

**МОБИЛЬНАЯ  
РЕЛЯЦИОННАЯ  
СУБД**

**ЛИНТЕР®**

Linter Standard  
Linter Bastion  
Linter RealTime  
Linter Multiversion

**Полнотекстовый поиск в базе  
данных**

**НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ**

---

 **РЕЛЭКС®**

## **Товарные знаки**

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

## **Интеллектуальная собственность**

Правообладателем продуктов ЛИНТЕР®, НЕВОД®, LAV™, ЛАКУНА является компания РЕЛЭКС (1990–2011). Все права защищены. Данный документ является собственностью компании РЕЛЭКС. Ни одна часть данного документа не может быть воспроизведена, передана, преобразована, сохранена в системе поиска информации, переведена на другой язык или компьютерный язык в какой-либо форме, какими-либо средствами, электронными, механическими, магнитными, оптическими, химическими, ручными или иными, без предварительного разрешения компании РЕЛЭКС.

## **О документе**

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

## **Адрес**

394006, г. Воронеж, ул. 20-летия Октября, 119.  
Тел./факс: (473) 2-711-711, 2-778-333.  
e-mail: market@relex.ru.

## **Адрес для корреспонденции**

394000, г. Воронеж, а/я 137.

## **Техническая поддержка**

Отдел поддержки и сопровождения программных продуктов:

телефон: (473) 2-711-711 с 9:00 до 18:00 мск.  
e-mail: support@relex.ru, market@relex.ru.

С целью повышения качества разрабатываемых программных средств и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам на Internet–странице [рекламация](#).

# Оглавление

<b>Предисловие</b> .....	<b>2</b>
Назначение документа.....	2
Для кого предназначен документ.....	2
Необходимые предварительные знания.....	2
Принятые обозначения и соглашения.....	2
Дополнительные документы.....	3
<b>Основные понятия</b> .....	<b>4</b>
<b>Условия применения</b> .....	<b>5</b>
<b>Фильтры</b> .....	<b>6</b>
Управление фильтрами.....	7
Создание внутреннего фильтра.....	7
Удаление фильтра.....	9
Изменение внешнего фильтра.....	9
Фильтры столбца.....	10
Фильтры файлов.....	11
Правила выделения слов.....	13
<b>Индексирование</b> .....	<b>14</b>
Создание фразового индекса.....	14
Правила выбора фильтра.....	15
Модификация фразового индекса.....	16
Обновление фразового индекса.....	17
Удаление фразового индекса.....	17
<b>Управление</b> .....	<b>18</b>
Поиск.....	18
Шаблон фразового поиска.....	20
Примеры полнотекстового поиска.....	23
Функции.....	25
Местоположение искомых элементов текста.....	25
Выборка текста.....	27
Время создания фразового индекса.....	29
Сформировать значение типа EXTFILE.....	29
Получить значение столбца типа файла.....	30
Получить номер фильтра.....	30
Дата обновления файла.....	31
Размер внешнего файла.....	32
Каталог внешних файлов.....	32
<b>Приложение. Расширения SQL</b> .....	<b>33</b>
<b>Указатель операторов и функций</b> .....	<b>34</b>

# Предисловие

## Назначение документа

Документ описывает возможности полнотекстового (фразового) поиска СУБД ЛИНТЕР.

Документ может использоваться для работы с любой версией СУБД ЛИНТЕР. Особенности конкретных версий оговариваются по тексту.

## Для кого предназначен документ



Документ предназначен для программистов, разрабатывающих приложения на основе СУБД ЛИНТЕР.

## Необходимые предварительные знания

Для работы с полнотекстовым поиском необходимо знать основы реляционных баз данных и языка баз данных SQL.

## Принятые обозначения и соглашения

Обозначение	Пример	Значение
Курсив	<i>Растровым</i> называется изображение...	Новый термин в тексте
Полужирный шрифт	В этом случае необходимо переносить <b>все</b> физические файлы.	Выделение в тексте
Подчеркнутый шрифт	Подробную информацию о работе программы можно получить на сайте <a href="http://www.dmk.ru">www.dmk.ru</a> .	Адреса страниц Internet
Текст, разделенный знаком ⇒	Выполните команду <b>View ⇒ Properties</b> (Вид ⇒ Свойства).	Последовательность выполнения команд
Текст, заключенный в <>, со знаком + между ними	<Ctrl>+<C>	В <> заключаются клавиши клавиатуры, знак + означает сочетание клавиш
Крупный моноширинный текст	SQL> _q	Текст командной строки
Мелкий моноширинный текст	Page Time Count	Текст программы
Заглавные буквы	BROWSE	Названия команд, слова, зарезервированные в SQL, ключевые слова
Курсив в <>	<return statement>	Определяемый элемент

Обозначение	Пример	Значение
Символ ::=		синтаксической конструкции Равенство по определению. Слева от знака стоит определяемое понятие, справа – собственно определение понятия
Квадратные скобки [ ]	<code>DBSTORE [-d -n -o -p -r -t -u]</code>	Необязательные элементы конструкции. В данном примере ключи не являются обязательными элементами команды
Вертикальная черта	<code>&lt;return value&gt; ::=</code> <code>&lt;value expression&gt;   NULL</code>	Указывает на то, что все предшествующие ей элементы списка являются необязательными и могут быть заменены любым другим элементом списка после этой черты
Фигурные скобки { }	<code>CODEPAGE</code> <code>{ 866</code> <code>  1251</code> <code>  KOI8}</code>	Указывают на то, что все, находящееся внутри них, является единым целым
Многоточие «...»	Характеристики столбца <code>MAKE CHAR(20)</code> <code>MODEL CHAR(20)</code> <code>...</code> <code>SQL&gt;</code>	Означает, что предшествующая часть может быть повторена любое количество раз либо продолжена по аналогии
Многоточие, внутри которого находится запятая «,...»		Указывает на то, что предшествующая часть оператора, состоящая из нескольких элементов, разделенных запятыми, может иметь произвольное число повторений
Текст со знаком  на сером фоне	 Если конфигурация страницы-шаблона не учитывала свойств, команда будет выполнена некорректно.	Примечание

## Дополнительные документы

- СУБД ЛИНТЕР. Архитектура СУБД.
- СУБД ЛИНТЕР. Справочник по SQL.
- СУБД ЛИНТЕР. Справочник кодов завершения.

# Основные понятия

Понятие «*полнотекстовый*» (или *фразовый*) поиск подразумевает поиск по полному тексту или по всем текстовым полям документа (базы данных). Любой текстовый документ, как правило, имеет внутреннюю структуру - деление на параграфы, отступ для заголовка, для подписи, таблицы. Текстовые редакторы позволяют делать эту структуру достаточно сложной - выделять текст шрифтами и вариантами их начертания, делать списки, выравнивание и т.д. и т.п. Кроме того, различные редакторы имеют определенные форматы хранения данных (.doc, .html, .rtf, .tex и др.). Некоторые документы (например, в формате .html), помимо средств визуального оформления информации, имеют разметку внутренней структуры - заголовок, тело документа, ключевые слова. Поэтому в задачу полнотекстового поиска входит понимание внутренней структуры и «расшифровка» разных форматов документов с помощью специальных средств - конверторов или фильтров.

СУБД ЛИНТЕР со средствами фразового поиска рекомендуется использовать в проектах, где основными определяющими факторами являются скорость поиска и извлечения текста по фразе в больших хранилищах информации (например, интернет-сервер). Средства фразового поиска дают возможность упростить схему хранения данных в приложении и избежать создания некоторых дополнительных таблиц.

Система фразового поиска обеспечивает:

- варианты поиска слов: по началу, окончанию, части слова, целому слову, поиск с использованием символов шаблона;
- поиск по словам, набранным с ошибками (нечеткий поиск). Поддерживаются три основных типа ошибок (пропуск, вставка, замена буквы);
- поиск с учетом и без учета регистра букв;
- поиск близкорасположенных слов и фраз с известным порядком слов;
- поиск по названию и значению атрибута в файлах с гипертекстовой разметкой;
- автоматическое определение кодировки русскоязычного текста.

В СУБД ЛИНТЕР версии 6.0 и выше дополнительно обеспечивается:

- поддержка многобайтных кодировок и иероглифических символов;
- хранение информации в кодировке UNICODE.

# Условия применения

Для применения фразового поиска необходимо выполнение следующих условий:

- 1) наличие в БД системных таблиц `$$$FILTER` и `$$$EXTENSION` (см. ниже) для поддержки системы фразового поиска. Для создания и загрузки указанных таблиц необходимо выполнить файлы `search.sql` и `default.sql`, поставляемые в составе дистрибутива СУБД ЛИНТЕР в подкаталоге `dict` (только для версии СУБД ЛИНТЕР с поддержкой фразового поиска);
- 2) наличие внешних фильтров (см. ниже) для работы с нестандартными форматами документов;
- 3) некоторые файлы формата PDF могут потребовать дополнительные файлы перекодировки. Стандартные файлы перекодировок могут быть получены с сайта <http://www.adobe.com>;
- 4) наличие в таблице `$$$CHARSET` записей с таблицами перекодировки для используемых кодовых страниц (только для СУБД ЛИНТЕР версии 6.0 и выше).

 В составе СУБД ЛИНТЕР версии 6.0 и выше присутствуют таблицы перекодировки для кодовых страниц 866, KOI8-R, 1251, 437, 850, 1252, ISO 8859-1, ISO 8859-2, ISO 8859-3, ISO 8859-4, ISO 8859-5, ISO 8859-6, ISO 8859-7, ISO 8859-8, ISO 8859-9, ISO 8859-10, ISO 8859-13, ISO 8859-14, ISO 8859-15.

# Фильтры

Быстрый поиск документов в системе фразового поиска возможен только с помощью индексирования входящих в документ слов. Перед индексацией тексты должны быть преобразованы к некоторому стандартному представлению. Например, с точки зрения смыслового значения (т.е. поискового запроса) приведенные визуальные фрагменты текста являются идентичными:

СУБД ЛИНТЕР

СУБД ЛИНТЕР

СУБД ЛИНТЕР

Так как в качестве документов могут использоваться документы любых форматов (ТХТ, DOC, RTF, PDF, HTML и т.д.), то обрабатывать все различные форматы в системе фразового поиска нецелесообразно. Логичным решением этой проблемы является система *конверторов* (или *фильтров*), которые занимаются проблемой извлечения собственно текста из данных, сохраненных в некотором, отличном от текстового, формате. На вход фильтру подается поток данных, на выходе получаем чистый текст. Например, используя конвертор HTML, можно получить текст из HTML-документа.

*Фильтром* в СУБД ЛИНТЕР называется динамическая библиотека, которая извлекает из документа его содержимое (в виде потока текста) и свойства (автор, размер, дата создания и т.д.). Нужный фильтр устанавливается при настройке СУБД или непосредственно специальными операторами языка SQL.

СУБД ЛИНТЕР имеет набор встроенных фильтров для некоторых наиболее распространенных форматов.

Фильтры являются подключаемыми модулями (библиотеками), любой пользователь БД может написать собственный фильтр для нужного типа файлов и включить его в СУБД (см. стр. 8).

В таблице 1 представлены фильтры, которые могут быть встроены в ядро СУБД ЛИНТЕР (так называемые внутренние).

**Таблица 1. Внутренние фильтры СУБД ЛИНТЕР**


Имя фильтра	Формат входного файла
ASCTEXT2TEXT	Текст в кодировке ASCII
ANSI2TEXT	Текст в кодировке ANSI
KOI8R2TEXT	Текст в кодировке KOI8-R
UNITEXT2TEXT	Текст в кодировке UNICODE
UTF82TEXT	Текст в кодировке UTF8
RUSTEXT2TEXT	Текст в одной из кодировок ASCII, ANSI или KOI8-R
ASCXML2TEXT	HTML, XML в коде ASCII
UNIXML2TEXT	HTML, XML в коде UNICODE
DOCRTF2TEXT	RTF, PDF, DOC, XLS, PPT, DOCX, XLSX, PPTX, ODT, ODS, ODP, SXW, SXC, SXI <sup>1)</sup>

Имя фильтра	Формат входного файла
NOTEXT2TEXT	Любой. Документ исключается из процесса индексирования.

1)

docx	– формат	MS Word 2007
xlsx	– формат	MS Excel 2007
pptx	– формат	MS PowerPoint 2007
odt	– формат	OpenOffice.org Writer 3.x
ods	– формат	OpenOffice.org Calc 3.x
odp	– формат	OpenOffice.org Impress 3.x
sxw	– формат	OpenOffice.org Writer 1.x
sxc	– формат	OpenOffice.org Calc 1.x
sxi	– формат	OpenOffice.org Impress 1.x

Фильтры до версии 6.0 возвращают текст в кодировке CP866, начиная с версии 6.0 – в UNICODE.

 Информация о всех доступных фильтрах (как внутренних, так и подключенных пользователями) хранится в системной таблице \$\$\$FILTER, имеющей следующую структуру:

```
CREATE TABLE $$$FILTER
(
  $$$ID      INTEGER,      /* Номер фильтра */
  $$$NAME    CHAR(LINTER_NAME_LENGTH), /* Имя фильтра */
  $$$KEY     INTEGER,      /* Контрольная сумма внешнего фильтра */
  $$$MODULE  CHAR(128),    /* Имя библиотеки - для внешних фильтров */
  $$$DESC    CHAR(256)    /* Комментарий */
);
```

Как сказано выше, данная таблица создается путем запуска файла search.sql.

## Управление фильтрами

### Создание внутреннего фильтра

#### Функция

Подключение внутреннего фильтра.

Хотя СУБД ЛИНТЕР и имеет набор встроенных внутренних фильтров, не все они обязательно должны использоваться. Это потенциально возможный набор. Чтобы фильтр стал доступным для применения, информация о нем должна быть помещена в системную таблицу \$\$\$FILTER.

#### Спецификация

<создание внутреннего фильтра> ::=

**CREATE INTERNAL FILTER** <имя фильтра> [ **DESCRIPTION** <описание> ];

<имя фильтра> ::= идентификатор

## Фильтры

---

<описание> ::= строковый литерал

### Синтаксические правила

- 1) <имя фильтра> должно совпадать с одним из predefined имен фильтров (см. таблицу 1);
- 2) дублирование фильтров в таблице \$\$\$FILTER не допускается, т.е. повторное добавление внутреннего фильтра игнорируется. Если необходимо модифицировать фильтр, его сначала надо удалить, затем создать заново;
- 3) <описание> – строковый литерал длиной не более 256 знаков. Его значение помещается в поле \$\$\$DESC таблицы \$\$\$FILTER.

### Общие правила

Созданный фильтр становится доступным для использования.

### Примеры

```
CREATE INTERNAL FILTER "ASCTEXT2TEXT";
```

```
CREATE INTERNAL FILTER ASCXML2TEXT DESCRIPTION 'версия 1.0';
```

## Создание внешнего фильтра

### Функция

Подключение к СУБД ЛИНТЕР фильтра, разработанного пользователем.

### Спецификация

<создание внешнего фильтра> ::=

```
CREATE [ EXTERNAL ] FILTER <имя фильтра> = <номер>
```

```
MODULE <спецификация файла>
```

```
[ DESCRIPTION <описание> ];
```

<имя фильтра> ::= идентификатор

<номер> ::= целое положительное число

<спецификация файла> ::= строковый литерал

<описание> ::= строковый литерал

### Синтаксические правила

- 1) <имя фильтра> не должно совпадать с одним из predefined имен внутренних фильтров (см. таблицу 1);
- 2) <номер> должен быть уникальным среди номеров таблицы \$\$\$FILTER;
- 3) дублирование фильтров в таблице \$\$\$FILTER не допускается, т.е. добавление внешнего фильтра с тем же именем (хотя и с другим <номером>) игнорируется;
- 4) <описание> - строковый литерал длиной не более 256 знаков. Его значение помещается в столбец Комментарий таблицы \$\$\$FILTER;
- 5) <спецификация файла> - имя файла (библиотеки) фильтра.

### Общие правила

Созданный фильтр становится доступным для использования.

### Пример

```
CREATE FILTER "Аннотация" = 17
```

```
MODULE 'f:\frase\filter\annotation.dll'  
DESCRIPTION 'для аннотаций книг';
```

## Удаление фильтра

### Функция

Удаление любого (внутреннего или внешнего) ранее установленного фильтра.

### Спецификация

<удаление фильтра>::=

**DROP FILTER** <имя фильтра>

<имя фильтра>::= идентификатор

### Синтаксические правила

<Имя фильтра> должно содержаться в таблице \$\$\$FILTER.

### Общие правила

- 1) указанный фильтр удаляется из системной таблицы \$\$\$FILTER и становится недоступным для использования;
- 2) информация о внутреннем фильтре сохраняется в БД;
- 3) файл внешнего фильтра физически не удаляется.

### Пример

```
DROP FILTER "ASCTEXT2TEXT";
```

## Изменение внешнего фильтра

### Функция

Замена файла (библиотеки) для установленного внешнего фильтра.

### Спецификация

<модификация фильтра>::=

**ALTER FILTER** <имя фильтра> **MODULE** <спецификация файла>

<имя фильтра>::= идентификатор

<спецификация файла>::=строковый литерал

### Синтаксические правила

<Имя фильтра> должно содержаться в таблице \$\$\$FILTER.

### Общие правила

- 1) для указанного фильтра прежнее имя файла заменяется новым (в системной таблице \$\$\$FILTER) и становится доступным для использования;
- 2) прежний файл фильтра физически не удаляется.

### Пример

```
ALTER FILTER "Аннотация"  
MODULE 'f:\frase\filter\annotation01.dll';
```

### Фильтры столбца

#### Назначение фильтра для столбца

##### Функция

Указание по применению фильтра для столбцов таблицы.

##### Спецификация

В конструкцию CREATE TABLE языка SQL СУБД ЛИНТЕР в спецификацию свойств столбца добавлен элемент:

```
<использование фильтра> ::=  
DEFAULT FILTER <имя фильтра>
```

```
CREATE TABLE <имя таблицы>  
(... <имя столбца> <тип> DEFAULT FILTER <имя фильтра> ...);
```

##### Синтаксические правила

<Имя фильтра> должно содержаться в таблице \$\$\$FILTER.

##### Общие правила

Установленный фильтр используется, если его действие не перекрыто другими указаниями по применению фильтра.


##### Пример

```
CREATE TABLE TEST_BLOB  
(  
    Id      INTEGER,  
    Name    CHAR(18),  
    Document BLOB DEFAULT FILTER ASCXML2TEXT  
);
```

 При применении фильтров могут возникнуть проблемы, связанные с тем, что:

1. файлы могут иметь разные кодировки.  
В СУБД ЛИНТЕР версии 5.9 и ниже предполагается, что все данные хранятся в кодировке CP866. Поэтому, если документ в другой кодировке (например, CP1251) будет записан в текстовое поле БД, то полнотекстовый поиск будет безрезультатным;
2. если для просмотра PDF-документа требуется ввод непустого пароля, то содержимое документа не индексируется, и на консоль выдается предупредительное сообщение;
3. не все PDF-документы предоставляют информацию, необходимую для извлечения текста и последующей индексации. Содержимое таких документов может не быть проиндексировано;
4. PDF-документы, содержащие иероглифы, могут потребовать дополнительных файлов с таблицами перекодировки. В случае отсутствия требуемых файлов иероглифический текст будет проигнорирован (см п. 3 на стр. 4);
5. файлы форматов RTF, XLS для MS Excel 5.0/95 и Excel 97-2002 и DOC для MS Word 6.0/95 могут потребовать наличия в системной таблице \$\$\$CHARSET записей с таблицами перекодировки. Текст в кодировке, для которой отсутствует запись с

таблицей перекодировки, проиндексирован не будет (справедливо для ЛИНТЕР версии 6.0 и выше). В случае ЛИНТЕР 5.9 будет проигнорирован текст, представленный в кодировках, отличных от 866, 1251 и KOI8-R.

 В настоящее время СУБД ЛИНТЕР реально не использует внешние фильтры.

## Модификация фильтра столбца

### Функция

Установка отсутствующего или замена установленного фильтра для столбца.

### Спецификация

```
<модификация фильтра> ::=
ALTER TABLE <имя таблицы>
ALTER COLUMN <имя столбца>
SET DEFAULT FILTER <имя фильтра>;
```

### Синтаксические правила

1) <имя фильтра> должно содержаться в таблице \$\$\$FILTER.

### Общие правила

Столбцу назначается указанный фильтр по умолчанию.

## Удаление фильтра столбца

### Функция

Отмена установленного для столбца фильтра.

### Спецификация

```
<отмена фильтра> ::=
ALTER TABLE <имя таблицы> ALTER COLUMN <имя столбца>
DROP DEFAULT FILTER;
```

### Синтаксические правила

<Имя столбца> должно ссылаться на столбец, для которого установлен фильтр.

## Фильтры файлов

Для столбцов типа EXTFILE выбор фильтра может быть сделан самой СУБД ЛИНТЕР автоматически - по расширению файла. Для этого используется системная таблица \$\$\$EXTENSION, имеющая следующую структуру:

```
CREATE TABLE $$$EXTENSION
(
    $$$EXT CHAR(LINTER_NAME_LENGTH), /* Расширение - CASE-SENSITIVE */
    $$$FILTER INTEGER /* ID фильтра по умолчанию */
);
```

Таблица создается с помощью файла search.sql.

### Установка фильтра для расширения файла

#### Функция

Установка фильтра по умолчанию для заданного расширения файла.

#### Спецификация

<установка фильтра для файла> ::=

**SET DEFAULT FILTER** <имя фильтра> **FOR** <расширение>;

<расширение> ::= строковый литерал

#### Синтаксические правила


- 1) <имя фильтра> должно ссылаться на один из установленных в БД фильтров (внутренних или внешних), т.е. тех, которые уже включены в таблицу \$\$\$FILTER;
- 2) <расширение> – строковый литерал, задающий расширение файла.

#### Общие правила

- 1) установленный фильтр автоматически применяется для столбцов типа EXTFILE в том случае, если явно не указано применение другого фильтра;
- 2) расширения файлов, для которых при запуске файла default.sql по умолчанию устанавливаются фильтры, приведены в таблице 2.

**Таблица 2. Фильтры по умолчанию для расширений файлов**

Расширение файла	Имя фильтра
TXT, txt	rustext2text
DOC, doc, RTF, rtf, PDF, pdf, DOCX, docx, XLSX, xlsx, PPT, ppt, PPTX, pptx, ODT, odt, ODS, ods, ODP, odp, SXW, sxw, SXC, sxc, SXI, sxi	docrtf2text
XML, xml, HTM, htm, HTML, html, PHTML, phtml, SHTML, shtml	ascxml2text

 Выбор фильтра по умолчанию для расширения файла осуществляется с учетом регистра. Таким образом, на основании таблицы 2 файлу myfile.Doc не будет сопоставлен фильтр по умолчанию.

### Отмена фильтра для расширения файла

#### Функция

Отмена ранее установленного фильтра по умолчанию для заданного расширения файла.

#### Спецификация

<отмена фильтра для файла> ::=

**CANCEL DEFAULT FILTER FOR** <расширение>;

<расширение> ::= строковый литерал

#### Синтаксические правила

Указанное <расширение> должно содержаться в таблице \$\$\$EXTENSION.

## Общие правила

Фильтр перестает быть фильтром, установленным по умолчанию для файлов с данным расширением.

## Правила выделения слов

Слово текста документа составляют:

- символы a-z, A-Z, А-Я, а-п, р-я, Ё, ё, 0-9 и символ ‘\_’ (ЛИНТЕР 5.9);
- буквенно-цифровые символы (согласно стандарту UNICODE) и символ ‘\_’ (ЛИНТЕР 6.0 и выше);
- символы, значимые в середине слова: ‘@’, ‘-’, ‘/’, ‘\’, ‘”’. Данные символы являются частью слова, если окружены указанными выше символами (буквенно-цифровыми и знаком подчеркивания). В частности, эти символы не могут быть первым или последним символом слова.

Имена и значения атрибутов подчиняются правилам, устанавливаемым спецификацией XML.

Фильтр xml2text использует следующие правила:

- имя атрибута начинается с символов a-z, A-Z, А-Я, а-п, р-я, Ё, ё, ‘\_’, ‘.’;
- имя атрибута продолжается символами a-z, A-Z, А-Я, а-п, р-я, Ё, ё, 0-9, ‘\_’, ‘.’, ‘-’;
- кодировка по умолчанию CP866;
- если документ содержит атрибут CONTENT, то кодировка документа определяется значением подстроки charset=... внутри значения атрибута CONTENT.

Фильтр unixml2text использует следующие правила:

- имя атрибута начинается с буквенно-цифровых символов (согласно стандарту UNICODE) или символов ‘\_’, ‘.’;
- имя атрибута продолжается буквенно-цифровыми символами, а также ‘\_’, ‘.’, ‘-’.

Значением атрибута является заключенная в одинарные или двойные кавычки строка с учетом следующих замен:

Последовательность	Символ
&quot;	“”
&amp;	‘&’
&lt;	‘<’
&gt;	‘>’
&nbsp;	‘ ‘


Максимальная длина слова, имени и значения атрибута составляет 64 буквы. Длинные слова усекаются до 64 букв.

# Индексирование

Все поисковые системы производят поиск с помощью заранее построенного *индекса*. Поэтому документы, участвующие в поиске, должны быть предварительно проиндексированы. Системы, не производящие индексацию (производящие поиск с помощью просмотра текстов), просто не в состоянии провести поиск в реальном времени по уже нескольким десяткам мегабайт текстовой информации. Для индексирования могут использоваться следующие методы:

- 1) индексирование ключевыми словами - в индекс входит каждое слово, встречающееся в словосочетании, за исключением слов, отнесенных к стоп-словам. Стоп-слова - это, как правило, высокочастотные служебные слова - предлоги, союзы. Из словосочетания «Война и мир» в индекс войдут два слова «война» и «мир». При поиске порядок введенных слов не имеет значения - каждое слово ищется отдельно, затем результаты поиска пересекаются;
- 2) фразовое индексирование: содержание поля или подполя заносится в индекс целиком. Из словосочетания «война и мир» в индексе получится один вход – «война и мир». При поиске нужно вводить слова в правильном порядке;
- 3) индексирование ключами - усечение каждого слова до определенного числа букв;
- 4) пермутационное индексирование – словосочетание «переворачивается», выводя на первое место каждое из слов, входящих в него.

В СУБД ЛИНТЕР реализованы сочетания первого и второго способов.

 В настоящее время список стоп-слов пуст.

## Создание фразового индекса

### Функция

Создание фразового индекса.

### Спецификация

<создание фразового индекса> ::=

```
CREATE [OR REPLACE] PHRASE [IMMEDIATE|DEFERRED] [XML]  
INDEX <имя столбца> ON <имя таблицы>;
```

### Синтаксические правила

- 1) <имя столбца> должно принадлежать столбцу без фразового индекса;
- 2) допустимый тип <имя столбца>: CHAR, VARCHAR, NCHAR, NCHAR VARYING, BLOB, EXTFILE;
- 3) для столбцов типа BLOB разрешен только модификатор DEFERRED;
- 4) флаг XML задает построение атрибутивного индекса;
- 5) значения по умолчанию флагов и модификаторов приведены в таблице 3.


**Таблица 3. Значения по умолчанию флагов и модификаторов**

Тип столбца	Флаг XML	Модификатор
CHAR	Сброшен	IMMEDIATE
VARCHAR	Сброшен	IMMEDIATE

Тип столбца	Флаг XML	Модификатор
NCHAR	Сброшен	IMMEDIATE
NCHAR VARYING	Сброшен	IMMEDIATE
BLOB	Сброшен	DEFERRED
EXTFILE	Сброшен	DEFERRED

## Общие правила

- 1) создается фразовый индекс, в который включается содержимое столбца <имя столбца>;
- 2) IMMEDIATE означает немедленное обновление индекса при обновлении поля, DEFERRED - обновление только по команде REBUILD PHRASE INDEX.
- 3) создание фразового индекса возможно только в том случае, если при запуске ядра СУБД ЛИНТЕР подсистеме полнотекстового поиска выделен буфер памяти. Размер буфера задается в страницах размером 4К с помощью ключа /pool командной строки запуска СУБД ЛИНТЕР. Минимальное значение 4. По умолчанию используется ключ /POOL=0, т.е. создание (перестройка) индекса невозможна.

 Индексированные документы имеют метки секретности. Все функции, связанные с полнотекстовым индексом или основанные на извлечении текста, а также функции для работы с полями типа EXTFILE, обрабатывают метки доступа.

## Правила выбора фильтра

Следующие правила определяют выбор фильтра, используемого при создании индекса (фильтры перечислены в порядке предпочтения):

- 1) для столбца типа BLOB:
  - фильтр, для которого значение поля \$\$\$ID в таблице \$\$\$FILTER равно номеру типа данных BLOB (если отличен от 0);
  - фильтр, заданный по умолчанию для столбца;
  - DOCRTF2TEXT, ASCXML2TEXT или UNIXML2TEXT, если найдена сигнатура файла форматов doc, xls, ppt, rtf, pdf, ps, xml;
  - UNITEXT2TEXT в случае наличия BOM (Byte Order Mark) в начале документа (последовательность байт ffh feh или feh ffh);
  - ASCTEXT2TEXT.
- 2) для столбца типа EXTFILE:
  - фильтр, заданный вторым (необязательным) параметром функции EXTFILE;
  - фильтр, заданный по умолчанию для столбца;
  - фильтр, заданный по умолчанию для расширения файла (с учётом регистра);
  - DOCRTF2TEXT, ASCXML2TEXT или UNIXML2TEXT, если найдена сигнатура файла форматов doc, xls, ppt, rtf, pdf, ps, xml;
  - UNITEXT2TEXT в случае наличия BOM (Byte Order Mark) в начале документа (последовательность байт ffh feh или feh ffh);
  - ASCTEXT2TEXT;

3) для столбца типа NCHAR, NCHAR VARYING:

- фильтр, заданный по умолчанию для столбца;
- UNITEXT2TEXT;

4) для столбца типа CHAR, VARCHAR:

- фильтр, заданный по умолчанию для столбца;
- фильтр неформатированного текста в кодировке столбца (для версии СУБД ЛИНТЕР меньше 6.0 всегда ASCTEXT2TEXT);

Фильтры ASCXML2TEXT и UNIXML2TEXT используются для файлов в форматах XML, XHTML, HTML. Тем не менее, файлы этих форматов обрабатываются существенно различным образом: в случае XML извлекаются пары <имя\_атрибута>=<значение\_атрибута>, которые могут быть включены в индекс, если задан флаг XML в конструкции CREATE PHRASE INDEX.

В случае HTML и XHTML атрибуты элементов игнорируются. Исключение составляют элементы META, содержащие метаинформацию, касающуюся документа. Элемент META содержит 2 обязательных атрибута:

NAME=Имя1 (или http-equiv) и CONTENT=Значение1.

В случае, если Имя1 равно "Author", "Keywords" или "Description", в атрибутный индекс добавляются пары <Имя1>=<Значение1>.

Если <Имя1>="Content-Type", то <Значение1> сканируется на предмет наличия информации об используемой документом кодировке, заданной в виде "charset=имя\_кодировки".

Формат файла определяется по наличию сигнатуры "<?xml" в начале документа (которой, возможно, предшествует BOM). В случае её отсутствия считается, что документ в формате HTML. В противном случае тип документа, согласно спецификации форматов XML и XHTML, определяется однозначно.

## Модификация фразового индекса

### Функция

Модификация фразового индекса.

### Спецификация

<модификация фразового индекса>::=

```
ALTER PHRASE INDEX <имя столбца> ON <имя таблицы>  
[ IMMEDIATE | DEFERRED ];
```

### Общие правила

- 1) <имя столбца> должно принадлежать столбцу с фразовым индексом;
- 2) IMMEDIATE означает немедленное обновление индекса при обновлении поля, DEFERRED - обновление только по команде REBUILD PHRASE INDEX;
- 3) для столбцов типов CHAR, VARCHAR, NCHAR, NCHAR VARYING по умолчанию действует модификатор IMMEDIATE;
- 4) для столбцов типа EXTFILE по умолчанию действует модификатор DEFERRED;

- 5) для столбцов типа BLOB разрешен только модификатор DEFERRED;
- 6) если предыдущий режим был DEFERRED, то установка режима IMMEDIATE вызовет немедленное обновление индекса.

## Обновление фразового индекса

### Функция

Обновление фразового индекса.

### Спецификация

<обновление фразового индекса> ::=

**REBUILD PHRASE INDEX** <имя столбца> **ON** <имя таблицы>;

### Синтаксические правила

<Имя столбца> должно принадлежать столбцу с фразовым индексом

### Общие правила

- 1) обновляет фразовый индекс, индексируя добавленные либо измененные с момента создания или последнего обновления индекса записи и исключая информацию об удаленных записях;
- 2) обновление фразового индекса выполняется автоматически для индексов с модификатором IMMEDIATE;
- 3) для перестройки индекса необходимо иметь категорию доступа CONNECT.

## Удаление фразового индекса

### Функция

Удаление существующего фразового индекса.

### Спецификация

<удаление фразового индекса> ::=

**DROP PHRASE INDEX** <имя столбца> **ON** <имя таблицы>;

### Синтаксические правила

<Имя столбца> должно принадлежать столбцу с фразовым индексом

# Управление

## Поиск

Для выполнения фразового поиска используется предикат фразового поиска (расширение конструкции поискового оператора SELECT языка БД SQL):

<предикат фразового поиска> ::=

<имя столбца> [(] [NOT ] CONTAINS[)]

[<модификатор> <модификатор> ... ]

<шаблон фразового поиска>

<модификатор> ::=

**SENSITIVE | PARTIALLY | AT\_BEGIN | AT\_END | FUZZY**

<шаблон фразового поиска> ::= строковый литерал

### Синтаксические правила

- 1) <имя столбца> должно принадлежать столбцу, по которому построен фразовый индекс;
- 2) модификаторы (по умолчанию отсутствуют) перечисляются в любом порядке и разделяются пробелами.

### Общие правила

В таблице 5 представлены значения модификаторов и их назначение.

**Таблица 5. Модификаторы фразового поиска**

Модификатор	Назначение	Представление в шаблоне	Местоположение в шаблоне	Пример
SENSITIVE	Задаёт чувствительный к регистру поиск	#	Перед словом	#Word
PARTIALLY	Задаёт поиск документов, в которых обозначенный шаблон поиска может встречаться в любом месте слова	*	В начале и конце	*WORD*
AT_BEGIN	Задаёт поиск документов, в которых обозначенный шаблон поиска может встречаться только в начале слов	*	В конце	WORD*
AT_END	Задаёт поиск документов, в которых обозначенный шаблон поиска может встречаться только в конце слов	*	В начале	*WORD
FUZZY	Задаёт нечеткий поиск	%	Перед словом	%WORD

Под словом понимается собственно слово, имя и значение атрибута.

При задании модификатора соответствующий специальный символ (см. столбец Представление в шаблоне) применяется ко всем словам шаблона фразового поиска.

Если модификатор должен быть применен не ко всем словам искомого документа, следует использовать специальный символ для конкретных слов поискового запроса.

## Примеры

а) поиск отдельных слов с учетом регистра букв

Конструкция

```
select id_doc from "ph" where text_doc contains '#БД #ЛИНТЕР';
```

эквивалентна

```
select id_doc from "ph" where text_doc contains sensitive 'БД  
ЛИНТЕР';
```

б) поиск без учета регистра (выдаются все документы с фразами «бд», «БД» «ЛИНТЕР», «Линтер» и т. п.)

```
select id_doc from "ph" where text_doc contains 'БД линтер';
```

Режим нечеткого поиска позволяет искать лексикографически близкие фразы, отличающиеся заменами, пропусками и вставками символов. Нечеткий поиск целесообразно применять при поиске слов с опечатками, а также в тех случаях, когда возникают сомнения в правильном написании — фамилии, названия организации и т. п.

в) следующий запрос выдаст, в частности, документы с ошибочной фразой «специальная»:

```
select id_doc from "ph" where text_doc contains fuzzy  
'специальная';
```

Ниже приведены примеры фраз, которые могут быть найдены и пропущены при фразовом поиске и степень их близости, подсчитанная с помощью SQL-функций `soundex()` и `difference()`.

Исходный документ	Найденный документ	<code>soundex()</code>	<code>difference()</code>
Специальная	✓	C167	0
Специальная	✓	C167	0
Спициальная	✓	C167	0
Специальная	✓	C167	0
Специальная		S167	1
Тпециальная		T167	1
Специа		C160	1
Специальная	✓	C167	0

## Шаблон фразового поиска

### Функция

Определение шаблона фразового поиска.

### Спецификация

<шаблон фразового поиска> ::= <поисковое выражение>

<поисковое выражение> ::=

<слово>

| <атрибут> = <значение>

| <фраза>

| <не равно> <поисковое выражение>

| <поисковое выражение> <или> <поисковое выражение>

| <поисковое выражение> [<и>] <поисковое выражение>

| (<поисковое выражение>)

<не равно> ::= !

<или> ::= |

<и> ::= &

<атрибут> ::= <слово>

<значение> ::= <слово> | ([ '#' | '%' | '\*' ] "'" <строка без символа "'> "' [ '\*' ])

| ([ '#' | '%' | '\*' ] "'" <строка без символа "'> "' [ '\*' ])

<фраза> ::= "'" (<слово> | <группа>) [[<расстояние>] (<слово> | <группа>) ...] "'"

<группа> ::= '(' (<слово> [<слово> ...] )', всего до 6 слов

<расстояние> ::= '|' (([ '+' | '-' ] <смещение>) | (<ниж\_граница> <верх\_граница>)) '|'


<смещение> ::= целое беззнаковое число в диапазоне 1..10

<ниж\_граница> ::= целое число в диапазоне -10..10

<верх\_граница> ::= целое число в диапазоне <ниж\_граница>..10

<слово> составляют:

- символы a-z, A-Z, А-Я, а-п, р-я, Ё, ё, 0-9 и символ '\_' (СУБД ЛИНТЕР 5.9);
- буквенно-цифровые символы (согласно стандарту UNICODE) и символ '\_' (СУБД ЛИНТЕР 6.0 и выше);
- символы, значимые в середине слова: '@', '-', '/', '\', '''. Указанные символы не могут быть первым символом слова, за исключением шаблона поиска по концу или части слова;
- специальные символы: '#', '%', '\*'. Эти символы могут встречаться лишь в начале слова, за исключением символа '\*', который может также быть последним символом слова (см. таблицу 5);
- одиночный символ '\*', заменяющий любое слово.

 Некоторые допустимые (см. Правила выделения слов) имена атрибутов не могут найдены согласно описанным правилам. Для их поиска следует использовать специальный символ '\*'.

 Неиндексированные документы полагаются пустыми.

Примеры:

а) подсчет количества непустых документов:

```
select count(id_doc) from "ph" where text_doc contains '*';
```

б) подсчет количества документов, содержащих гипертекстовую разметку:

```
select count(id_doc) from "ph" where tex_doc contains '*=*';
```

## Общие правила

3) значение специальных символов '#', '%', '\*' определяет Таблица 5;

4) конструкция ! <поисковое выражение> задает поиск документов, не содержащих данное поисковое выражение.

Пример.

Найти все версии документов, где отсутствует упоминание о встроенных базах данных.

Конструкция

```
select id_doc from "ph" where text_doc contains '!(встроенн*
баз* дан*)';
```

эквивалентна

```
select id_doc from "ph" where text_doc contains '!встроенн*
!баз* ! дан*';
```

и эквивалентна

```
select id_doc from "ph" where text_doc not contains 'встроенн*
баз* дан*';
```

5) конструкция <поисковое выражение> | <поисковое выражение> задает поиск документов, содержащих либо первое, либо второе, либо оба поисковых выражения.

Пример.

Найти документы, в которых упоминается о любых встроенных системах или базах данных.

```
select id_doc from "ph"
where text_doc contains 'встроенн* | (баз* дан*)';
```

6) конструкция <поисковое выражение> [&] <поисковое выражение> задает поиск документов, содержащих одновременно оба поисковых выражения. Знак '&' может быть опущен.

Пример.

Подсчитать количество документов, в которых упоминается о базах данных типа ЛИНТЕР.

```
select count(id_doc) from "ph"
where text_doc contains 'линтер* & (баз* дан*)';
```

7) конструкция (<поисковое выражение>) определяет логический порядок разбора поискового выражения.

8) конструкция <атрибут> = <значение> задает атрибутивный поиск в документах, отвечающих спецификации XML (в том числе HTML); <атрибут> задает имя атрибута, <значение> задает значение этого атрибута.

Примеры:

а) подсчитать количество документов, которые содержат атрибут “title” со значением “Базы данных”:

```
select count(id_doc) from "ph"
where text_doc contains 'title="Базы данных"';
```

Так как значение атрибута состоит из двух слов, его следует заключить в кавычки.

б) подсчитать количество документов, которые содержат атрибут “Author”:

```
select count(id_doc) from "ph"
where text_doc contains 'Author=*';
```

в) подсчитать количество документов, которые содержат атрибут “Company” с учетом регистра и значение “РЕЛЕКС” без учета регистра и, возможно, с ошибкой в написании:

```
select count(id_doc) from "ph"
where text_doc contains '#Company=%Релекс';
```

г) подсчитать количество документов, которые содержат атрибут “Citation” с известным окончанием значения, содержащего символ “”:

```
select count(id_doc) from "ph"
where text_doc contains 'Citation=*"жить хорошо!"(с)";
```

9) <фраза> задает поиск последовательно расположенных в тексте слов;

Пример.

```
select count(id_doc) from "ph"
where text_doc contains '"СУВД ЛИНТЕР"';
```

10) <расстояние> задает «расстояние» между парой слов. Так, значение 1 (значение по умолчанию) соответствует последовательно идущим словам, значение 2 показывает, что между данными словами должно находиться некоторое одно слово и т.д. Отрицательные значения указывают на обратный порядок слов.

11) допустимы 3 способа задания расстояния:

- <ниж\_граница> и <верх\_граница> задают интервал расстояний между словами;
- <смещение> без указания знака задает симметричный интервал [-<смещение>, <смещение>];
- <смещение> с указанием знака задает точное расстояние между словами;

Примеры:

а) подсчитать количество документов, в которых встречается точная фраза “СУВД ЛИНТЕР”.

Конструкция

```
select count(id_doc) from "ph"
where text_doc contains '"СУБД ЛИНТЕР"';
```

эквивалентна

```
select count(id_doc) from "ph"
where text_doc contains '"СУБД |+1| ЛИНТЕР"';
```

и эквивалентна

```
select count(id_doc) from "ph"
where text_doc contains '"СУБД |1 1| ЛИНТЕР"';
```

б) Подсчитать количество документов, в которых рядом встречаются слова “СУБД” и “ЛИНТЕР”.

```
select count(id_doc) from "ph"
where text_doc contains '"СУБД |1| ЛИНТЕР"';
```

12) <группа> задает множество слов, из которых любое может находиться в указанной позиции;

Пример.

Подсчитать количество документов, в которых рядом встречаются слова “Релэкс” и одно из слов “ЛИНТЕР”, “ЛАКУНА”, “НЕВОД”.

```
select count(id_doc) from "ph"
where text_doc contains '"РЕЛЭКС |1| (ЛИНТЕР ЛАКУНА НЕВОД)";
```

13) Длинные слова в шаблоне поиска усекаются до 64 букв;

14) Апостроф является допустимым символом внутри слова для фразового поиска (должен удваиваться).

Пример.

```
select * from TEST_SQLCHK where WORD contains 'biologist's';
```

## Примеры полнотекстового поиска

1) Найти документы, содержащие одновременно слова «база» и «данных».

```
select id_doc from "Документы"
where text_doc contains 'база данных';
```

2) Найти документы, содержащие словосочетание «база данных».

```
select id_doc from "Документы"
where text_doc contains '"база данных"';
```

3) Найти количество документов, в которых встречаются словосочетания “база данных” или “база знаний” в различных формах, таких как “в базе данных” или “базу знаний”.

```
select id_doc from "ph"
where text_doc contains at_begin '"баз (данн знан)";
```

4) Найти документы, содержащие слово “Релэкс”, возможно, набранное с опечатками, т. е. в числе прочих слова “Релекс”(опечатка), “Рейлекс”(лишняя буква), “Релкс” (пропущена буква).

```
select id_doc from "ph"  
where text_doc contains fuzzy 'релекс'
```

5) Полнотекстовый поиск в иерархическом запросе.

```
select d.id, d.name, to_char(docdate, 'mm/dd/yyyy hh:mi'),  
       pathname,      userfilename,      d_filesize,      folder_id,  
f.iconpath  
from documents d, filetype f  
where  
  folder_id in  
    select id  
      from folders  
    start with  
      id in (55, 73) connect by prior id=parent_id  
  and filedoc contains 'boot sector'  
  and d.filetype=f.id
```

# Функции

## Местоположение искомых элементов текста

### Функция

Предоставление списка позиций (местоположения) заданных элементов текстовых данных. Под элементом текста понимается набор символов, ограниченный знаками пробела.

### Спецификация

GETTEXTPOS (<имя столбца>,<поисковое выражение>[,<тип поиска>]  
[,<начало поиска>][,<объем поиска>] )

<поисковое выражение>::=<шаблон поиска> [|<шаблон поиска>...]

<шаблон поиска>::=<LIKE-шаблон> | <CONTAINS-шаблон>

<LIKE-шаблон>::= <символьный литерал>

< CONTAINS-шаблон>::= <символьный литерал>

<тип поиска> ::= 1 | 2 | 4

<начало поиска>::= целочисленное положительное значение

<объём поиска>::= целочисленное неотрицательное значение

### Синтаксические правила

1) <Имя столбца> должно соответствовать столбцу типа BLOB, EXTFILE, CHAR, VARCHAR, NCHAR, NCHAR VARYING.

2) <LIKE-шаблон> должен соответствовать спецификации <предиката подобия> (см. документ Справочник по SQLб раздел «Предикат подобия») и может содержать стандартные специальные символы: подчеркивание «\_» представляет собой указатель на произвольный символ, процент «%» - указатель на подстроку (возможно, пустую).

```
create or replace table test(c char(100));
insert into test(c) values ('11 22 333 11 4411 55 666 1177 811 1199');
select gettextpos(c,'11%',2,1,2) from test;
|00000000002 0000000013 1 2 11 2|
```

```
select gettextpos(c,'11%|22|4%',2,1,10) from test;
|00000000006 0000000000 1 2 4 2 11 2 14 4 26 4 35 4|
```

3) Следует учитывать различие между шаблоном подобия в <предикате подобия> и в <LIKE-шаблоне> данной функции. В <предикате подобия> шаблон подобия распространяется на массив символов, в то время как в данной функции он распространяется только на элементы текста, например:

```
create or replace table tst (c varchar(500));
insert into tst(c) values ('<11>');
insert into tst(c) values ('<11 aa>');
insert into tst(c) values ('<11aa>');
```

При выполнении этого запроса находим все 3 записи:

```
select rowid from tst where c like '<11%>';
```

```
1  
2  
3
```

При выполнении нижеследующего запроса находим элементы только в 1 и 3 строках, т. к. во второй строке набор символов '<11 aa>' рассматривается функцией GetTextPos как два разных элемента текста, каждый из которых удовлетворяет шаблону подобия <11%>:

```
select gettextpos(c, '<11%>',2,1,length(c)) from tst;  
1  
3
```

4) <CONTAINS–шаблон> должен соответствовать спецификации <предиката фразового поиска> (см. документ «СУБД ЛИНТЕР. Полнотекстовый поиск»).

```
select gettextpos(c, '11*',1,1,10) from test;  
|00000000004 0000000000 1 2 11 2 26 4 35 4|
```

5) Если одновременно указывается несколько шаблонов поиска, то все они должны быть однотипными (либо только <LIKE–шаблоны>, либо <CONTAINS–шаблоны>).

6) <Тип поиска> – битовая маска, задающая атрибуты поиска:

- 1 (001) – поиск по <CONTAINS –шаблону>;
- 2 (010) – поиск по <LIKE –шаблону>;
- 4 (100) – поиск с учетом регистра (допускается объединение с двумя предыдущими атрибутами – т. е. значение 5 и 6 соответственно).

7) Если <тип поиска> не задан, по умолчанию используется значение 1 (поиск по <CONTAINS–шаблону>), и в этом случае последующие аргументы функции не должны задаваться (будут использованы их значения по умолчанию).

Конструкция

```
select gettextpos(c, '11*') from test;
```

эквивалента

```
select gettextpos(c, '11*',1,1,0) from test;
```

8) <Начало поиска> задает номер позиции (значение типа INTEGER) в данных, начиная с которой необходимо выполнять поиск элементов по шаблону. Отсчет позиций начинается с 1. Аргумент не обязательный; если не задан, по умолчанию принимается значение 1.

9) Если <начало поиска> не задано, по умолчанию используется 1 (поиск с начала), и в этом случае последующий аргумент функции не должен задаваться (будет использовано значение по умолчанию);

Конструкция

```
select gettextpos(c, '11*',1) from test;
```

эквивалента

```
select gettextpos(c, '11*',1,1,0) from test;
```

10) <Объем поиска> (значение типа INTEGER) ограничивает количество маркируемых элементов:

- 0 – маркировать все найденные элементы;

- n – выполнять прямой поиск; маркировать заданное (n) количество элементов;
- -n – выполнять обратный поиск; маркировать заданное (n) количество элементов.

Аргумент не обязательный; если не задан, по умолчанию принимается значение 0.

11) Если аргумент <начало поиска> не задан или имеет значение 1, а значение аргумента <объем поиска> – отрицательное, то поиск выполняется с конца данных.

```
select gettextpos(c, '11*',1,1,-3) from test;
|0000000003 0000000011 11 2 26 4 35 4|
```

## Возвращаемое значение

1) Тип возвращаемого значения – VARCHAR(2000).

2) Структура возвращаемой строки:

<количество><разделитель><продолжение поиска> [<описатель элемента>]...

<описатель элемента> :=

<разделитель><позиция элемента><разделитель><длина элемента>

где:

- <разделитель> – символ пробела;
- <количество> – количество промаркированных элементов текста. Поле имеет фиксированную длину – 10 символов;
- <продолжение поиска> – позиция, с которой необходимо продолжать сканирование данных (после последнего выданного маркированного элемента). Поле имеет фиксированную длину – 10 символов. Если сканирование данных выполнено полностью, значение поля будет равно 0;
- <описатель элемента> – описатель маркированного элемента текста. Элементы описателя имеют парное значение: <позиция элемента> <длина элемента>, которое представляет, соответственно, позицию найденного элемента и его фактическую длину.

```
create or replace table test(c char(256));
insert into test values ('RELEX is one of the leading Russian
application and system software developers');
Запрос осуществляет поиск элементов текста, начинающихся с "develop" (с
учетом регистра) или совпадающих с "russian" (без учета регистра).
```

```
select gettextpos(c, '#develop*|russian',1,1,0) from test;
| 0000000002 0000000000 29 7 69 10 |
```

Результат поиска:

найдено 2 элемента текста (<количество> равно 0000000002)

найжены все элементы (<продолжение поиска> равно 0000000000)

первый найденный элемент находится на 29 позиции и имеет длину 7 символов (слово Russian)

второй найденный элемент находится на 69 позиции и имеет длину 10 символов (слово developers).

## Выборка текста

### Функция

Получение заданной порции текстовых данных .

### Спецификация

**GETTEXT**(<имя столбца>, <смещение>, <длина>)

### Синтаксические правила

- 1) <имя столбца> должно принадлежать столбцу типа BLOB, EXTFILE, CHAR, VARCHAR, NCHAR, NCHAR VARYING.
- 2) <смещение> – целое положительное число, задающее начальную позицию требуемой порции текста. Первый символ текста имеет смещение 1;
- 3) <длина> – размер требуемой порции текста в символах (целое число в диапазоне от 1 до 2000). Количество символов (и смещение) задаётся в символах оригинального документа. Если у пользователя стоит кодировка MBCS или UTF-8, то длина результата преобразования может превысить 4000 байт, разрешённых для столбца. В результате часть данных не будет передана. Нужно учитывать эту особенность в случае, когда извлекаются непрерывные куски текста несколькими порциями.

### Возвращаемое значение

- 1) возвращается затребованная порция прошедшего через фразовый фильтр текста, определяемая указанными смещением и длиной;
- 2) в возвращаемом значении порция текста после последнего символа текста столбца заполняется пробелами;
- 3) если размер требуемой порции текста превышает 2000 символов, то возвращается 2000 символов текста, а оставшаяся часть результата заполняется пробелами (ЛИНТЕР 6.0 и выше);
- 4) для файлов типа XML и HTML функция возвращает только чистый текст (пары «атрибут-значение» игнорируются);
- 5) значение NULL возвращается в случае, если требуемый файл не существует (для столбца типа EXTFILE) или если в процессе получения требуемой порции текста произошла ошибка;
- 6) правила выбора фразового фильтра совпадают с правилами, используемыми при создании фразового индекса со значениями флагов, принятыми по умолчанию для данного типа столбца (см. функцию create phrase index);
- 7) для версии 6.0 и выше: даже если в системной таблице \$\$\$CHARSET не задана кодировка 866, 1251 или 20866, то перекодировка текста все равно будет выполнена правильно с помощью встроенных в СУБД ЛИНТЕР внутренних таблиц кодировки.
- 8) В противном случае символы с кодами меньше 0x80 извлекаются как есть (подразумевается 7-битный ASCII), а остальные символы заменяются на '?'

 Поскольку выдается текст, прошедший через фильтр, в нем могут отсутствовать знаки препинания, символы разметки и т.п.

### Пример

Найти позицию третьего повторения фразы «баз данных» в документе с идентификатором 10

```
select instr(gettext(text_doc,1,1500), 'баз данных',1,3) from  
"ph" where id_doc=10;
```

## Время создания фразового индекса

### Функция

Получить дату и время последнего индексирования записи.

### Спецификация

**INDEXTIME** (<имя\_столбца>)

### Синтаксические правила

1) <имя\_столбца> – столбец данных любого типа, для которого разрешено создание фразового индекса.

### Возвращаемое значение

- 1) значение типа DATE, содержащее дату и время (по Гринвичскому меридиану) последней индексации данных столбца;
- 2) NULL-значение в случае:
  - фразовый индекс для данного столбца не создан или запись не индексирована;
  - запись изменялась после индексации (для символьных полей с индексом, имеющим модификатор DEFERRED);
  - значение столбца NULL.

### Пример

```
select indextime(text_doc) from "ph";
```

## Сформировать значение типа EXTFILE

### Функция

Сформировать значение типа EXTFILE для использования в операции INSERT/UPDATE (т. к. значение типа EXTFILE нельзя задать явно).

### Спецификация

**EXTFILE** (<имя\_файла> | NULL [, <имя\_фильтра>])

### Синтаксические правила

- 1) <имя\_файла> задает полный или относительный путь к имени файла;
- 2) <имя\_фильтра> должно соответствовать зарегистрированному имени фильтра.

### Возвращаемое значение

- 1) значение типа EXTFILE;
- 2) NULL, если значение аргумента <имя\_файла> равно NULL.



При добавлении (изменении) значения типа EXTFILE в столбец заносится (изменяется) ссылка на внешний файл. Проверка корректности ссылки, т. е. реального существования файла, не выполняется

### Примеры

```
insert into tabl("Музыка", "Слова")
values (extfile('music.doc'), extfile('text.doc'));
```

```
insert into tab1("Музыка", "Слова") values  
(extfile(?), extfile(?));
```

## Получить значение столбца типа файла

### Функция

Получить значение столбца типа EXTFILE.

### Спецификация

**FILENAME** (<имя\_столбца>)

### Синтаксические правила

<Имя\_столбца> должно принадлежать столбцу с типом данных EXTFILE.

### Возвращаемое значение

- 1) символьная строка char (512), содержащая путь к внешнему файлу заданного столбца. Реальное существование файла не проверяется;
- 2) символы с кодом, большим 127, заменяются символом '?' (знак вопроса);
- 3) если имя каталога было явно указано при формировании значения типа EXTFILE, то оно включается в возвращаемое значение, при этом символ-разделитель ':' (двоеточие) в спецификации каталога заменяется на символ '|' (вертикальная черта);
- 4) если значение аргумента NULL, результат NULL.

### Пример

```
drop table ext;  
create table ext (id int, ext1 extfile);  
insert into ext values (1, EXTFILE('c:\autoexec.bat'));  
insert into ext values (2, EXTFILE('c:\config.sys'));  
insert into ext values (3, EXTFILE('d:\test1.txt'));  
insert into ext values (4, EXTFILE('c:\test\test2.txt'));  
update ext set ext1 = EXTFILE('c:\autoexec.bat', ASCTEXT2TEXT)  
where id <=2;  
select id, cast filename(ext1) as char (20) from ext;  
|1|c|/autoexec.bat  
|2|c|/autoexec.bat  
|3|d|/test1.txt  
|4|d|/test/test2.txt
```

## Получить номер фильтра

### Функция

Получить номер фильтра, установленного для столбца типа EXTFILE.

## Спецификация

**FILTER** (<имя\_столбца>)

## Синтаксические правила

<Имя\_столбца> должно принадлежать столбцу с типом данных EXTFILE.

## Возвращаемое значение

- 1) значение типа INTEGER, содержащее номер установленного для данного столбца фильтра. Совпадает со значением необязательного параметра функции EXTFILE (если задан). В противном случае возвращается 0;
- 2) если значение аргумента NULL, результат NULL.

## Пример

Запрос выдает тип внешнего файла и назначенный этому файлу фильтр.

```
select distinct
substr(filename(text_doc),instr(filename(text_doc),'.')) as
"Тип файла",
filter(text_doc) as "Номер фильтра" from "ph";
```

Тип файла	Номер фильтра
.doc	-15
.htm	-12
.xml	-12

## Дата обновления файла

### Функция

Получить дату и время последнего изменения внешнего файла.

## Спецификация

**FILETIME** (<имя\_столбца>)

## Синтаксические правила

<Имя\_столбца> должно принадлежать столбцу с типом данных EXTFILE.

## Возвращаемое значение

- 1) значение типа DATE, содержащее дату и время (по Гринвичскому меридиану) последнего изменения существующего внешнего файла;
- 2) если значение аргумента NULL, результат NULL.

## Пример

```
select filetime(text_doc) from "ph";
```

### Размер внешнего файла

#### Функция

Получить размер внешнего файла.

#### Спецификация

**FILESIZE** (<имя\_столбца>)

#### Синтаксические правила

<Имя\_столбца> должно принадлежать столбцу с типом данных **EXTFILE**.

#### Возвращаемое значение

- 1) значение типа **BIGINT**, содержащее размер внешнего файла;
- 2) **NULL**-значение в случае:
  - внешний файл не существует;
  - значение столбца **NULL**.

#### Пример

Найти файл максимального размера.

```
SELECT id_doc FROM "ph"  
WHERE filesize(text_doc) = (SELECT MAX(filesize(text_doc)) FROM  
"ph");
```

### Каталог внешних файлов

#### Функция

Получить путь к каталогу, в котором по умолчанию располагаются внешние файлы.

#### Спецификация

**DEFAULT** (<имя\_столбца>)

#### Синтаксические правила

<Имя\_столбца> должно принадлежать столбцу с типом данных **EXTFILE**.

#### Возвращаемое значение

- 1) Путь к каталогу, где ищутся внешние файлы, для которых указан относительный путь.

# Приложение

## Расширения SQL

### Создание таблицы со столбцом типа **EXTFILE**

```
CREATE OR REPLACE TABLE <имя_таблицы> ( ... <имя_столбца> EXTFILE  
[ ROOT ' <корневой_каталог>' ] ... );
```

При хранении в БД имя файла преобразуется следующим образом:

- для DOS/WIN32 все символы '\' заменяются на '/';
- для UNIX спецификация файла не меняется.

Если присутствует конструкция **ROOT**, то файлы с относительными именами (у которых первый символ не '/' в UNIX и не имя устройства в DOS/WIN32) ищутся относительно указанного каталога, иначе относительно каталога БД.

Если значение <корневой каталог> задает относительный путь, то он считается относительно каталога БД.

Значение <корневой каталог>, заданное в конструкции **ROOT**, хранится для типа **EXTFILE** как **DEFAULT**-значение, содержащее текстовую строку.

### Добавление данных

```
INSERT INTO <имя_таблицы> (... <имя_столбца> ...)  
  VALUES (... EXTFILE('<имя_файла>' [, <имя_фильтра>]) | NULL ... );  
<имя_файла> ::= NULL | ? | <спецификация файла>
```

### Изменение данных

```
UPDATE INTO <имя_таблицы> SET <имя_столбца> =  
  EXTFILE('<имя_файла>' [, <имя_фильтра>]) | NULL ... ;
```

### Установка нового корневого каталога

```
ALTER TABLE <имя_таблицы> ALTER COLUMN <имя_столбца>  
  SET ROOT '<имя_каталога>';
```

### Отмена корневого каталога

```
ALTER TABLE <имя_таблицы> ALTER COLUMN <имя_столбца>  
  DROP ROOT;
```

# Указатель операторов и функций

**ALTER FILTER**, c.9  
**ALTER PHRASE INDEX**, c.16  
**CANCEL DEFAULT FILTER FOR**, c.12  
**CREATE EXTERNAL FILTER**, c.8  
**CREATE INTERNAL FILTER**, c.7  
**CREATE PHRASE INDEX**, c.14  
**DEFAULT**, c.27  
**DEFAULT FILTER**, c.9  
**DROP DEFAULT FILTER**, c.10  
**DROP FILTER**, c.8  
**DROP PHRASE INDEX**, c.16  
**EXTFILE**, c.25  
**FILENAME**, c.25  
**FILESIZE**, c.27  
**FILETIME**, c.27  
**FILTER**, c.26  
**GETTEXT**, c.24  
**INDEXTIME**, c.25  
**REBUILD PHRASE INDEX**, c.16  
**SET DEFAULT FILTER**, c.10  
**SET DEFAULT FILTER FOR**, c.11



