблица»
!Просматриваем текст последнего выполненного SQL-запроса
SQL>list
!Среди выданных командой list параметров находим текст
 выполненного SQL-запроса
Запрос:  select  count(*) from avto;
!Вызовем редактор для исправления ошибки
SQL>edit
!В окне редактора вносим исправление в текст SQL-скрипта и
 сохраняем  его в
 файле по умолчанию inl.buf, закрываем редактор

!Выполняем исправленный SQL-скрипт
SQL>exec
!Полученный результат смотрим в указанном в SQL-скрипте файле tst
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
|         1000|
```

# EHEAD

### Формат

```
EHEAD
```

### Назначение

Управление выводом заголовков столбцов пустой выборки данных.

### Описание

По умолчанию если выборка данных поискового запроса пустая, то информация о её столбцах не выводится.

Данная команда заставляет формировать и отображать шапку даже для пустой выборки данных.

Команда работает как переключатель – изменяет текущий режим на противоположный.

### Пример

```
SQL>select make, model from auto where personid=0;

INL : выдано строк: 0
SQL>ehead
SQL>select make, model from auto where personid=0;
 MAKE                 MODEL
 ----                 -----
 INL : выдано строк: 0

SQL>ehead
SQL>select make, model from auto where personid=0;
 INL : выдано строк: 0
```

# EXEC

### Формат

```
EXEC
```

### Назначение

Выполнение SQL-запроса.

### Описание

По команде `EXEC inl` выполняет SQL-запрос из своего внутреннего буфера (если он не пуст). Внутренний буфер `inl` может:

- быть пустым (в начале работы `inl`);
- содержать последний выполненный в текущем сеансе SQL-запрос;
- содержать текст отредактированного, но еще не выполненного SQL-запроса.

Для просмотра текущего состояния внутреннего буфера программы `inl` используется команда LIST.

Если в буфере хранился SQL-запрос с параметрами, то при выполнении `EXEC` необходимо каждый раз вводить значения параметров в интерактивном режиме или явно задавать их в SQL-скрипте.

### Пример

```
SQL>time
SQL>count
SQL>select count(*) from auto where year=70;
|       465|
; редактирование SQL-запроса
SQL>edit
; просмотр  внутреннего буфера inl
SQL>list
…
Запрос : select count(*) from auto where year=71;
…
SQL>exec
|       535|
SQL>
```

# EXIT|QUIT

### Формат

```
EXIT|QUIT
```

### Назначение

Завершение работы программы `inl`.

### Описание

При выполнении `EXIT` (`QUIT`):

- если необходимо, выполняется `COMMIT` незавершенной транзакции;
- разрывается соединение с БД;
- завершается выполнение `inl` и управление передается в операционную систему.

**Примечание**
В интерактивном сеансе завершить работу `inl` можно, нажав клавиши **<Ctrl>+<C>**, затем клавишу **<Enter>**.

### Пример

```
…
; в среде ОС Windows
SQL>exit
c:\>
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

# FORMAT

## Формат

FORMAT

## Назначение

Переключатель форматирования.

## Описание

Утилита inl позволяет отображать результаты выполнения SELECT-запросов в двух видах:

- естественном – информация выводится в соответствии с типом данных, хранящихся в БД;
- символьном – выводимая информация преобразуется к символьному виду независимо от ее хранения в БД.

Команда FORMAT управляет сменой режимов переключателя форматирования (меняет текущий режим форматирования на противоположный).

Если переключатель форматирования включен (положение вкл.), то inl преобразует записи выборки данных в символьный вид.

Если переключатель форматирования выключен (положение выкл.), то inl оставляет записи выборки данных в естественном двоичном виде. Это требуется обычно при выводе записей выборки данных в файл для последующей работы с ними другими средствами или для визуального просмотра.

Первоначально по умолчанию переключатель имеет положение выкл.

## Пример

```
SQL>list
      -- Текущие установки INL --
…

format   :выкл.
…
SQL>select make,year,weight from auto;
Make         YEAR  WEIGHT
----         ----  ------
|FORD        |  71|     2900|
|ALPINE      |  70|     1826|
…
SQL>format
SQL>list
      -- Текущие установки INL --
…

format   :вкл.
…
```

```
SQL>select make,year,weight from auto;
Make         YEAR  WEIGHT
----         ----  ------
|FORD        |G    |T    |
|ALPINE      |F    |"    |
…
```

# HEADER

## Формат

```
HEADER: [<символьная строка>] [-]  {[<символьная строка>] [-]…}
```

## Назначение

Определение заголовка выборки данных.

## Описание

По умолчанию заголовок выборки данных при выполнении SELECT-запроса представляет собой имена столбцов (или их синонимы) результирующей выборки, разграниченные заданным символом-разделителем. Если выбирается значение агрегатной функции или вычисляемого выражения, то в качестве имени используются пробелы (или синонимы выбираемых значений, если заданы).

С помощью команды `HEADER` можно заменить однострочный заголовок таблицы выборки данных, формируемый `inl` по умолчанию, на любой другой по желанию.

Команда выполняется по следующим правилам:

- команда относится к непосредственно следующему за ней SELECT-запросу. Если между `HEADER` и SELECT-запросом выполнялись другие SQL-операторы, то значение `HEADER` теряется (однако не SQL-команды `inl` значения команды `HEADER` не сбрасывают, за исключением команд `inl` BROWSE, DISHEAD, EXIT, что следует из их предназначения);
- если команда задана, то заголовок по умолчанию SELECT-запроса заменяется на заголовок из команды `HEADER`;
- команда EXEC не распространяется на команды `HEADER`, то есть при повторении выполнения SELECT-запроса команду `HEADER` надо явно повторить;
- в табличном режиме отображения данных команда не действует;
- максимальная длина формируемого по команде заголовка 800 символов;
- не уместившийся в одной строке текст заголовка можно перенести на следующую строку. Признаком переноса строки заголовка является символ дефиса «-» в последней позиции строки;
- можно создать заголовок, состоящий из нескольких строк. Признаком конца очередной строки заголовка является символ дефиса «-» в последней позиции строки. Максимальное количество строк заголовка – 10.

## Пример

```
SQL> header:          Утверждаю-
2>          Директор предприятия-
3>             "Виртуал" Прошан И.Я.-
4> -
5>            Штатное расписание отд. N 5-
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
6>------------------------------------------- -
7>   |Таб.N |   ФИО   |Должность| Оклад  |-
8>------------------------------------------- -
SQL>select "Таб_N", "ФИО", "Должность","Оклад"
1> from "Штаты" where "N_Отдел"=5;
```

# HELP

## Формат

```
HELP
```

## Назначение

Получение справочной информации о командах программы `inl`.

## Описание

Команда выдает на экран полный список обрабатываемых программой `inl` команд с краткими пояснениями (таблица 4).

Таблица 4. Результат выполнения команды HELP

| Имя команды | Назначение команды |
|---|---|
| `HELP` | Показать список команд |
| `TIME` | Включить/выключить выдачу времени |
| `FORMAT` | Переводить/не переводить ответ в символьный вид |
| `USERNAME` | Подключиться с другим именем пользователя |
| `PAGE` | Включить/выключить выдачу выборки данных по страницам |
| `COUNT` | Выводить/не выводить количество записей |
| `DISTAIL` | Разрешить/запретить вывод лидирующего и хвостового ограничителей |
| `DISFILL` | Разрешить/запретить дополнение VAR типов до максимальной ширины |
| `DISSEP` | Разрешить/запретить вывод сепаратора |
| `DISREP` | Разрешить/запретить вывод результатов выполнения процедур |
| `DISHEAD` | Разрешить/запретить вывод заголовка текста |
| `echo on` | Вывод выполняемого SQL-запроса в стандартный поток вывода |
| `echo off` | Запрет вывода выполняемого SQL-запроса в стандартный поток вывода |
| `echo error` | Вывод SQL-запроса в стандартный поток вывода в случае ошибки |
| `BROWSE` | Включить/выключить выдачу результата запроса в табличном режиме |
| `_<filename>` | Запуск запроса из файла |
| `OUTFIL[E]:` | Определение файла вывода ответа |
| `RESULT` | Определение файла вывода ответа на следующий запрос |
| `HEADER:` | Определение заголовка ответа |
| `UNLOAD:` | Определение разделителя |
| `CREATE` | Создание таблицы, индекса или другого объекта БД |
| `CREATE PROC <filename>` | Создание процедуры (текст берётся из файла) |

| Имя команды | Назначение команды |
|---|---|
| `CREATE TRIG <filename>` | Создание триггера (текст берётся из файла) |
| `ALTER` | Изменить описание таблицы |
| `ALTER PROC <filename>` | Изменение процедуры (текст берётся из файла) |
| `GRANT` | Предоставление привилегий пользователю |
| `DROP` | Удалить таблицу или индекс |
| `REVOKE` | Отменить привилегии пользователя |
| `PRESS` | Перестроить все индексы и сжать номера записей |
| `REBUILD` | Восстановить таблицу |
| `SELECT` | Выбор строк из таблицы |
| `INSERT` | Вставка строк в таблицу |
| `UPDATE` | Замена строк |
| `DELETE` | Удаление строк из таблицы |
| `LIST` | Вывод текущих установок утилиты `inl` |
| `!` | Начало комментария для вывода на экран |
| `EXEC` | Выполнить запрос |
| `EDIT` | Редактировать запрос |
| `OPTIMISTIC` | Установить режим обработки транзакций optimistic<br>Режим OPTIMISTIC устарел (использовать не рекомендуется). |
| `PESSIMISTIC` | Установить режим обработки транзакций pessimistic |
| `COMMIT` | Завершить текущую транзакцию |
| `ROLLBACK` | Откатить текущую транзакцию |
| `BACKUP` | Архивация базы данных |
| `TRUNCATE` | Очистка таблицы и усечение файлов |
| `PRIORITY:` | Установить приоритет запроса |
| `CORRECT` | Исправление индекса для указанной записи |
| `SH` | Выполнить команду ОС |
| `SHOW` | Показать описание таблицы |
| `EXIT` | Завершение работы |
| `CODEPAGE` | Установить кодовую таблицу |
| `SLEEP` | Приостановить выполнение |
| `DBINFO` | Получить информацию о БД |
| `IGNORE` | Игнорировать указанный код завершения |
| `BLOB` | Операции с BLOB, формат команды<br>`blob {insert\|clear\|append\|get}{rowid=<row_id> <user_name>.<table_name>.<column_name>\| column=<col_num>} [type=<blob_type>] {file=<file_name>\| <text_blob_body>}` |

 **© Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026**

# HISTORY

## Формат

```
HIST[ORY] [<номер>] [on|off|clear]
```

## Назначение

Предоставление справочной информации о ранее выполненных командах или повторное выполнение ранее выполненной команды программы `inl`.

## Описание

Команда выполняется только в среде ОС типа Linux, ЗОСРВ Нейтрино и при условии, что программа `inl` собрана с директивой USE_READLINE.

**Примечание**
Если при выполнении команды `history` список ранее выполненных команд пуст, значит программа `inl` собрана без директивы USE_READLINE (иначе в списке присутствовала бы, как минимум, команда `history`).

При вызове команды без параметра на консоли отображается список ранее выполненных пользователем ОС команд (история) программы `inl` с их порядковыми номерами. Все выполняемые команды запоминаются с первого запуска программы `inl` в файле `.inl_history` в домашнем каталоге пользователя ОС. Хранится (и, соответственно, выдается) максимум 200 последних выполненных команд. Нумерация команд непрерывно-последовательная (то есть после команды с номером 200 нумерация будет 201, 202 и так далее).

```
SQL> history
1: select * from auto;
2: dissep
3: show t
4: history
5: exit
6: select * from auto;
7: history
```

При вызове команды с указанием параметра <номер> выполняется команда с заданным номером (если команда с таким номером существует). Выполненной команде присваивается очередной порядковый номер.

В ОС типа Windows команда не работает, но список ранее выполненных команд можно получить, нажав клавишу **<F7>** для отображения окна с историей команд и выбрать оттуда нужную команду.

```
0: SYSTEM
1: outfile:tst
2: select count)*) from auto;
3: select count(*) from auto;
```

Опция on (действует по умолчанию) устанавливает режим протоколирования в файл `.inl_history` выполняемых команд. Если `inl` запущена с ключом -nohist, то команда **history on** отменяет действие этого ключа и возобновляет протоколирование.

Опция off отменяет режим протоколирования в файл `.inl_history` выполняемых команд.

Опция clear удаляет все записи в файле протоколирования `.inl_history`.

# IGNORE

## Формат

```
IGNORE <код завершения>
```

<Код завершения> – возможный код завершения, возвращаемый ядром СУБД ЛИНТЕР при обработке SQL-запроса.

## Назначение

Игнорирование заданного кода завершения при пакетной обработке SQL-запросов.

## Описание

При пакетной обработке SQL-скрипта `inl` выдает на консоль диагностические сообщения для всех ошибочных ситуаций, возникающих при обработке SQL-запросов, и формирует код возврата обработки SQL-скрипта, который может быть получен и проанализирован внешней программой (например, командным процессором), вызвавшей `inl` на выполнение (см. раздел «Коды завершения»).

Если в процессе обработки SQL-скрипта СУБД вернула хотя бы один ненулевой код завершения, то код возврата всего скрипта будет также ненулевым, что подразумевает наличие ошибки в SQL-скрипте.

Команда `IGNORE` заставляет игнорировать заданный код завершения, обеспечивая тем самым нулевой код возврата скрипта.

Диагностическое сообщение, соответствующие коду завершения, указанному в команде `IGNORE`, выдается с пометкой «игнорируется», например:

```
INL: состояние выполнения : 2202 (игнорируется)
```

С помощью `IGNORE` можно задать только один код завершения. Для нескольких кодов завершения надо использовать несколько команд `IGNORE`.

Установленные значения сохраняются только на время текущего сеанса работы `inl`.

Повторное выполнение команды `IGNORE` с указанием того же самого кода завершения отменяет его игнорирование.

Просмотр текущих установок команды `IGNORE` выполняется с помощью команды LIST.



**Примечание**
Если в SQL-скрипте используется команда `ignore`, рекомендуется перед завершением его работы отменить игнорирование указанного кода завершения (чтобы команда не влияла на выполнение других SQL-скриптов по тому же соединению с СУБД ЛИНТЕР).

## Пример

```
SQL> !запрет выдачи диагностического сообщения для кодов 501 и 73
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
SQL> ignore 501 игнорирование кода завершения 501
SQL> ignore 73  игнорирование кода завершения 73
SQL> …
SQL> ignore 501 отмена игнорирования кода завершения 501
SQL> …
```

# LIST

### Формат

`LIST`

### Назначение

Вывод текущих установок программы `inl`.

### Описание

Текущие установки программы `inl` содержат следующую информацию:

1) состояние переключателей режимов работы `inl`, задаваемых командами:
   - TIME;
   - PAGE;
   - FORMAT;
   - COUNT;
   - OPTIMISTIC;

**Примечание**
Режим OPTIMISTIC устарел (использовать не рекомендуется).

   - PESSIMISTIC;
   - BROWSE.
2) символ разделителя (заданный командой UNLOAD:) и выходной файл (заданный командой OUTFIL:);
3) текст последнего обработанного SQL-запроса (эта информация полезна при выполнении команды EXEC);
4) кодовая страница, в которой работает `inl`;
5) приоритет, по которому СУБД ЛИНТЕР обрабатывает SQL-запросы, поступающие от `inl`;
6) список кодов завершения СУБД ЛИНТЕР, информация о которых не должна выводиться на экран или в выходной файл.

### Пример

```
;вывод текущих установок на экран
SQL>LIST
                    -- Текущие установки INL --

 time          : вкл.
```

```
 page         : вкл.
 format       : выкл.
 count        : вкл.
 optimistic   : выкл.
 pessimistic  : выкл.
 priority     : 0
 browse       : выкл.
 outfil       :
 unload       : |
 Код. стр.    : MS-DOS 866
 Запрос: SELECT Tn,Fio,Dolgn,Ocl FROM Kadry WHERE
 Notd=111;
 Игнорировать коды завершения :2202
SQL>

;вывод текущих установок в файл
inl -u SYSTEM/MANAGER8 > output.txt
…
list
…
exit
```

# OPTIMISTIC

### Формат

`OPTIMISTIC`

### Назначение

Переключатель режима обработки транзакций.

### Описание

Эта команда переключает текущий режим работы СУБД на режим `Optimistic Concurrency Control` и обратно.

Выключенный (положение выкл.) переключатель `OPTIMISTIC` устанавливает режим работы СУБД ЛИНТЕР с возможностью «теплого» рестарта.

**Примечание**
Режим OPTIMISTIC устарел (использовать не рекомендуется).

Сразу после запуска `inl` переключатель режима обработки транзакций установлен в положение `выкл.`

# OUTFILE

### Формат

`{OUTFILE|OUTFIL}: [<спецификация файла>]|[CON]`

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

## Назначение

Спецификация выходного файла.

## Описание

По умолчанию результаты выполнения команд и/или SQL-операторов `inl` выводит на экран терминала пользователя (выходной файл по умолчанию). Для переназначения вывода используется команда `OUTFILE:`. Параметр <спецификация файла> задает местоположение и имя выходного текстового файла в формате соответствующей операционной системы, например, `C:\linter\test\crt_table.tst`.

Если в качестве выходного файла указано `CON`, то текущий выходной файл закрывается и вывод осуществляется на терминал.

В выходной файл записываются:

1) выборка данных select-запроса (в форматированном или неформатированном виде);
2) заголовок выборки данных (определенный командой HEADER:);
3) итоговая статистика.

Команда выполняется по следующим правилам:

- устройство и каталог (каталоги), указанные в <спецификации файла> должны существовать на момент выполнения команды (автоматически не создаются);
- если не указано местоположение выходного файла, он создается в текущем каталоге;
- данные в выходной файл записываются в коде ASCII;
- результаты выполнения SQL-оператора будут направляться в файл, определенный командой `OUTFILE:`, до тех пор, пока не будет произведено новое назначение выходного файла. При этом выходной файл, открытый предшествующей командой `OUTFILE:`, закроется и откроется новый специфицированный файл;
- если в `OUTFILE:` указан файл, который уже существует, то выходные данные добавляются в этот файл;
- комментарии, коды завершения и их диагностические сообщения всегда выдаются на экран терминала;
- команда `OUTFILE:` может быть введена как с терминала, так и из SQL-скрипта;
- если параметр <спецификация файла> не указан, то в качестве выходного файла используется экран терминала.

### Примечания

1. При запуске `inl` выходным файлом по умолчанию установлен экран терминала.
2. Узнать текущее имя выходного файла можно с помощью команды LIST.

## Примеры

```
SQL>outfile:/usr/home/Fill.ans (в среде ОС Linux, ЗОСРВ Нейтрино)

;создание выходного файла в текущем каталоге
SQL>outfile:tst
…
```

```
;создание другого выходного файла в заданном каталоге
SQL>outfile:d:/linter/script/crt_db.sql
…
;перенаправление вывода на экран терминала
SQL>outfile:con
…
;продолжение вывода в файл tst
SQL>outfile:tst
…
```

# PAGE

### Формат

```
PAGE
```

### Назначение

Разрешить/запретить разбивку выводимой информации по страницам.

### Описание

Команда PAGE выполняется только в строчном режиме функционирования `inl` и относится к SQL-запросам, потенциально возвращающим множественную выборку данных (несколько экранов терминала):

- SELECT;
- EXECUTE PROCEDURE;
- TEST TABLE.

Установленный в положение вкл. (включен) переключатель разбивки по страницам заставляет `inl` выдавать результаты выборки данных порциями по 20 записей.

**Примечание**
Если запись выборки данных занимает несколько строк экрана, то 20 записей выборки данных могут потребовать несколько экранов и в этом случае первые экраны выборки данных будут потеряны. В данной ситуации необходимо использовать команду OUTFILE: для вывода результатов выборки данных в файл с последующим просмотром их системными средствами или перейти в табличный режим функционирования `inl`.

После выдачи очередной порции записей выборки данных `inl` вступает в диалог с пользователем и ждет указаний о дальнейшей работе:

```
INL : нажмите любую клавишу (q для выхода):
```

Ввод символа 'q' эквивалентен отказу от выдачи следующей порции выборки данных и переходу к вводу новых команд для формирования другого SQL-запроса, нажатие любой другой клавиши продолжит выдачу записей выборки данных.

Выключенный (положение выкл.) переключатель разбивки по страницам устанавливает режим выдачи всех записей выборки данных на запрос без разбивки по страницам.


© Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

Если ранее была выдана команда OUTFIL, то разбиение по страницам не выполняется.

Команда PAGE работает как циклический двоичный переключатель, то есть каждое выполнение PAGE отменяет текущий режим и устанавливает противоположный. Установленный режим сохраняется до изменения его новой командой PAGE или конца работы inl.

Сразу после запуска утилиты inl переключатель разбивки по страницам по умолчанию установлен в положение вкл.

Для просмотра текущего состояния переключателя режима разбивки страниц используется команда LIST.

Смена значения переключателя производится по команде PAGE.

**Примечание**
В режиме приема команд из файла (а не с терминала) команда PAGE игнорируется.

### Пример

Выдать список служащих, получающих минимальную зарплату:

```
SQL>Select cast ' Минимальная зарплата: ' as char(20),
1>to_char(MIN(Salary)) from person
2>union
3>SELECT DISTINCT FirstNam,Name FROM Person
4>WHERE Salary =(SELECT MIN(Salary) FROM Person) order
5>by 1 asc;

| Минимальная зарплата|      10200         |
|ANNETTE              |PERREAULT           |
|ART                  |SPIEGEL             |
|BILL                 |MOUREAU             |
|BRENDA               |MOUREAU             |
|CHARLES              |WAGNER              |
|CHARLY               |FERRARI             |
|CLARA                |WAGNER              |
|DALIAH               |COLVILLE            |
|EDDY                 |ALEXANDER           |
|FORTUNA              |RAEBIGER            |
|FRANCOISE            |QUIHLLAULT          |
|GERARDIII            |TERZI               |
|JACK                 |LAWLER              |
|JEFFERSON            |LAWLER              |
|JO                   |RAY                 |
|JOHN                 |QUILLION            |
|LILIAN               |KOLENCE             |
|MARTHA               |DAVENPORT           |
|PUALA                |HOROWITZ            |
```

```
INL : нажмите любую клавишу (q для выхода):
```

# PESSIMISTIC

### Формат

```
PESSIMISTIC
```

### Назначение

Команда изменения режима обработки запросов.

### Описание

При использовании сложных (много взаимосвязанных таблиц) и/или распределенных транзакций (то есть транзакций, которые включают изменения данных более чем на одном узле) рекомендуется PESSIMISTIC-режим обработки данных.

При этом режиме все модификации пишутся сразу в БД, но все измененные (первым пользователем-инициатором PESSIMISTIC-режима) записи блокируются. Другие пользователи в этот момент не могут читать или модифицировать записи, измененные первым пользователем до конца транзакции.

Транзакция, модифицирующая данные в PESSIMISTIC-режиме, работает с блокировками `Exclusive-Lock` на уровне записей (то есть другие пользователи не имеют доступа к записи, пока работающая с ней Exclusive-транзакция не подаст запрос `COMMIT` или `ROLLBACK`).

При выполнении `COMMIT` модификации, произведенные транзакцией, остаются в БД (в системный журнал БД только ставится отметка о конце транзакции).

При выполнении `ROLLBACK` модификации, произведенные транзакцией, удаляются из БД в соответствии с системным журналом.

# PRECOUNT

### Формат

```
PRECOUNT
```

### Назначение

Управляет выводом информации о размере выборки данных поискового запроса; до выгрузки выборки данных или после. На основании полученной информации

```
количество строк: N
```

можно принять решение прервать выборку данных после выдачи первой порции данных. Например, если количество записей выборки данных будет 1 млн., то возможно надо отменить выполнение данного поискового запроса и сформулировать новый для изменения поисковых условий с целью уменьшения размера выборки данных.

Команда работает как переключатель – изменяет текущий режим на противоположный.

По умолчанию при запуске `inl` установлен режим вывода размера выборки данных после отображения всех её записей.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

### Описание

При выполнении некоторых поисковых запросов иногда необходимо узнать размер будущей выборки данных, например, если количество записей выборки данных будет 1 млн., то просмотреть такую выборку будет проблематично и, возможно, надо отменить выполнение такого поискового запроса и сформулировать новый запрос с измененными поисковыми условиями с целью уменьшения объема выборки.

### Примеры

```
SQL>precount
SQL>select make, model from auto limit 3;
INL : количество строк: 3
MAKE                 MODEL
 ----                 -----
|FORD                |MERCURY COMET GT V8 |
|ALPINE              |A-310               |
|AMERICAN MOTORS     |MATADOR STATION     |
 INL : выдано строк: 3
SQL>precount
SQL>select make, model from auto limit 3;
MAKE                 MODEL
 ----                 -----
|FORD                |MERCURY COMET GT V8 |
|ALPINE              |A-310               |
|AMERICAN MOTORS     |MATADOR STATION     |
 INL : выдано строк: 3
```

# PRIORITY

### Формат

```
PRIORITY: [<значение>]
```

### Назначение

Изменение приоритета канала.

### Описание

Команда устанавливает новый приоритет каналу, выделенному СУБД ЛИНТЕР для работы с `inl`.

Параметр <значение> – целочисленное положительное число в диапазоне от 0 до 249 (0 – минимальный приоритет, 249 – максимальный). Назначенный приоритет присваивается всем выполняемым SQL-запросам, подаваемым программой `inl`.

Установленный приоритет действует до назначения другого приоритета или до конца текущего сеанса `inl`.

Сразу после запуска `inl` приоритет канала по умолчанию установлен в `0`.

Для просмотра текущего значения приоритета канала используется команда LIST.

### Примеры

```
SQL>list
…
priority  :0
…
; Получить список владельцев автомобилей марки «FORD»,
; чьи имена начинаются на D, и моделей их машин.
SQL>SELECT FirstNam, Name, Model FROM  Person, Auto
1>WHERE FirstNam LIKE 'D%' AND Make ='FORD' AND
2>Person.PersonId =Auto.PersonId;
INL : start time : 14:32:20 end time : 14:32:22
FIRSTNAM  NAME    MODEL
--------  ----    -----
|DIANA      |HEAFNER    |MUSTANG BOSS 351       |
|DEBORAH    |SPIEGEL    |GRAN TORINO SPORT V8   |
|DIANA      |WATSON     |PINTO RUNABOUT         |
|DAISY      |WOOLSEY    |LTD COUNTRY SQUIRE     |
|DIANA      |COLVILLE   |GRAN TORINO SPORT V8   |
|DANIEL     |SRC        |MERCURY MONTEREY       |
|DAN        |TANIMOTO   |CAPRI RS 2600          |
|DANIEL     |ALDEN      |MERCURY MONTEREY       |
INL : number of rows shown: 8
SQL>

SQL>priority:240
SQL>list
…
priority  :240
…
; этот же запрос будет обработан ядром с более высоким приоритетом
SQL>exec
…
; установка приоритета по умолчанию
SQL>priority:
SQL>list
…
priority  :0
…
```

# RESULT

### Формат

```
RESULT: [<спецификация файла>]
```

### Назначение

Спецификация выходного файла для следующего SQL-запроса.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

### Описание

Команда RESULT позволяет записывать результаты выполнения SQL-запросов в индивидуальные файлы.

Она выполняется аналогично команде OUTFILE, но распространяется только на один SQL-запрос, выполняемый непосредственно за этой командой. После обработки такого запроса выдача результатов последующих SQL-запросов будет продолжена на экран терминала или в файл, указанный в команде OUTFILE (если команда OUTFILE была ранее подана).

### Пример

```
outfile:auto.tst
select count(*) from auto;
-- сформировать список моделей в файле model.lst
result:model.lst
select distinct model from auto;
-- сформировать список изготовителей в файле make.lst
result:make.lst
select distinct make from auto;
-- продолжить выдачу результатов в файл auto.tst
…
```

# SH

### Формат

```
SH <команда операционной системы>
```

### Назначение

Выполнение команды операционной системы.

### Описание

Команда позволяет, не выходя из программы inl, выполнить любую допустимую команду операционной системы.

### Пример

Использование обращения к командному интерфейсу операционной системы.

```
SQL>SH ls -l
total 8
-rw-r--r--  1  root  sys  978   May  19 20:22  c.dir
-rw-rw-rw-  1  root  sys  2035  May  12 10:43  com_proc.c
-rw-rw-rw-  1  root  sys  992   May  15 12:55  com_proc.o
-rw-rw-rw-  1  root  sys  3128  May  12 10:43  dbc.h
-rw-rw-rw-  1  root  sys  23203 May  12 10:43  dbc_tcp.c
-rw-rw-rw-  1  root  sys  14400 May  15 12:55  dbc_tcp.o
-rw-rw-rw-  1  root  sys  2774  May  12 10:43  dbcs_err.h
-rw-rw-rw-  1  root  sys  2797  May  12 10:43  dbcscomm.h
SQL>
```

# SHOW

## Формат

```
SHOW <табличный объект>

<табличный объект>::=
<имя базовой таблицы>
|<имя синонима таблицы>
|<имя представления>
```

## Назначение

Получение справочной информации об объекте БД.

## Описание

Все имена в <табличном объекте> допускает символ обобщения – `%`.

Для получения информации о всех табличных объектах БД необходимо подать запрос `SHOW %`.

В разделе «Характеристики столбца» (см. «Примеры») для каждого столбца табличного объекта выдается следующая информация:

1) тип данных столбца и, в зависимости от типа данных, длина столбца и точность представления данных;
2) атрибуты столбца (первичный ключ, ссылочный ключ, автоиндексация и т.п.);
3) признак индексируемости столбца и параметры индекса:
   - номер уровня характеризует объем индексного файла и, в общем случае, информирует о том, сколько страниц индексного файла будет предварительно считано при поиске значения данного столбца;
   - номер вершины показывает относительный номер страницы индексного файла, где находится вершина данного столбца;
   - информация о фразовых индексах и их типах.
4) о кодировках для СУБД ЛИНТЕР.

Если в табличном объекте определены CHECK-ограничения, опции GENERATED-столбцов, атрибуты столбцов `AUTOINC RANGE`, `AUTOINC INITIAL`, то выводится информация об этих параметрах, например:

```
I INTEGER CHECK(("I" > 0) AND ("I" < 10)) DEFAULT 1
L INTEGER AUTOINC INITIAL(1)
CH CHAR(99)
NCH NCHAR(21) DEFAULT 'abcd                  '
B BIGINT CHECK("B" < 100) DEFAULT 100 CHECK(("I"+1) > 0)
```

**Примечание**
Для распределенной таблицы выдается дополнительная информация в виде:

```
* Узел : 'Имя_узла' (ID=Идентификатор_узла)
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

Если <именем табличного объекта> является <имя синонима>, выводится сообщение о том, синонимом какой таблицы (представления) он является, например:

```
show all_users
Описание представления "SYS.ALL_USERS"
```

**Примечание**
Строка «Количество индексов» информирует о количестве простых одностолбцовых индексов.

Если нужно задать в команде SHOW точное имя таблицы, то следует перед подчеркиваниями в случае их наличия в имени таблицы (и процентами, хотя это редкий вариант) задавать символы обратного слэша. Конструкцию ESCAPE '\' добавить во все SQL-запросы с LIKE в inl и другие утилиты, где используется LIKE, так как по умолчанию в режиме /COMPATIBILITY=STANDARD конструкция ESCAPE '\' не действует. В трансляторе SQL LIKE для случая, когда все метасимволы в шаблоне экранированы, преобразовывать в равенство (сейчас это делается только тогда, когда метасимволов в шаблоне нет). Во всех тестах добавить вручную '\' перед '_' в команду SHOW, если там предполагается именно точное имя таблицы. Если же в команде SHOW задается шаблон имени таблицы, содержащий метасимволы '_' и '%,' то при работе в режиме /COMPATIBILITY=STANDARD следует всегда ставить '%' в конце шаблона.

### Примеры

1) Справочная информация о таблице auto:

```
show auto

Описание таблицы «SYSTEM.AUTO»
* Номер таблицы : 245
* Предельный ROWID  : 1022
* Последний занятый ROWID : 1000
* Номер текущей строки  : 1000
* Процент заполнения страницы : 100
* Порог освобождения страницы   : 0 (не установлен)
* Длина строки  : 113
* Количество столбцов : 13
* Количество индексов : 1
* Файлов индексов: 1 ("SY00" 2)
* Файлов данных: 1 ("SY00" 13)
INL : нажмите любую клавишу (q для выхода) :
Характеристики столбца
----------------------
MAKE  CHAR(20)
MODEL CHAR(20)
…
TABLE CHARSET CP866(#2)
Характеристики простых индексов
Имя индекса Тип
PERSONID  Primary
```

```
SQL>
```

2) Справочная информация о таблице с ограничениями целостности:

```
create or replace table "TA1" ("I" int default 1 check ( ("I" > 0)
 AND ("I" < 10) ),
"L" int autoinc default 1,
"CH" char(99),
"NCH" nchar(21) default n'abcd',
"B" bigint default 100 check ( "B" < 100 ), check (I+1 > 0));
select cast LINTER_DICT_INFO(1,$$$S11,5) as char(80) from $$$SYSRL
 where $$$S13='TA1';
show TA1
Описание таблицы «TA1»
…
Характеристики столбца
----------------------
I       INTEGER CHECK(("I" > 0) AND ("I" < 10)) DEFAULT 1
L       INTEGER AUTOINC INITIAL(1)
CH      CHAR(99) CHARSET CP1251(#4)
NCH     NCHAR(21) DEFAULT 'abcd                  '
B       BIGINT CHECK("B" < 100) DEFAULT 100
TABLE CHARSET CP1251(#4)
CHECK(("I"+1) > 0)
```

3) Справочная информация о таблице, содержащей столбец геометрического типа данных:

```
create table tp (p polygon);
show TP
Описание таблицы «SYSTEM.TP»
…
Характеристики столбца
----------------------
P   VARBYTE(1028) / POLYGON
TABLE CHARSET CP1251(#4)
```

# SLEEP

## Формат

```
SLEEP [<время>]
<время> – целочисленное неотрицательное значение.
```

## Назначение

Приостанов выполнения программы `inl` на указанное в параметре <время> количество секунд. Если параметр <время> не задан, то по умолчанию пауза равна нулю.

## Описание

В результате выполнения команды `SLEEP` на экране появляется сообщение:

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
INL: Пауза на <время> сек.
```

и функционирование программы приостанавливается на <время> секунд.

# STRUCTURE TABLE

## Формат

```
STRUCTURE TABLE <табличный объект>
<табличный объект>::=[<схема объекта>.]<имя базовой таблицы>
```

## Назначение

Выгрузка на консоль утилиты текста SQL-оператора создания табличного объекта.

## Описание

Все имена в <табличном объекте> допускают символ обобщения `%`.

Команда выполняется от имени текущего пользователя, поэтому для выгрузки чужих схем необходимо иметь доступ к системным таблицам БД.

Если задан несуществующий в БД <табличный объект>, соответствующий код завершения не выдается.

Имена элементов <табличного объекта> регистрозависимы.

Если нужно задать в команде `STRUCTURE TABLE` точное имя таблицы, то следует перед подчеркиваниями в случае их наличия в имени таблицы (и процентами, хотя это редкий вариант) задавать символы обратного слэша. Конструкцию `ESCAPE '\'` добавить во все SQL-запросы с `LIKE` в `inl` и другие утилиты, где используется `LIKE`, так как по умолчанию в режиме `/COMPATIBILITY=STANDARD` конструкция `ESCAPE '\'` не действует. В трансляторе SQL `LIKE` для случая, когда все метасимволы в шаблоне экранированы, преобразовывать в равенство (сейчас это делается только тогда, когда метасимволов в шаблоне нет). Во всех тестах добавить вручную '\' перед '_' в команду `STRUCTURE TABLE`, если там предполагается именно точное имя таблицы. Если же в команде `STRUCTURE TABLE` задается шаблон имени таблицы, содержащий метасимволы '_' и '%,' то при работе в режиме `/COMPATIBILITY=STANDARD` следует всегда ставить '%' в конце шаблона.

## Примеры

1) structure table AUTO

```
SQL> structure table AUTO
 CREATE  TABLE "SYSTEM"."AUTO" CHARACTER SET "CP866" (
  "MAKE" CHAR(20) CHARACTER SET "CP866" DEFAULT NULL,
  "MODEL" CHAR(20) CHARACTER SET "CP866" DEFAULT NULL,
  "BODYTYPE" CHAR(15) CHARACTER SET "CP866" DEFAULT NULL,
  "CYLNDERS" INTEGER DEFAULT NULL,
  "HORSEPWR" INTEGER DEFAULT NULL,
  "DSPLCMNT" INTEGER DEFAULT NULL,
  "WEIGHT" INTEGER DEFAULT NULL,
  "COLOR" CHAR(10) CHARACTER SET "CP866" DEFAULT NULL,
  "YEAR" INTEGER DEFAULT NULL,
```

```
  "SERIALNO" CHAR(16) CHARACTER SET "CP866" DEFAULT NULL,
  "CHKDATE" INTEGER DEFAULT NULL,
  "CHKMILE" INTEGER DEFAULT NULL,
  "PERSONID" INTEGER,
  PRIMARY KEY ("PERSONID"));
```

2) structure table SYSTEM."AUTO"
3) struct tab SYS%.A%

# TIME

## Формат

```
TIME [<формат>]
<формат>::= ['']<символьная строка>['']
```

Одинарные кавычки используются в случае, если <формат> содержит пробелы.

Допустимые элементы формата:

| Элемент формата | Описание |
|---|---|
| YYYY, YYY, YY, Y | Цифра года (2015, 015, 15, 5) |
| Q | Квартал (от 1 до 4) |
| MM | Месяц (от 1 до 12) |
| DDD | День года (от 001 до 366) |
| DD | День месяца (от 01 до 31) |
| D | День недели (от 1 до 7) |
| HH, HH12 | Относительный час дня (am, pm)(от 00 до 12) |
| HH24 | Абсолютный час дня (от 00 до 24) |
| MI | Минуты (от 00 до 59) |
| SS | Секунды (от 00 до 59) |
| MS | Миллисекунды (от 00 до 100) |
| TT | Тикеты (сотые доли секунды) (от 00 до 99) |
| T | Тикеты (десятые доли секунды) (от 0 до 9) |
| CEN | Век (столетие) (от 01 до 21) |
| TH | Тысячелетие (от 1 до 3) |
| MONTH, Month, month | Полное название месяца |
| MON, Mon, mon | Сокращенное (три символа) название месяца |
| DAY, Day, day | Полное название дня недели |
| DY, Dy, dy | Сокращенное (три символа) название дня недели |
| MONR | Номер месяца римскими цифрами |
| CENR | Номер века (столетия) римскими цифрами |
| ER | Уточнение века (две буквы A.D, P.D) |
| MID | Уточнение даты (две буквы am, pm) |

Разделителями элементов формата могут быть двоеточие, дефис, пробел, точка и прямая/косая черта.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

## Назначение

Разрешить/запретить выдачу временной статистики о выполняемых SQL-запросах.

## Описание

Команда TIME управляет режимом отображения временной статистики о выполняемых SQL-запросах. Она работает только в строчном режиме отображения информации. Если переключатель режима отображения времени включен (положение вкл.), то inl проводит замер и выдачу информации о времени начала и конца выполнения запроса (с точностью до секунды), например,

```
INL: начальное время : 16.08.42 конечное время :16.08.43
```

Для подсчета времени используется локальное время, установленное в компьютере (в операционной системе), на котором выполняется inl. Под замером времени следует понимать, что inl фиксирует время передачи запроса на выполнение и время получения первой записи выборки данных на этот запрос.

Если переключатель времени выключен (положение выкл.), то замер и, соответственно, выдача показаний времени обработки запроса не происходит.

Команда TIME без <формата> работает как циклический двоичный переключатель, то есть каждое выполнение TIME отменяет текущий режим отображения и устанавливает противоположный. Установленный режим сохраняется до конца работы inl.

Сразу после запуска inl переключатель режима отображения времени по умолчанию установлен в положение вкл.

Вызов команды TIME с <форматом> устанавливает режим отображения даты в установленном формате в положение вкл. (независимо от текущего режима). Если <формат> является некорректным, то выдается диагностическое сообщение «Неверное значение» и состояние режима не меняется.

Для просмотра текущего состояния переключателя режима отображения времени используется команда LIST.

### Примечание

Результаты выполнения команды TIME всегда выводятся на экран терминала, то есть в выходной файл, создаваемый по команде OUTFIL, они не попадают. Однако, если inl запущена с командой перенаправления вывода (например, inl > out.txt), то в этом случае результаты выполнения TIME будут помещены в указанный файл.

## Примеры

1)

```
…
SQL>list
          --Текущие установки INL--
time       :вкл.
…
SQL>select count (*) from auto
1>where color in ('YELLOW','GREEN','BLACK');
INL: начальное время : 10.55.58 конечное время :10.55.58
```

```
|      367|
INL : выдано строк: 1
SQL>time
SQL>list
         --Текущие установки INL--
time      :выкл.
…
```

2)

```
SQL>select count (*) from auto
1>where color in ('YELLOW','GREEN','BLACK');
|      367|
INL : выдано строк: 1
SQL>
```

3)

```
time 'DD-MM-YYYY Q'
16-01-2015 1
time 'DD-MM-YY HH'
16-01-2015 03
time 'DD-MM-YY HH CEN'
16-01-2015 03 21
time 'DD-MONTH-YY HH:MM:SS'
16-YANUARY-15 03:01:10
time 'DD-MONR-YYYY'
16-I-2015
time 'DD-MM-YYYY CENR'
16-01-2015 XXI
time 'DD-MM-YYYY CENR MID'
16-01-2015 XXI pm
time 'DD-MM-YYYY ER'
16-01-2015 A.D.
```

# UNLOAD

### Формат

```
UNLOAD: [<символ>]
```

### Назначение

Определение разделителя.

### Описание

При выдаче результатов выполнения SELECT-операторов в строчном режиме `inl` разделяет столбцы выдаваемой информации с помощью специального символа – разделителя столбцов. По умолчанию разделителем является символ «|». С помощью команды `UNLOAD` пользователь может установить любой другой разделитель, например,

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

«:». Новый разделитель действует до его изменения командой UNLOAD либо до конца текущего сеанса inl.

Если параметр <символ> не задан, используется разделитель по умолчанию.

### Примеры

```
;действие разделителя по умолчанию
SQL>select make, year from auto;
|FORD          |     71|
|ALPINE        |     70|
…
;разделитель – символ «:»
SQL>unload::
SQL>select make, year from auto;
:FORD          :     71:
:ALPINE        :     70:
…
; отказ от разделителя
SQL>unload:<пробел>
SQL>select make, year from auto;
FORD              71
ALPINE            70
…
; разделитель – символ «*»
SQL>unload:*!%
SQL>select make, year from auto;
*FORD          *     71*
*ALPINE        *     70*
…

; вернуться к разделителю по умолчанию
SQL>unload:
SQL>
```

# USERNAME

### Формат

```
USERNAME <имя пользователя>[/<пароль>]
```

### Назначение

Инициирование работы под другим именем пользователя в текущей БД без выхода из inl.

### Описание

Команда выполняется следующим образом:

1) если параметр <имя пользователя>[/<пароль>] задан полностью (то есть одновременно имя и пароль), то это значение передается ядру СУБД, с которой

в данный момент работает inl для проверки в БД пользователя с указанными регистрационными данными;

2) если в команде задано только <имя пользователя>, то после нажатия клавиши **<Enter>** в ответ на приглашение:

```
SQL>
```

необходимо ввести пароль пользователя (не более 18 символов) без двойных кавычек;

3) если имя пользователя и пароль введены правильно, inl продолжает работу с БД от имени нового пользователя. При этом: текущий приоритет канала сбрасывается, транзакционный режим ставится по умолчанию (autocommit, если другой режим не был задан переменной среды окружения LINTER_INLDEFCONNMODE). Иначе будет выдано сообщение об ошибке: «Неверное имя пользователя» или «Неверный пароль»;

4) если команда USERNAME завершилась неудачно (имя или/и пароль не совпали), текущий канал не закрывается и можно продолжать работать под текущими регистрационными данными.

### Примеры

```
inl -u SYSTEM/MANAGER8
SQL>time
SQL> select count(*) from AUTO;
|        1000|
INL  : выдано строк: 1
SQL>create user "Склад" identified by 'чы34эъ78';
SQL>username "Склад"/"чы34эъ78"
SQL> select count(*) from AUTO;
INL  : состояние выполнения   :2202
неизвестная таблица
SQL> select count(*) from SYSTEM.AUTO;
|        1000|
INL  : выдано строк: 1
SQL>username SYSTEM
Пароль пользователя: ****** (введено MANAGER8)
SQL> select count(*) from AUTO;
|        1000|
INL  : выдано строк: 1
SQL>
```

# Работа с BLOB-данными

## Командный режим

### Формат

```
 BLOB {INSERT | CLEAR | APPEND | GET}
{ROWID=<rowid записи> [<имя пользователя>.]<имя таблицы>.<имя
 столбца>
|COLUMN=<номер столбца>}
```

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
[TYPE=<тип BLOB-данных>]{FILE=<имя файла> | <BLOB-значение>}
<BLOB-значение>:=непрерывная последовательность шестнадцатеричных
 цифр
<тип BLOB-данных>::=целочисленное значение
```

## Назначение

Выполнение основных операций с BLOB-данными в командном режиме.

## Описание

Команда CLEAR очищает BLOB-значение.

Команда APPEND добавляет порцию BLOB-данных в конец BLOB-значения.

Команда INSERT эквивалентна последовательности команд CLEAR и APPEND.

Команда GET извлекает BLOB-значение. В этой команде имя пользователя можно не указывать (выполняется обращение к таблице текущего пользователя).

Опция TYPE задает тип добавляемых BLOB-данных. Учитывается только при выполнении команды BLOB INSERT.

Если задана опция COLUMN, все операции с BLOB-данными применяются к указанному столбцу текущей записи последней DML-операции. Отсчет столбцов начинается с 1.

Если задана опция ROWID, все операции с BLOB-данными применяются к столбцу <имя столбца> в указанной в <rowid записи> записи последнего выполненного SQL-запроса (выборки). В этом случае, при необходимости, будет выполнен отдельный select-запрос для получения записи с указанным <rowid записи>.

В командах INSERT и APPEND параметр <имя файла> задает спецификацию файла содержащего добавляемые (в бинарном виде) BLOB-данные.

В команде GET параметр <имя файла> задает спецификацию файла, в который должны быть записаны (в бинарном виде) извлекаемые BLOB-данные. Если спецификация файла не задана, BLOB-данные будут извлекаться СУБД ЛИНТЕР в текстовом виде в стандартный выходной поток с помощью функции printf.

Имена пользователей, таблиц и столбцов в команде могут задаваться в двойных кавычках.

Если спецификация файла (путь и/или имя) содержат пробелы и/или символы кириллицы, то её необходимо обрамлять двойными кавычками, а двойные кавычки в именах – экранировать, например,

```
SQL> blob insert column=2 file="c:\Program files\Film.avi";
SQL>blob insert column=2 file="c:\Коллекция музыки\Группировка
\"Ленинград\".mp3";
```

Для инициализации добавления BLOB-порций необходимо использовать NULL-значение, а не пустую строку.

Например,

```
1) в этом случае будет добавлена порция BLOB-данных
```

```
с предшествующим пробелом
(NBL1.SQL = <пробел> + NBL.SQL):
username SYSTEM/MANAGER8
create or replace table tbl(i int, b blob);
insert into tbl values(1, '');
blob append column=2 file=NBL.SQL;
blob get rowid=1 system.TBL.B file=NBL1.SQL;

2) в этом случае добавленная и считанная порции BLOB-данных
будут идентичными (NBL1.SQL =  NBL1.SQL):
username SYSTEM/MANAGER8
create or replace table tbl(i int, b blob);
insert into tbl values(1, NULL);
blob append column=2 file=NBL.SQL;
blob get rowid=1 system.TBL.B file=NBL1.SQL;
```

Данные в BLOB-файл добавляются порциями, максимальный размер – 16*4048 байт. Если добавляемое BLOB-значение превышает максимальный размер порции, то оно будет добавляться несколькими порциями. При работе в режиме `autocommit` после вставки каждой порции будет подана команда `COMMIT`.

## Примеры

1)

```
username SYSTEM/MANAGER8
create or replace table test(i int, bl1 blob, bl2 blob);
insert into test(i,bl1,bl2) values(1,NULL,NULL);
blob insert column=2 333333ABCD;
blob append column=3 444444ABCD;
select rowid, i, lenblob(bl1), getblob(bl1,1,5),
  lenblob(bl2), getblob(bl2,1,5) from TEST;
blob clear rowid=1 system.TEST.BL1;
blob append rowid=1 SYSTEM."TEST".bl2 ffffff;
!blob 1:
blob get rowid=1 system.TEST.BL1;
!blob 2:
blob get rowid=1 system.TEST.BL2;
!blob to file "blob.txt" (8 bytes):
blob get rowid=1 system.TEST.BL2 file=blob.txt;
insert into test(i,bl1) values(2,NULL);
!read blob from file:
blob append column=2 file=blob.txt;
!check blob value:
blob get rowid=2 system.TEST.BL1;
```

Результат работы примера:

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

```
ROWID       I
 -----       -
|          1|          1|          5| 33 33 33 AB CD|          5|
 44 44 44 AB CD|
blob 1:
blob 2:
444444ABCDFFFFFF
```

2)

```
create or replace table test_findrtf(b_rtf blob, b_pdf blob,
  b_doc blob, b_xls blob, b_ppt blob, b_ps blob);

insert into test_findrtf values(NULL,NULL,NULL,NULL,NULL,NULL);

blob insert column=1 file=1.rtf
blob insert column=2 file=1.pdf
blob insert column=3 file=1.doc
blob insert column=4 file=1.xls
blob insert column=5 file=1.ppt
blob insert column=6 file=1.ps
select * from test_findrtf;
```

3)

```
username SYSTEM/MANAGER8
create or replace table test(i int, bl1 blob, bl2 blob);
insert into test(i,bl1,bl2) values(1,NULL,NULL);
!insert (1,1):
blob insert column=2
1
11
111
abcd;
!append (1,2):
blob append column=3 222222ABCD;
select rowid, i, lenblob(bl1), getblob(bl1,1,5),
  lenblob(bl2), getblob(bl2,1,5) from TEST;
insert into test(i,bl1,bl2) values(2,NULL,NULL);
!insert (2,1):
blob insert column=2 333333ABCD;
!append (2,2):
blob append column=3 444444ABCD;
select rowid, i, lenblob(bl1), getblob(bl1,1,5),
  lenblob(bl2), getblob(bl2,1,5) from TEST;
!get blobs, row 1:
select i, bl1, bl2 from TEST where rowid=1;
blob get column=2;
blob get column=3;
```

```
!get blobs, row 2:
select i, bl1, bl2 from TEST where rowid=2;
blob get column=2;
blob get column=3;
!clear (1,1):
blob clear rowid=1 system.TEST.BL1;
!clear (2,2):
blob clear rowid=2 system.TEST.BL2;
select rowid, i, lenblob(bl1), getblob(bl1,1,5),
  lenblob(bl2), getblob(bl2,1,5) from TEST;
!append (1,2):
blob append rowid=1 SYSTEM."TEST".bl2
000
0001234;
!insert (2,1):
blob insert rowid=2 SYSTEM."TEST".bl1
000012
3456;
select rowid, i, lenblob(bl1), getblob(bl1,1,5),
  lenblob(bl2), getblob(bl2,1,10) from TEST;
!get blob (1,2):
blob get rowid=1 SYSTEM."TEST".bl2;
!get blob (2,1):
blob get rowid=2 SYSTEM."TEST".bl1;
```

4)

```
create or replace table test(i int, bl1 blob, bl2 blob);
insert into test(i,bl1,bl2) values(1,NULL,NULL);
blob insert column=2 type=101 111111abcd;
blob insert column=3 type=111 111111abcdef;
insert into test(i,bl1,bl2) values(2,NULL,NULL);
blob insert rowid=2 SYSTEM."TEST".bl1 type=201 111111abcd;
blob insert rowid=2 SYSTEM."TEST".bl2 type=211 111111abcdef;
```

# Табличный режим

Для работы с BLOB-данными (просмотр, добавление, удаление, выгрузка) в табличном режиме необходимо установить табличный режим просмотра (browse вкл.), для чего воспользоваться командой BROWSE для перехода в этот режим (если текущее состояние browse выкл.), и установить текущей ту строку таблицы, для которой будет выполняться операция с BLOB-данными. Для существующих в таблице строк текущая строка устанавливается с помощью команды `SELECT`; если предполагается добавлять BLOB-данные для новой строки таблицы, то предварительно должна быть выполнена команда `INSERT`, в которой значение BLOB-столбца пропущено или задано как NULL-значение. Вместе с BLOB-столбцом можно выбирать столбцы других типов данных. В появившемся окне перейти в BLOB-столбец (используя клавиши **<стрелка влево/вправо>** или **<Ctrl>+<стрелка влево/вправо>**), в результате высветится окно (рисунок 6) для работы с BLOB-данными.

 © Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026



Рисунок 6. Окно для работы с BLOB-данными

В окне (рисунок 6) в столбце BLOB-данных отображаются атрибуты загруженных BLOB-данных (в частности, первое число – это длина BLOB-данных в байтах). Панель внизу экрана содержит справочную информацию.

Клавишные команды работы с BLOB-данными (все они относятся только к выделенной строке загруженной таблицы):

- **<Alt>+<S>** – просмотр BLOB-данных. Независимо от характера данных, они высвечиваются в текстовом виде в стандартном окне просмотра файлов встроенного редактора операционной системы;
- **<Alt>+<C>** – удаление BLOB-данных. После удаления BLOB-столбец получает NULL-значение;
- **<Alt>+<L>** – загрузка BLOB-данных. При выполнении этой команды высвечивается стандартное окно для поиска и выбора файла, содержащего загружаемые данные;
- **<Alt>+<U>** – выгрузка BLOB-данных. При выполнении этой команды высвечивается стандартное окно для спецификации файла, в который должны быть выгружены данные.
- **<Alt>+<I>** – добавление BLOB-данных. При выполнении этой команды высвечивается стандартное окно для спецификации файла, который должен быть добавлен в конец BLOB-столбца.

# Автодополнение ввода

Автодополнение (в том числе, и в верхнем регистре) имен ключей, некоторых SQL-операторов и их ключевых слов, а также имен файлов с префиксом "_" поддерживается после нажатия клавиши **<Tab>** только в среде ОС типа Linux, ЗОСРВ Нейтрино и при условии, что утилита `inl` собрана с директивой USE_READLINE.

Например, ввод этих двух команд является эквивалентным:

```
select count(*) from auto where model='FORD';
sel<Tab> cou<Tab>(*) fr<Tab> wh<Tab> model='FORD';
```

# Сообщения программы

## Информационные сообщения

Информационные сообщения содержат справочную информацию и сведения о текущей работе программы `inl`. Смысл этих сообщений понятен из контекста интерактивного сеанса.

## Диагностические сообщения

Диагностические сообщения подразделяются на две группы: сообщения, порожденные неправильными командами пользователя и сообщения, связанные с проблемами функционирования собственно программы `inl` в операционной системе (системные ошибки). В первом случае пользователь должен проанализировать сообщение об ошибке, исправить ее и повторить выполнение команды, во втором случае следует обратиться к администратору СУБД ЛИНТЕР.

### Пользовательские ошибки

```
INL : слишком длинная строка ответа
INL : не открыт выводной файл. Код ошибки nnnn
INL : имя пользователя отсутствует в командной строке
INL : командный файл отсутствует в командной строке
INL : имя сервера отсутствует в командной строке
INL : длинное имя сервера
INL : неверное имя пользователя
INL : неверный пароль
INL : канал не открыт. ЛИНТЕР не загружен
INL : командный файл уже активен
INL : неверная команда
INL : длинное имя файла
INL : неверный синтаксис
INL : имя отношения не определено
INL : неверное имя синонима
```

### Системные ошибки

Ниже под значением nnnn понимается числовой код завершения, выдаваемый операционной системой для СУБД ЛИНТЕР.

```
INL : канал не открыт. Код ошибки nnnn
INL : командный файл не открыт. Код ошибки nnnn
INL : ошибка закрытия канала. Код ошибки nnnn
INL : ошибка операционной системы. Код ошибки nnnn
INL : ошибка открытия канала. Код ошибки nnnn
INL : ошибка завершения транзакции. Код ошибки nnnn
INL : ошибка отката транзакции. Код ошибки nnnn
```


© Командный интерфейс. АО НПП «РЕЛЭКС», 1990-2026

# Коды завершения

Помимо диагностических сообщений, выдаваемых на терминал или в выходной файл и предназначенных для визуального контроля выполнения программы, `inl` формирует код завершения (код возврата) для проверки результата своей работы другими программными средствами. Имя переменной, в которую помещается код возврата, зависит от операционной системы (см. соответствующую документацию). В таблице 5 приведены коды завершения, их символьные обозначения и значения.

Таблица 5. Дополнительная информация о SQL-запросах

| Код завершения | Символьное обозначение кода завершения | Значение |
|---|---|---|
| 0 | `exit_success` | Нормальное завершение |
| 1 | `exit_help` | Выдача справочной информации |
| 2 | `exit_wrong_parameter` | Ошибка в параметрах командной строки |
| 3 | `exit_not_connected` | Нет соединения с БД |
| 4 | `exit_some_query_has_2202` | В БД нет заданного объекта (соответствует коду завершения 2202 СУБД) |
| 5 | `exit_some_query_has_error` | Ошибка в SQL-запросе |
| 6 | `exit_cannot_open_the_file` | Ошибка открытия файла |
| 7 | `exit_some_resource_locked` | Требуемая запись или таблица заблокированы |
| 8 | `exit_no_memory` | Недостаточно памяти |
| 9 | `exit_proc_raised_exception` | Хранимая процедура завершилась исключением |