СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

ЛИНТЕР®

ЛИНТЕР БАСТИОН

ЛИНТЕР СТАНДАРТ

Полнотекстовый поиск в базе данных

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 2
Назначение документа 2
Для кого предназначен документ 2
Необходимые предварительные знания 2
Основные понятия 3
Назначение 3
Условия применения 3
Элементы языка 5
Фильтры 5
Управление фильтрами 6
Создание внутреннего фильтра 6
Создание внешнего фильтра 7
Удаление фильтра 8
Изменение внешнего фильтра 8
Фильтры столбца 9
Назначение фильтра для столбца 9
Модификация фильтра столбца 10
Отмена фильтра столбца 10
Фильтры файлов 11
Установка фильтра для указанного типа файлов 11
Отмена фильтра для расширения файла 12
Правила распознавания слов при полнотекстовом поиске 12
Индексы 13
Создание фразового индекса 14
Правила выбора фильтра 15
Изменение модификатора фразового индекса 16
Обновление фразового индекса 16
Удаление фразового индекса 17
Управление полнотекстовым поиском 17
Предикат полнотекстового поиска 17
Шаблон полнотекстового поиска 19
Примеры полнотекстового поиска 24
Функции 24
Местоположение искомых элементов текста во внешнем файле 24
Выборка порции текста внешнего файла 28
Время последнего индексирования внешнего файла 29
Идентификация хранимых во внешнем файле данных 30
Путь к внешнему файлу 31
Номер фильтра внешнего файла 31
Дата обновления внешнего файла 32
Размер внешнего файла 32
Каталог размещения внешних файлов 33
Приложение. Расширения SQL 34
Указатель операторов и функций 35

Предисловие

Назначение документа

Документ описывает возможности полнотекстового (фразового) поиска СУБД ЛИНТЕР.

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

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.

Дополнительные документы

- Справочник по SQL
- Справочник кодов завершения
- Запуск и останов СУБД ЛИНТЕР в среде ОС Windows
- Запуск и останов СУБД ЛИНТЕР в среде ОС Linux

© Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

Основные понятия

Назначение

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 – конвертеров или фильтров.

СУБД ЛИНТЕР со средствами полнотекстового 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 полнотекстового поиска дают возможность упростить схему хранения данных в приложении и избежать создания некоторых дополнительных таблиц.

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

Условия применения

Для применения полнотекстового поиска необходимо выполнение следующих условий:

1) наличие в БД системных таблиц `$$$FILTER` и `$$$EXTENSION` для поддержки системы полнотекстового 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 полнотекстового поиска);
2) наличие внешних фильтров 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.



Примечание
В составе СУБД ЛИНТЕР 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.

5) увеличение размера `/PPOOL` (соотносимого с имеющимися задачами и возможностями СУБД) положительно влияет на скорость создания и обновления индексов (см. описание ключа /PPOOL в документе «Запуск и останов СУБД ЛИНТЕР в среде ОС Windows» или описание ключа /PPOOL в документе «Запуск и останов СУБД ЛИНТЕР в среде ОС Linux»).

Элементы языка

Фильтры

Быстрый поиск документов в системе полнотекстового 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



Так как в качестве документов могут использоваться документы любых форматов (TXT, DOC, RTF, PDF, HTML и т.д.), то обрабатывать все различные форматы в системе полнотекстового поиска нецелесообразно. Логичным решением этой проблемы является система *конвертеров* (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 конвертер 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 (пункт Создание внешнего фильтра).

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

Имя фильтра	Формат входного файла
UNIXML2TEXT	HTML, XML в коде UNICODE
DOCRTF2TEXT	RTF, PDF, DOC, XLS, PPT, DOCX, XLSX, PPTX, ODT, ODS, ODP[1)]
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

Фильтры возвращают текст в кодировке UNICODE.

Примечание
Информация о всех доступных фильтрах (как внутренних, так и подключенных пользователями) хранится в системной таблице `$$$FILTER`. Создается путем запуска файла `search.sql`. Таблица имеет 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 */
);
```

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

[1] <создание внутреннего фильтра>::=
CREATE [IF NOT EXISTS |OR REPLACE] INTERNAL FILTER <имя фильтра>
[DESCRIPTION <описание>];

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

[2] <имя фильтра>::= <идентификатор>
[3] <описание>::= <строковый литерал>

Синтаксические правила

1) <Имя фильтра> должно совпадать с одним из предопределенных имен фильтров (см. таблицу 1).
2) Дублирование фильтров в таблице $$$FILTER не допускается, т.е. повторное добавление внутреннего фильтра игнорируется. Если необходимо модифицировать фильтр, его сначала надо удалить, затем создать заново.
3) Длина <строкового литерала> в <описании> – не более 256 знаков. Его значение помещается в поле $$$DESC таблицы $$$FILTER.

Общие правила

Созданный фильтр становится доступным для использования.

Примеры

```
CREATE OR REPLACE INTERNAL FILTER "ASCTEXT2TEXT";

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

Создание внешнего фильтра

Функция

Подключение к СУБД ЛИНТЕР фильтра, разработанного пользователем.

Спецификация

[1] <создание внешнего фильтра>::=
CREATE [IF NOT EXISTS | OR REPLACE]
[EXTERNAL] FILTER <имя фильтра>=<номер>
MODULE <спецификация файла> [DESCRIPTION <описание>];
[2] <имя фильтра>::= <идентификатор>
[3] <номер>::= целое положительное число
[4] <спецификация файла>::= <строковый литерал>
[5] <описание>::= <строковый литерал>

Синтаксические правила

1) <Имя фильтра> не должно совпадать с одним из предопределенных имен внутренних фильтров (см. таблицу 1).
2) <Номер> должен быть уникальным среди номеров таблицы $$$FILTER.
3) Дублирование фильтров в таблице $$$FILTER не допускается, то есть добавление внешнего фильтра с тем же именем (хотя и с другим <номером>) игнорируется.
4) Длина <строкового литерала> в <описании> – не более 256 знаков. Его значение помещается в поле $$$DESC таблицы $$$FILTER.
5) <Спецификация файла> задает местоположение и 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

[1] <удаление фильтра>::= DROP FILTER <имя фильтра>;
[2] <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 (файл фильтра) сохраняется в БД.
3) Файл внешнего фильтра физически не удаляется (сохраняется в файловой системе).

Пример

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

Изменение внешнего фильтра

Функция

Замена файла (библиотеки) для установленного внешнего фильтра.

Спецификация

[1] <модификация внешнего фильтра>::=
ALTER FILTER <имя фильтра> MODULE <спецификация файла>;
[2] <имя фильтра>::= <идентификатор>
[3] <спецификация файла>::= <строковый литерал>

Синтаксические правила

1) <Имя фильтра> должно содержаться в таблице $$$FILTER.
2) <Спецификация файла> задает местоположение и имя нового файла (библиотеки) фильтра.

Общие правила

1) Для указанного фильтра текущий файл (библиотека) фильтра заменяется новым (в системной таблице $$$FILTER) и становится доступным для использования.
2) Прежний файл фильтра физически не удаляется (сохраняется в файловой системе).

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

Пример

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

Фильтры столбца

Назначение фильтра для столбца

Функция

Применение фильтра для столбца таблицы (при создании таблицы).

Спецификация

[1] <применение фильтра>::=
CREATE TABLE [<имя схемы>.]<имя таблицы>
(… <имя столбца> <тип> DEFAULT FILTER <имя фильтра>…);
[2] <имя схемы>::= <идентификатор>
[3] <имя таблицы>::= <идентификатор>
[4] <тип>::= <идентификатор>
[5] <имя столбца>::= <идентификатор>
[6] <имя фильтра>::= <идентификатор>

Синтаксические правила

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

Общие правила

1) 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;
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;

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.

Примечание
В текущей версии СУБД ЛИНТЕР работа с внешними фильтрами не поддерживается.

Модификация фильтра столбца

Функция

Изменение фильтра для столбца таблицы (при модификации таблицы).

Спецификация

[1] <модификация фильтра>::=
ALTER TABLE [<имя схемы>.]<имя таблицы>
ALTER COLUMN <имя столбца> SET DEFAULT FILTER <имя фильтра>;
[2] <имя схемы>::= <идентификатор>
[3] <имя таблицы>::= <идентификатор>
[4] <имя столбца>::= <идентификатор>
[5] <имя фильтра>::= <идентификатор>

Синтаксические правила

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

Общие правила

Столбцу назначается указанный фильтр по умолчанию.

Пример

```
ALTER TABLE TEST_BLOB ALTER COLUMN Document SET DEFAULT FILTER
 "UNIXML2TEXT";
```

Отмена фильтра столбца

Функция

Отмена установленного для столбца фильтра (при модификации таблицы).

Спецификация

[1] <отмена фильтра>::=
ALTER TABLE [<имя схемы>.]<имя таблицы>
ALTER COLUMN <имя столбца> DROP DEFAULT FILTER;
[2] <имя схемы>::= <идентификатор>
[3] <имя таблицы>::= <идентификатор>
[4] <имя столбца>::= <идентификатор>

Синтаксические правила

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

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

Пример

```
ALTER TABLE TEST_BLOB ALTER COLUMN Document DROP DEFAULT FILTER;
```

Фильтры файлов

Для столбцов типа EXTFILE выбор фильтра может быть сделан самой СУБД ЛИНТЕР автоматически – по расширению файла либо явно задан с помощью SQL конструкции. Для автоматического выбора фильтра используется системная таблица $$$EXTENSION, имеющая следующую структуру:

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

Таблица создается с помощью файла search.sql.

Установка фильтра для указанного типа файлов

Функция

Установка фильтра по умолчанию для указанного типа файлов.

Спецификация

[1] <установка фильтра для файла>::=
SET DEFAULT FILTER <имя фильтра> FOR <тип файла>;
[2] <имя фильтра>::= <идентификатор>
[3] <тип файла>::= <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) <Тип файла> – строковый литерал, задающий расширение (тип) файла, который должен обрабатываться данным фильтром.

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	docrtf2text
XML, xml, HTM, htm, HTML, html, PHTML, phtml, SHTML, shtml	ascxml2text

Примечание
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.

Пример

Установить фильтр DOCRTF2TEXT для файлов типа `.ppp`.

```
SET DEFAULT FILTER "DOCRTF2TEXT" FOR 'ppp';
```

Отмена фильтра для расширения файла

Функция

Отмена ранее установленного фильтра по умолчанию для указанного типа файлов.

Спецификация

[1] <отмена фильтра для файла>::=
CANCEL DEFAULT FILTER FOR <тип файла>;

Синтаксические правила

Указанный `<тип файла>` должен содержаться в таблице `$$$EXTENSION`.

Общие правила

Указанный фильтр перестает применяться по умолчанию для данного типа файлов.

Пример

Отменить фильтр для файлов типа `.ppp`.

```
CANCEL DEFAULT FILTER FOR 'ppp';
```

Правила распознавания слов при полнотекстовом поиске

Распознаваемый элемент «Слово» текста документа может включать:

- буквенно-цифровые символы (согласно стандарту UNICODE) и символ '_';
- 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	'>'
	' '

Максимальная длина слова, имени и значения атрибута составляет 64 однобайтового символа. Длинные слова при поиске усекаются до 64 однобайтовых символов.

Индексы

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) обычное индексирование: содержание поля заносится в индекс целиком. Из словосочетания «база данных» формируется одно индексное значение – «база данных». В этом случае для правильного поиска нужно вводить все слова в соответствующем порядке и с соответствующими разделителями. Предикаты LIKE и SIMILAR также позволяют использовать обычный индекс для поиска данных, соответствующих шаблонам;
2) фразовое, или полнотекстовое индексирование: каждое слово, входящее в документ, индексируется отдельно, разделители не учитываются. Таким образом, из словосочетания «база данных» в индекс войдут два слова: «база» и «данных». Предикат фразового поиска CONTAINS требует обязательного существования соответствующего фразового индекса. При поиске с использованием этого предиката могут как задаваться, так и не задаваться порядок введенных слов, расстояние между словами и т.д.
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.

Примечание
В настоящее время список стоп-слов пуст.

Создание фразового индекса

Функция

Создание фразового индекса.

Спецификация

[1] <создание фразового индекса>::=
CREATE [IF NOT EXISTS | OR REPLACE] PHRASE [<модификатор>]
[XML] INDEX <имя столбца> ON [<имя схемы>.]<имя таблицы>;
[2] <модификатор>::= IMMEDIATE | DEFERRED
[3] <имя столбца>::= <идентификатор>
[4] <имя схемы>::= <идентификатор>
[5] <имя таблицы>::= <идентификатор>

Синтаксические правила

1) <Имя столбца> должно принадлежать столбцу без фразового индекса.
2) Допустимый тип <имени столбца>: CHAR, VARCHAR, NCHAR, NCHAR VARYING, BLOB, EXTFILE.
3) Для столбцов типа BLOB разрешен только модификатор DEFERRED.
4) Флаг XML задает построение атрибутного 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
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) Модификатор 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 необходимый буфер памяти. Размер буфера задается в страницах размером 4K с помощью ключа /ppool командной строки запуска СУБД ЛИНТЕР. Минимальное значение 4K. По умолчанию используется ключ /PPOOL=32.

Примечания

1. Индексируемые документы могут иметь метки секретности. Все функции, связанные с полнотекстовым индексом или основанные на извлечении текста, а также функции для работы с полями типа EXTFILE, обрабатывают и учитывают при поиске метки секретности.
2. Если последнее слово в документе заканчивается символом @, -, /, ', \, то этот символ при создании индекса не учитывается.

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;
   - 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 документов атрибуты элементов текста игнорируются. Исключение составляют META-элементы, содержащие метаинформацию, касающуюся документа. META-элемент содержит 2 обязательных атрибута:

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

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

Если <Имя1>="Content-Type", то <Значение1> сканируется на предмет наличия информации об используемой документом кодировки, 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 создан в формате HTML. В противном случае тип документа, согласно спецификации форматов XML и XHTML, определяется однозначно.

Изменение модификатора фразового индекса

Функция

Изменение модификатора фразового индекса.

Спецификация

[1] <изменение модификатора>::=
ALTER PHRASE INDEX <имя столбца> ON
[<имя схемы>.]<имя таблицы> <модификатор>;
[2] <имя столбца>::= <идентификатор>
[3] <имя схемы>::= <идентификатор>
[4] <имя таблицы>::= <идентификатор>

Общие правила

1) <Имя столбца> должно принадлежать столбцу с фразовым индексом.
2) Модификатор 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 (эквивалентно конструкции REBUILD PHRASE INDEX).
7) Для полного перестроения индекса без DROP PHRASE INDEX можно использовать конструкцию CREATE OR REPLACE PHRASE INDEX.

Обновление фразового индекса

Функция

Обновление фразового индекса.

Спецификация

[1] <обновление фразового индекса> : : =
REBUILD PHRASE INDEX <имя столбца> ON
[<имя схемы>.]<имя таблицы>;
[2] <имя столбца> : : = <идентификатор>
[3] <имя схемы> : : = <идентификатор>
[4] <имя таблицы> : : = <идентификатор>

Синтаксические правила

`<Имя столбца>` должно принадлежать столбцу с фразовым индексом.

Общие правила

1) Выполняется обновление фразового индекса (индексируются добавленные либо измененные с момента создания или последнего обновления индекса элементы текста, и исключается информация об удаленных элементах текста).
2) Для индексов с модификатором IMMEDIATE обновление фразового индекса выполняется автоматически.
3) Для перестройки индекса необходимо иметь категорию доступа CONNECT.

Удаление фразового индекса

Функция

Удаление существующего фразового индекса.

Спецификация

[1] <удаление фразового индекса> : : =
DROP PHRASE INDEX <имя столбца> ON
[<имя схемы>.]<имя таблицы>;
[2] <имя столбца> : : = <идентификатор>
[3] <имя схемы> : : = <идентификатор>
[4] <имя таблицы> : : = <идентификатор>

Синтаксические правила

`<Имя столбца>` должно принадлежать столбцу с фразовым индексом.

Управление полнотекстовым поиском

Предикат полнотекстового поиска

Функция

Для выполнения полнотекстового поиска используется предикат полнотекстового поиска (расширение конструкции поискового оператора SELECT языка SQL).

Спецификация

[1] <предикат полнотекстового поиска> : : =
<имя столбца> [(] [NOT] CONTAINS[)]
[<модификатор> <модификатор> ...]
<шаблон полнотекстового поиска>;
[2] <имя столбца> : : = <идентификатор>

[3] <модификатор> : : =
SENSITIVE | PARTIALLY | AT_BEGIN | AT_END | FUZZY
[4] <шаблон полнотекстового 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 4 представлены значения модификаторов и их назначение.

Таблица 4. Модификаторы полнотекстового поиска

Модификатор	Назначение	Представление в шаблоне	Местоположение в шаблоне	Пример
SENSITIVE	Задает чувствительный к регистру поиск	#	Перед словом	#Word
PARTIALLY	Задает поиск документов, в которых обозначенный шаблон поиска может встречаться в любом месте слова	*	В начале и конце	*WORD*
AT_BEGIN	Задает поиск документов, в которых обозначенный шаблон поиска может встречаться только в начале слов	*	В конце	WORD*
AT_END	Задает поиск документов, в которых обозначенный шаблон поиска может встречаться только в конце слов	*	В начале	*WORD
FUZZY	Задает 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» в таблице 4) применяется ко всем словам шаблона полнотекстового поиска.

Если модификатор должен быть применен не ко всем словам искомого документа, следует использовать специальный символ (см. столбец «Представление в шаблоне» в таблице 4) 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 «спецальная»:

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

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

Исходный документ	Найденный документ	soundex()	difference()
Спецальная	✔	C167	0
Специальноя	✔	C167	0
Спициальная	✔	C167	0
Специалная	✔	C167	0
Sпециалная	S167	1	
Тпециалноя	T167	1	
Специа	C160	1	
Специальнаяя	✔	C167	0

Шаблон полнотекстового поиска

Функция

Определение шаблона полнотекстового поиска.

Спецификация

[1] <шаблон полнотекстового поиска> : : = <поисковое выражение>
[2] <поисковое выражение> : : =
<слово>
| <атрибут> = <значение>
| <фраза>
| <не равно> <поисковое выражение>
| <поисковое выражение> <или> <поисковое выражение>
| <поисковое выражение> [<и>] <поисковое выражение>
| (<поисковое выражение>)
[3] <не равно> : : = !
[4] <или> : : = |
[5] <и> : : = &
[6] <атрибут> : : = <слово>
[7] <значение> : : =
<слово>
| (['#' | '%' | '*'] "" <строка без символа> "" ['*'])
| (['#' | '%' | '*'] ' <строка без символа> ' ['*'])
[8] <фраза> : : =
""(<слово> |<группа>)
[[<расстояние>] (<слово> |<группа>) …]""
[9] <группа> : : = '('<слово> [<слово>…]')', всего до 6 слов
[10] <расстояние> : : =
'|' ((['+' | '-'] <смещение>) | (<нижняя граница> <верхняя граница>)) '|'
[11] <смещение> : : = целое беззнаковое число в диапазоне 1..10
[12] <нижняя граница> : : = целое число в диапазоне -10..10
[13] <верхняя граница> : : =
целое число в диапазоне <нижняя граница>..10

<слово> составляют:

- буквенно-цифровые символы (согласно стандарту UNICODE) и символ '_';
- символы, значимые в середине слова: '@', '-', '/'. Указанные символы не могут быть первым и последним символом слова, за исключением шаблона поиска по концу или части слова;
- специальные символы: '#', '%', '*'. Эти символы могут содержаться только в начале или в середине слова, кроме символа '*', который может быть и последним символом слова (см. таблицу 4);
- одиночный символ '*', заменяющий любое слово.

Примечания

1. Некоторые допустимые (см. раздел Правила распознавания слов при полнотекстовом поиске) имена атрибутов не могут быть найдены согласно описанным правилам. Для их поиска следует использовать специальный символ '*'.
2. 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:

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

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

Общие правила

1) Значение специальных символов '#', '%', '*' определяет таблица 4.
2) Конструкция ! <поисковое выражение> задает поиск документов, не содержащих данное поисковое выражение.

   Примеры.

   а) 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 not contains 'встроенн* баз* дан*';
```

   б) найти все версии документов, где отсутствуют упоминания об отдельных словах: встроенные, базы, данные.

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

3) 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*';
```

4) 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 и о СУБД ЛИНТЕР.

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

5) Конструкция (<поисковое выражение>) определяет логический порядок разбора поискового выражения.
6) Конструкция <атрибут> = <значение> задает атрибутный поиск в документах, отвечающих спецификации XML (в том числе HTML); <атрибут> задает имя атрибута, <значение> задает значение этого атрибута.

Фильтры ASCXML2TEXT и UNIXML2TEXT используются для файлов в форматах XML, XHTML, HTML.

Файлы этих форматов обрабатываются следующим образом:

- в случае XML-файла извлекаются пары <имя атрибута=значение атрибута>, которые могут быть включены в индекс в случае использования модификатора xml в конструкции create phrase index;
- в случае HTML и XHTML файлов атрибуты элементов игнорируются. Исключение составляют META-элементы, содержащие метаинформацию, касающуюся документа в целом. META-элемент 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 кодировки, 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 создан в формате HTML. В противном случае тип документа определяется согласно спецификации форматов XML и XHTML;
- теги без значений типа <display-name> игнорируются в любом случае.

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!(c)"';
```

7) <Фраза> задает поиск последовательно расположенных в тексте слов.

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

Пример.

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

8) <Расстояние> задает «расстояние» между парой слов. Так, значение1 (значение по умолчанию) соответствует последовательно идущим словам, значение2 показывает, что между данными словами должно находиться некоторое одно слово и т.д. Отрицательные значения указывают на обратный порядок слов.

9) Допустимы 3 способа задания расстояния:

- <нижняя граница> и <верхняя 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 «СУБД 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"';
```

10) <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)"';
```

11) Длинные слова в шаблоне поиска усекаются до 64 букв.

12) Апостроф является допустимым символом внутри слова для полнотекстового 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 во внешнем файле

Функция

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

пробела. Для документов с элементами форматирования текста (типа `.doc`, `.pdf` и др.), функция обычно применяется в паре с функцией GetText, которая преобразует форматированное представление документа в обычный неформатированный текст.

Спецификация

[1] <синтаксис>::= GETTEXTPOS(<имя столбца>, <поисковое выражение>[, <тип поиска>] [, <начало поиска>][, <объем поиска>])
[2] <имя столбца>::= <идентификатор>
[3] <поисковое выражение>::= <шаблон поиска> [|<шаблон поиска>...]
[4] <шаблон поиска>::= <LIKE-шаблон> | <CONTAINS-шаблон>
[5] <LIKE-шаблон>::= <символьный литерал>
[6] <CONTAINS-шаблон>::= <символьный литерал>
[7] <тип поиска>::= 1 | 2 | 5 | 6
[8] <начало поиска>::= целочисленное положительное значение
[9] <объём поиска>::= целочисленное неотрицательное значение

Синтаксические правила

1) `<Имя столбца>` должно соответствовать столбцу типа BLOB, EXTFILE, CHAR, VARCHAR, NCHAR, NCHAR VARYING. Расширения языка SQL для работы с типом данных EXTFILE приведены в приложении.

2) `<LIKE-шаблон>` должен соответствовать спецификации `<предиката подобия>` (см. документ «Справочник по SQL», пункт «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;
|0000000002 0000000013 1 2 11 2|

select gettextpos(c,'11%|22|4%',2,1,10) from test;
|0000000006 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 (слова), 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 заданные элементы текста (слова) только в 1 и 3 строках, т.к. во второй строке набор символов '_11 aa_' рассматривается функцией GetTextPos, как два разных элемента текста (два слова), каждый из которых не удовлетворяет шаблону поиска _11%_:

```
select gettextpos(c, '_11%_',2,1,length(c)) from tst;
|0000000001 0000000000 1 4                          |
|0000000000 0000000000                               |
|0000000001 0000000000 1 6                          |
```

4) <CONTAINS-шаблон> должен соответствовать спецификации <предиката полнотекстового поиска>.

```
select gettextpos(c,'11*',1,1,10) from test;
|0000000004 0000000000 1 2 11 2 26 4 35 4|
```

Длина обоих шаблонов должна быть не больше 64 символов (иначе шаблон поиска усекается до 64 символов).

5) Если одновременно указывается несколько шаблонов поиска, то все они должны быть однотипными (либо только <LIKE-шаблоны>, либо <CONTAINS-шаблоны>).
6) <Тип поиска> это битовая маска, задающая атрибуты поиска:
   - 1 (001) – поиск по <CONTAINS-шаблону>;
   - 2 (010) – поиск по <LIKE-шаблону>;

   4 (100) – поиск с учетом регистра (обязательно объединение с одним из двух предыдущих атрибутов), т.е. <тип поиска> может так же принимать значения:
   - 5 (101) – поиск по <CONTAINS-шаблону> с учетом регистра;
   - 6 (110) – поиск по <LIKE-шаблону> с учетом регистра.

   Из двух флагов-атрибутов CONTAINS (1) и LIKE (2) обязательно должен быть установлен ровно один.
7) Если <тип поиска> не задан, по умолчанию используется значение1 (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 в байтах (значение типа INTEGER), начиная с которой необходимо выполнять поиск элементов по шаблону. Отсчет позиций начинается с 1. Аргумент необязательный. Если не задан, по умолчанию принимается значение1.
9) Если <начало поиска> не задано (подразумевается поиск с начала), то 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 необязательный. Если не задан, по умолчанию принимается значение 0.

11) Если аргумент <начало поиска> не задан или имеет значение1, а значение аргумента <объем поиска> – отрицательное, то поиск выполняется с конца данных.

Пример.

```
create or replace table test(c char(100));
insert into test(c) values ('25 a11 22 333 11bc 4411 55 666 1177
 811 1199');

select gettextpos(c,'11*',1,1,-3) from test;
|0000000000 0000000000 |
(искомые элементы текста не найдены)

select gettextpos(c,'11*',1,length(c)/2,-3) from test;
|0000000001 0000000000 15 4|

(найден 1 элемент текста 11bc, находящийся на 15 позиции и имеющий
 длину 4 символа).
```

12) Аргументы <поисковое выражение>, <начало поиска> и <объем поиска> могут быть заданы <SQL-параметром>, который должен содержать спецификацию типа данных параметра.

```
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,? (char(5)),2, ?(int),? (int)) from test;
11%
1
2
|0000000002 0000000013 1 2 11 2|
```

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 (в байтах) 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)
найдены 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 порции текста внешнего файла

Функция

Получение заданной порции текстовых данных внешнего файла.

Спецификация

[1] <синтаксис>::=
GETTEXT(<имя столбца>,<смещение>,<длина>)
[2] <имя столбца>::= <идентификатор>
[3] <длина>::= целое положительное число

Синтаксические правила

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

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

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 допустимые для столбца 4000 байт. 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.
4) Для файлов типа XML и HTML функция возвращает только чистый текст (пары «атрибут-значение» игнорируются).
5) NULL-значение 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, раздел Создание фразового индекса).
7) 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. В противном случае символы с кодами меньше 0x80 извлекаются как есть (подразумевается 7-битный ASCII), а остальные символы заменяются на '?'.

Примечание
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 последнего индексирования внешнего файла

Функция

Получение даты и времени последнего индексирования записи типа данных EXTFILE.

Спецификация

[1] <синтаксис> : : = INDEXTIME(<имя столбца>)

[2] <имя столбца>::=<идентификатор>

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";
```

Идентификация хранимых во внешнем файле данных

Функция

Формирование идентификатора хранимой в типе данных EXTFILE информации для использования в операции INSERT/UPDATE (т.к. значение идентификатора информации в этих операциях нельзя задать явно).

Спецификация

[1] <синтаксис>::=
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.

Примечание
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 tab1("Музыка", "Слова")
values (extfile('music.doc'), extfile('text.doc'));
```

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

Путь к внешнему файлу

Функция

Получение пути к файлу, хранящемуся в указанном столбце типа данных EXTFILE.

Спецификация

[1] <синтаксис>::= FILENAME(<имя столбца>)
[2] <имя столбца>::= <идентификатор>

Синтаксические правила

1) <Имя столбца> должно принадлежать столбцу с типом данных EXTFILE.

Возвращаемое значение

1) Символьная строка char (511), 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|c:\test\test2.txt   |
```

Номер фильтра внешнего файла

Функция

Получение номера фильтра, установленного для столбца типа данных EXTFILE.

Спецификация

[1] <синтаксис>::= FILTER(<имя столбца>)
[2] <имя столбца>::= <идентификатор>

Синтаксические правила

1) <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 внешнего файла

Функция

Получение даты и времени последнего изменения внешнего файла, значение пути к которому хранится в столбце типа данных EXTFILE.

Спецификация

[1] <синтаксис>::= FILETIME(<имя столбца>)
[2] <имя столбца>::= <идентификатор>

Синтаксические правила

1) <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

Получение размера внешнего файла, значение пути к которому хранится в столбце типа данных EXTFILE.

 © Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

Спецификация

[1] <синтаксис>::= FILESIZE(<имя столбца>)
[2] <имя столбца>::= <идентификатор>

Синтаксические правила

1) <Имя столбца> должно принадлежать столбцу с типом данных EXTFILE.

Возвращаемое значение

1) Значение типа BIGINT, содержащее размер внешнего файла.
2) NULL-значение в случае: если внешний файл не существует или 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 размещения внешних файлов

Функция

Получение пути к каталогу, в котором по умолчанию располагаются типы данных EXTFILE.

Спецификация

[1] <синтаксис>::= DEFAULT(<имя столбца>)
[2] <имя столбца>::= <идентификатор>

Синтаксические правила

1) <Имя столбца> должно принадлежать столбцу с типом данных EXTFILE.

Возвращаемое значение

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

Пример

```
create or replace table "ph" (id_doc int, ext extfile root '.
\sql_test\phrase');
select default(ext) from "ph";
|.\sql_test\phrase
```

Приложение
Расширения SQL

Создание таблицы со столбцом типа EXTFILE

```
CREATE OR REPLACE TABLE [<имя схемы>.]<имя таблицы>
   ( ... <имя столбца> EXTFILE [ ROOT '<корневой
 каталог>' ] ... );
```

При хранении в БД имя файла преобразуется следующим образом:

- для DOS/Windows все символы '\' заменяются на '/';
- для ОС Linux, ЗОСРВ Нейтрино спецификация файла не меняется.

Если присутствует конструкция ROOT, то файлы с относительными именами (у которых первый символ не '/' в ОС Linux, ЗОСРВ Нейтрино и не имя устройства в Windows) 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 схемы>.]<имя таблицы> [(... <имя столбца> ...)]
  VALUES (... EXTFILE('<имя файла>' [, <имя фильтра>]) |
 NULL ... );
<имя файла>::=NULL | ? |<спецификация файла>
```

Изменение данных

```
UPDATE [<имя схемы>.]<имя таблицы> SET <имя столбца> =
  EXTFILE('<имя файла>' [,<имя фильтра>]) | NULL ... ;
```

Установка нового корневого каталога

```
ALTER TABLE [<имя схемы>.]<имя таблицы> ALTER COLUMN <имя столбца>
  SET ROOT '<имя каталога>';
```

Отмена корневого каталога

```
ALTER TABLE [<имя схемы>.]<имя таблицы> ALTER COLUMN <имя столбца>
  DROP ROOT;
```


© Полнотекстовый поиск в базе данных. АО НПП «РЕЛЭКС», 1990-2026

Указатель операторов и функций

A
ALTER FILTER, 8
ALTER PHRASE INDEX, 16

C
CANCEL DEFAULT FILTER FOR, 12
CREATE EXTERNAL FILTER, 7
CREATE INTERNAL FILTER, 6
CREATE PHRASE INDEX, 14

D
DEFAULT, 33
DEFAULT FILTER, 9
DROP DEFAULT FILTER, 10
DROP FILTER, 8
DROP PHRASE INDEX, 17

E
EXTFILE, 30

F
FILENAME, 31
FILESIZE, 33
FILETIME, 32
FILTER, 31

G
GETTEXT, 28
GETTEXTPOS, 25

I
INDEXTIME, 29

R
REBUILD PHRASE INDEX, 17

S
SET DEFAULT FILTER, 10
SET DEFAULT FILTER FOR, 11