



Ред База Данных

Версия 2.1

Полнотекстовый поиск

© Корпорация Ред Софт 2008

Данный документ содержит описание полнотекстового поиска в СУБД «Ред База Данных» 2.1, а также основных приемов работы с подсистемой полнотекстового поиска. Документ рассчитан на пользователей, знакомых с принципами организации баз данных СУБД «Ред База Данных» и языком SQL.

www.red-soft.biz

Содержание

Глоссарий.....	4
Введение.....	5
1 Службные таблицы полнотекстового поиска.....	7
2 Хранимые процедуры полнотекстового поиска.....	9
3 Пример использования полнотекстового поиска.....	13
3.1 Создание индекса.....	13
3.2 Удаление индекса.....	13
3.3 Добавление полей в индекс.....	13
3.4 MIME формат документа.....	14
3.5 Удаление полей из индекса.....	14
3.6 Обновление метаданных системы полнотекстового поиска.....	14
3.7 Переиндексация.....	15
3.8 Поиск.....	15
3.9 Синтаксис запросов поиска.....	16
3.9.1 Оператор OR.....	17
3.9.2 Оператор AND.....	17
3.9.3 Оператор "+".....	17
3.9.4 Оператор NOT.....	17
3.9.5 Оператор "-".....	17
3.9.6 Группировка булевых операторов.....	18

Глоссарий

Демон – процесс, работающий в фоновом режиме без прямого общения с пользователем.

Индекс – объект, создаваемый системой Lucene на основе анализа содержимого документа, необходимый для организации поиска.

Индексация – обновление индекса.

Метаданные – данные о данных. Совокупность системных объектов (таблиц, триггеров и т. д.).

Морфологический поиск – поиск с учётом морфологии (всех возможных форм слова).

Поиск по неточному соответствию – позволяет отыскать требуемую информацию при наличии орфографических ошибок в документе или в запросе.

Полнотекстовый поиск - поиск документа в базе данных текстов на основании содержимого этих документов, а также совокупность методов оптимизации этого процесса.

Порядок сопоставления (Collation) – устанавливает для определённого набора символов порядок их следования (то есть правила сравнения и сортировки символов), а также определяет детали верхнего и нижнего регистров одного и того же символа. В совокупности с набором символов порядок сопоставления определяет алфавит для конкретного языка.

Терм поиска – термин, который ищется на полное совпадение. Lucene поддерживает простые и сложные термы. Простые термы состоят из одного слова, сложные из нескольких, при этом отдельные термы в составе сложного терма заключаются в кавычки.

Фраза поиска – исходная информация для осуществления поиска с помощью поисковой системы. Поисковый запрос задаётся в виде набора термов и операторов.

MIME-формат – в контексте системы полнотекстового поиска это формат документа, исходя из которого определяются правила извлечения данных из документа при индексации.

Lucene – свободно распространяемая библиотека для высокоскоростного полнотекстового поиска.

RDB\$DB_KEY – Это "номер записи". Его можно использовать в качестве уникального идентификатора записи, так же как и ее поле первичного ключа. Однако rdb\$db_key по ходу работы может меняться. Физически он представляет собой номер таблицы, номер страницы и смещение на запись (причем не на конкретную версию, а вообще на пакет версий этой записи, если они есть).

Введение

В «Ред База Данных» 2.1 реализованы средства полнотекстового поиска в базе данных. Полнотекстовый поиск основан на свободно-распространяемой библиотеке Lucene. Эта библиотека предоставляет функции индексирования и поиска, доступ к этим функциям реализуется через API Lucene.

Полнотекстовый поиск позволяет:

- производить поиск по неточному соответствию;
- производить морфологический поиск;
- производить поиск по нескольким объектам БД (поиск по нескольким столбцам и таблицам).

Поиск может осуществляться по текстовым полям (char, varchar), а также по бинарным и текстовым BLOB-полям. При этом BLOB-поля бинарного типа могут содержать внутри себя документы следующих приложений:

- Acrobat (pdf);
- MS Word (doc);
- MS Excel (xls);
- Microsoft PowerPoint;
- RTF;
- Open Office Writer (odt);
- Html.

В «Ред База Данных» 2.1 в системе полнотекстового поиска используется реализация Lucene на языке Java, поэтому для использования описываемого в этом руководстве функционала необходимо установить JDK не ниже 1.6. Настройка параметров взаимодействия сервера «Ред База Данных» с виртуальной машиной Java осуществляется с помощью конфигурационного файла `jvm.conf`, который расположен в корневом каталоге установки сервера.

Для обозначения комментариев в этом файле используется символ «#». Текст, следующий за символом «#», до конца строки также является комментарием. В файле необходимо настроить следующие параметры:

- полный путь к виртуальной машине
- путь к java-библиотекам

Полный путь к виртуальной машине указывается в первой, не закомментированной строке. При этом может быть указан как путь так и имя переменной окружения, которая его хранит:

```
C:\Java1_6\jre\bin\client\jvm.dll #полный путь  
<jvm_from_JAVA_HOME> #имя переменной окружения
```

Путь к Java-библиотекам, которые реализуют функции полнотекстового поиска задается с помощью параметра `-cp` (сокр. от class path). По умолчанию сервер подгружает *.jar файлы, расположенные только в директории `java_lib`, относительно корневой директории сервера. С помощью параметра `-cp` можно указать дополнительные *.jar файлы, которые будут подгружаться вместе с *.jar файлами из директории `java_lib`. При этом все имена jar-файлов перечисляются в одну строку, в качестве разделителя используется символ «;» для Windows систем, и символ «:» для Unix систем:

```
-cp java_lib/commons-  
beanutils-1.6.1.jar;java_lib/commons-  
collections-2.1.jar
```

Для корректной работы полнотекстового поиска необходимо наличие следующих Java-библиотек:

- fb-lu.jar
- jaybird-esp-2.1.2.jar
- jaybird-full-2.1.2.jar
- lius.jar
- lucene-highlighter-2.1.0.jar
- lucene-analyzers-2.1.0.jar
- lucene-highlighter-2.1.0.jar
- MimeType.jar.

Эти библиотеки должны располагаться либо в подкаталоге java-lib либо пути к ним должны быть прописаны в параметре -cp в конфигурационном файле jvm.conf.

1 Служебные таблицы полнотекстового поиска

Для функционирования полнотекстового поиска в базе данных должны присутствовать все необходимые служебные объекты. Для автоматического создания этих объектов необходимо в конфигурационном файле сервера указать в качестве значения параметра `InitScript` путь к скрипту инициализации. После чего этот скрипт будет выполняться при создании каждой новой базы данных. По умолчанию имя скрипта инициализации — `init.sql`, он находится в корневом каталоге сервера. Например:

```
InitScript = init.sql
```

В полнотекстовом поиске используются четыре служебные таблицы:

Таблица 1.1 - Служебные таблицы системы полнотекстового поиска

Таблица	Описание
FTS\$INDICES	Метаданные индекса
FTS\$INDEX_SEGMENTS	Метаданные поля БД, входящего в индекс
FTS\$POOL	Содержит RDB\$DB_KEY для измененных, но не проиндексированных полей
FTS\$LUCENE_FILE_SYSTEM	Эмуляция файловой системы для хранения данных индекса

В служебной таблице FTS\$INDICES хранятся метаданные индексов. Описание FTS\$INDICES приведено в таблице 1.2:

Таблица 1.2 - Структура таблицы FTS\$INDICES

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$STORE	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Описание, где хранится индекс. Может быть NULL, в этом случае индекс будет храниться в таблице FTS\$LUCENE_FILE_SYSTEM. Значение по умолчанию NULL. Если задано значение "file", то индексы будут сохраняться в файловой системе
FTS\$DESCRIPTION	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Комментарий к индексу
FTS\$INDEX_STATUS	CHAR(1)	Статус индекса. Это поле может принимать следующие значения: 'I' inactive – индекс неактивный; 'N' new – индекс создан, требует полное переиндексирование; 'U' needs metadata update – требуется изменение метаданных, триггеров и т.д.; 'D' drop – индекс отмечен к удалению; 'C' complete – для индекса сделаны все изменения в метаданных и он индексируется

В таблице FTS\$INDEX_SEGMENTS хранятся данные о составе (сегментах) индексов – метаданные полей, входящих в индекс. Структура FTS\$INDEX_SEGMENTS приведена в таблице 1.3:

Таблица 1.3 - Структура таблицы FTSSINDEX_SEGMENTS

Имя поля	Тип	Описание
FTSSINDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTSSRELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица
FTSSFIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле
FTSSTRIGGER_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя триггера, который будет заполнять таблицу FTSSPOOL, после изменения данных (см. далее)
FTSSANALIZER	CHAR(255) CHARACTER SET UNICODE_FSS	Служебная информация, имя анализатора
FTSSMIME_TYPE	CHAR(127) CHARACTER SET UNICODE_FSS	Имя MIME формата документа, хранящегося в индексируемом поле, если в столбце хранятся документы одного типа ¹
FTSSMIME_FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя столбца, в котором хранится информация для определения MIME формата документа, хранящегося в индексируемом поле

Таблица FTSSPOOL содержит значения RDB\$DB_KEY для измененных, но не проиндексированных записей. FTSSPOOL имеет следующую структуру (таблица 1.4).

Таблица 1.4 - Структура таблицы FTSSPOOL

Имя поля	Тип	Описание
FTSSDB_KEY	CHAR(8) CHARACTER SET OCTETS	RDB\$DB_KEY записи, которая была добавлена, изменена или удалена

Таблица FTSSLUCENE_FILE_SYSTEM служит для хранения данных индекса. Данные индекса хранятся во внешних файлах и подгружаются в таблицу FTSSLUCENE_FILE_SYSTEM в BLOB-поле FTSSFILE_BODY. Структура FTSSLUCENE_FILE_SYSTEM приведена в таблице 1.5.

Таблица 1.5 - Структура таблицы FTSS LUCENE_FILE_SYSTEM

Имя поля	Тип	Описание
FTSSINDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTSSFILE_NAME	VARCHAR(255) CHARACTER SET UNICODE_FSS	Имя файла
FTSSLAST_MODIFY_TIME	TIMESTAMP	Время последнего изменения файла
FTSSFILE_BODY	BLOB SUB_TYPE 0 SEGMENT SIZE 1024	Содержимое файла

¹ Допустимые значения поля FTSSMIME_TYPE приведены в таблице 2.3.

2 Хранимые процедуры полнотекстового поиска

Для управления индексами в полнотекстовом поиске используются следующие хранимые процедуры:

- FTSCREATE_INDEX;
- FT\$ADD_FIELD_TO_INDEX;
- FT\$APPLY_METADATA_CHANGES;
- FT\$REINDEX;
- FT\$FULL_REINDEX;
- FT\$STARTDAEMON;
- FT\$SEARCH;
- FT\$DROP_FIELD_FROM_INDEX;
- FT\$DROP_INDEX.

Процедура FTSCREATE_INDEX используется для создания индекса:

Таблица 2.1 - Входные параметры процедуры FTSCREATE_INDEX

Имя поля	Тип	Описание
FT\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса.
FT\$STORE	BLOB SUB_TYPE 1 SEGMENT_SIZE 80 CHARACTER SET UNICODE_FSS	Описание, где хранится индекс. Может быть NULL, тогда индекс будет храниться в таблице FT\$LUCENE_FILE_SYSTEM. Значение по умолчанию NULL. Если задано значение "file", то индексы будут сохраняться в файловой системе.
FT\$DESCRIPTION	BLOB SUB_TYPE 1 SEGMENT_SIZE 80 CHARACTER SET UNICODE_FSS	Комментарии к индексу. Значение по умолчанию NULL.

Процедура FT\$ADD_FILED_TO_INDEX добавляет индексируемое поле в индекс. Входные параметры процедуры FT\$ADD_FILED_TO_INDEX приведены в таблице 2.2.

Таблица 2.2 - Входные параметры процедуры FT\$ADD_FILED_TO_INDEX

Имя поля	Тип	Описание
FT\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса.
FT\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица.
FT\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле.
FT\$ANALIZER	CHAR(255) CHARACTER SET UNICODE_FSS	Имя анализатора. Может принимать значение NULL, в таком случае будет выбран анализатор Standard (English). Значение по умолчанию NULL. ²
FT\$MIME_TYPE	CHAR(127)	MIME тип указывает, для документа

² Допустимые значения имен анализаторов приведены в таблице 2.4.

Имя поля	Тип	Описание
	CHARACTER SET UNICODE_FSS	какого типа строиться индекс. Может принимать значение NULL. Доступные MIME-типы документов и их соответствие приложениям сведены в таблицу 2.3
FTS\$MIME_FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя поля для определение MIME формата документов. Может принимать значение NULL.

Возможные комбинации входных параметров рассмотрены ниже:

- Если значения параметров FTS\$MIME_TYPE и FTS\$MIME_FIELD_NAME равны NULL, то индекс строится не для документов.
- Если значение параметра FTS\$MIME_TYPE равно NULL, а параметра FTS\$MIME_FIELD_NAME не равно NULL, то индекс строится для документов. В этом случае, исходя из значения параметра FTS\$MIME_TYPE, определяется один MIME формат для всех документов, хранящихся в добавляемом поле.
- Если значение параметра FTS\$MIME_FIELD_NAME не равно NULL, а параметра FTS\$MIME_TYPE равно NULL, то индекс строится для документов. MIME формат документа, в этом случае, задается для каждой записи с помощью дополнительного поля, имя которого указано в качестве значения параметра FTS\$MIME_FIELD_NAME.

Таблица 2.3 - Соответствие MIME форматов типам документов

Тип документа	Указываемый MIME-формат документа
PDF	application/pdf
Microsoft Excel	application/vnd.ms-excel
Microsoft Word	application/msword
Microsoft PowerPoint	application/vnd.ms-powerpoint
RTF	application/rtf
Open Office Writer (ODT)	application/vnd.oasis.opendocument.text
HTML	text/html

Значение входного параметра FTS\$ANALIZER определяет, какой тип анализатора будет использоваться при индексации добавляемого поля:

Таблица 2.4 - Соответствие имен анализаторов и языков

Имя анализатора	Язык
English	Английский
Standard	Английский
Russian	Русский
German	Немецкий
French	Французский
Czech	Чешский
Brazilian	Бразильский
Chinese	Китайский
Dutch	Голландский
Greek	Греческий
CJK	Китайское письмо

Внимание! Корректность результатов поиска напрямую зависит от типа выбранного анализатора

Для подтверждения изменения метаданных необходимо выполнять процедуру FTS\$APPLAY_METADATA_CHANGES. Входные и выходные параметры у процедуры отсутствуют. Данную процедуру необходимо вызывать после изменения состава индекса (добавления либо удаления полей из индекса).

Служебная процедура FTSS\$REINDEX производит полную (по всем записям, с заменой ранее существующего индекса) переиндексацию указанного индекса. Процедура имеет один входной параметр – имя индекса (Таблица 2.5).

Таблица 2.5 - Входной параметр процедуры FTSS\$REINDEX

Имя поля	Тип	Описание
FTSS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса.

Для выполнения полной переиндексации для всех индексов, созданных в БД, выполняется процедурой FTSS\$FULL_REINDEX. Процедура не имеет входных и выходных параметров.

Для того, чтобы при изменении индексируемых наборов данных переиндексация выполнялась автоматически можно запустить «демон» переиндексации. Запуск демона выполняется процедурой FTSS\$STARTDAEMON. «Демон» выполняет непрерывный (каждые 0,1 с) мониторинг таблицы FTSS\$POOL и переиндексацию измененных записей, после чего соответствующие записи удаляются из FTSS\$POOL. Процедура FTSS\$STARTDAEMON не имеет входных и выходных параметров.

Для извлечения данных из индекса используется процедура FTSS\$SEARCH. Входные параметры процедуры приведены в таблице 2.6. Обязательными входными параметрами являются имя индекса, по которому будет осуществляться поиск, и условие поиска.

Таблица 2.6 - Входные параметры процедуры FTSS\$SEARCH

Имя поля	Тип	Описание
FTSS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса.
FTSS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индексируемой таблицы. Может принимать значение NULL, тогда поиск будет идти по всем таблицам, входящим в индекс.
FTSS\$FILTER	VARCHAR(4000) CHARACTER SET UNICODE_FSS	Фильтр, по которому будет осуществляться поиск (См. http://lucene.apache.org/java/2_3_1/queryparsersyntax.html).
FRAGMENT_SIZE	INTEGER	Параметр, задающий отображаемое количество символов фрагмента текста, в котором находится строка, соответствующая условиям поиска. По умолчанию используется значение, равное 50.

Процедура FTSS\$SEARCH имеет следующие выходные параметры: значение RDB\$DB_KEY для найденных записей, значение соответствия найденной записи условию поиска, имя таблицы, в которой найдены данные. Описание выходных параметров приведено в таблице 2.7.

Таблица 2.7 - Выходные параметры процедуры FTSS\$SEARCH

Имя поля	Тип	Описание
RELATION	VARCHAR(512)	Таблица, в которой найдено поле, удовлетворяющее фильтру
ROW_ID	CHAR(8) CHARACTER SET OCTETS	Значение RDB\$DB_KEY для таблицы, содержащей запись.
SCOPE	DOUBLE PRECISION	Оценка соответствия возвращаемой записи условию поиска.

Имя поля	Тип	Описание
HIGHLIGHT	VARCHAR(512)	Фрагмент текста, содержащий строку, удовлетворяющей условиям поиска. Найденная строка заключается в теги .

Процедура FTSDROP_FILED_FROM_INDEX используется для удаления индексируемого поля из индекса. Входные параметры приведены в таблице 2.8

Таблица 2.8 - Входные параметры процедуры FTSDROP_FILED_FROM_INDEX

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса.
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица.
FTS\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле.

Для удаления индекса из системы полнотекстового поиска используется процедура FTSDROP_INDEX. Описание входного параметра процедуры приведено в таблице 2.9.

Таблица 2.9 - Входные параметры процедуры FTSDROP_FILED_FROM_INDEX

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHARACTER SET UNICODE_FSS	Имя индекса.

3 Пример использования полнотекстового поиска

В общем случае протокол использования системы полнотекстового поиска можно представить следующим образом:

1. создание индекса;
2. добавление/удаление полей в индекс;
3. обновление метаданных системы полнотекстового поиска;
4. выполнение переиндексации;
5. поиск;
6. удаление индекса (если необходимо).

3.1 Создание индекса

Сначала необходимо создать индекс. Для этого следует выполнить процедуру FTSCREATE_INDEX, указав необходимые входные параметры. Обязательным входным параметром является только имя индекса. Второй входной параметр служит для указания на то, где должен храниться индекс. Он может принимать значение NULL или "file". По умолчанию используется значение NULL, в этом случае индекс будет храниться в таблице FTSLUCENE_FILE_SYSTEM. Если задано значение "file", то индексы будут сохраняться в файловой системе. Значение третьего параметра задает описание для индекса.

Пример создания индекса:

```
EXECUTE PROCEDURE FTSCREATE_INDEX ('SIMPLE_INDEX',  
'file', 'Simple index for FTS')
```

При этом в таблицу FT\$INDICES добавится запись. Значение поля FT\$INDEX_STATUS равно 'N', что означает, что индекс только что создан, требует полной переиндексации.

3.2 Удаление индекса

Для удаления индекса используется процедура FTSDROP_INDEX, которая имеет один обязательный входной параметр – имя индекса.

Пример удаления индекса:

```
EXECUTE PROCEDURE FTSDROP_INDEX ('SIMPLE_INDEX');
```

При удалении индекса удаляются:

- Соответствующая запись из таблицы FT\$INDICES;
- Связанные с индексом записи из таблицы FT\$INDEX_SEGMENTS;
- Связанные с индексом записи из таблицы FTSLUCENE_FILE_SYSTEM;
- Файлы индекса на диске.

3.3 Добавление полей в индекс

После того как индекс создан, в него можно добавлять поля из таблиц базы

данных. Для добавления поля в индекс используется процедура FTSS\$ADD_FIELD_TO_INDEX.

Два последних параметра процедуры FTSS\$ADD_FIELD_TO_INDEX и используются, если индекс формируется для документов, хранящихся в бинарных BLOB-полях. Значение параметра FTSS\$MIME_TYPE задает единый MIME формат документа для всех записей в индексируемом поле, а значение параметра FTSS\$MIME_FIELD_NAME имя поля в таблице, значения которого будут определять MIME формат для каждой записи в индексируемом поле.

Процедура имеет следующие обязательные входные параметры: имя индекса, имя таблицы, имя поля. Четвертый параметр позволяет задать тип анализатора. Доступные типы анализаторов приведены в таблице 9. По умолчанию используется значение NULL. В этом случае используется анализатор Standard (English).

Пример добавления поля в индекс(индекс строится не для документов):

```
EXECUTE PROCEDURE FTSS$ADD_FIELD_TO_INDEX  
( 'SIMPLE_INDEX', 'TABLE1', 'FIELD1', NULL, NULL, NULL)
```

Пример добавления поля в индекс(индекс строится для документов, все документы одного типа):

```
EXECUTE PROCEDURE FTSS$ADD_FIELD_TO_INDEX  
( 'SIMPLE_INDEX', 'TABLE1', 'FIELD1', 'English',  
'application/pdf', NULL)
```

После добавления поля в индекс в таблице FTSS\$INDEX_SEGMENTS должна появиться соответствующая запись.

3.4 MIME формат документа

Если индекс формируется для документов, то в процедуре добавления поля в индекс требуется указать MIME формат документа.

В системе полнотекстового поиска «Ред База Данных» при индексации документов можно указать один из семи MIME-форматов. Соответствие типов MIME-документов значениям, указываемым в поле FTSS\$MIME_TYPE, приведено в таблице 2.3.

3.5 Удаление полей из индекса

Для удаления полей из индекса в системе полнотекстового поиска используется процедура FTSS\$DROP_FIELD_FROM_INDEX. Процедура имеет три обязательных входных параметра: имя индекса, из состава которого удаляется поле; имя таблицы, которая содержит это поле; имя удаляемого поля. Пример удаления поля из состава индекса приведен ниже:

```
EXECUTE PROCEDURE FTSS$DROP_FIELD_FROM_INDEX  
( 'SIMPLE_INDEX', 'TABLE1', 'FIELD1')
```

После удаления поля из состава из таблицы FTSS\$INDEX_SEGMENTS удалится запись, связывающая поле и индекс.

3.6 Обновление метаданных системы полнотекстового поиска

После изменения структуры индекса, в частности после удаления поля из

его состава, нужно обновить метаданные системы полнотекстового поиска. Для этого предназначена процедура FTSS\$APPLY_METADATA_CHANGES. Процедура FTSS\$APPLY_METADATA_CHANGES не имеет ни входных, ни выходных параметров. Пример использования приведен ниже:

```
EXECUTE PROCEDURE FTSS$APPLY_METADATA_CHANGES
```

Процедура FTSS\$APPLY_METADATA_CHANGES выполняет удаление неиспользуемых триггеров, которые создаются при добавлении полей в индекс и остаются после их удаления из индекса, либо удаления индекса.

3.7 Переиндексация

Для того чтобы обновить данные индекса следует выполнить переиндексацию. Переиндексация может быть выполнена вызовом одной из трех процедур:

- FTSS\$FULL_REINDEX
- FTSS\$REINDEX
- FTSS\$STARTDAEMON

Процедура FTSS\$FULL_REINDEX выполняет полную переиндексацию для всех индексов в системе полнотекстового поиска. Процедура FTSS\$FULL_REINDEX не имеет входных параметров. Пример вызова процедуры:

```
EXECUTE PROCEDURE FTSS$FULL_REINDEX
```

Процедура FTSS\$REINDEX позволяет выполнить полную переиндексацию по указанному индексу. Процедура имеет один обязательный входной параметр – имя индекса, например:

```
EXECUTE PROCEDURE FTSS$REINDEX ('SIMPLE_INDEX')
```

Процедура FTSS\$STARTDAEMON запускает «демона» переиндексации. «Демон» выполняет, каждые 0,1 секунды, мониторинг таблицы FTSS\$POOL. В таблице FTSS\$POOL сохраняются RDB\$DB_KEY изменившихся записей. По значению RDB\$DB_KEY «демон» выполняет переиндексацию только изменившихся записей, после чего RDB\$DB_KEY переиндексированных записей удаляется из таблицы FTSS\$POOL. «Демона» следует запускать в отдельном коннекте к базе. Пример запуска «демона» переиндексации:

```
EXECUTE PROCEDURE FTSS$STARTDAEMON
```

3.8 Поиск

Для извлечения данных из индекса в системе полнотекстового поиска используется процедура FTSS\$SEARCH.

Процедура имеет следующие входные параметры: имя индекса, имя таблицы, фраза поиска. Обязательными входными параметрами являются имя индекса, по которому будет выполняться поиск, и, собственно, фраза поиска. Второй входной параметр – имя индексируемой таблицы может принимать значение NULL, в этом случае поиск выполняется по всем таблицам, входящим в индекс. Если же указано имя индексируемой таблицы, то поиск будет выполняться только по этой таблице.

Выходные параметры процедуры: таблица, в которой найдены данные, удовлетворяющие условию поиска, значение RDB\$DB_KEY для найденных записей, значение соответствия найденной записи условию поиска, отображаемое ко-

личество символов результата поиска.

Пример поиска слова test только по таблице TABLE1 в индексе SIMPLE_INDEX, длина фрагмента результата поиска равна 100 символов:

```
SELECT * from FTS$SEARCH ('SIMPLE_INDEX', 'TABLE1',
'test', 100);
```

Допустим, что при поиске было найдено две записи, в этом случае результат поиска для приведенного выше примера может иметь вид:

ROW_ID	SCOPE	RELATION	HIGHLIGHT
85000000E4010000	0,298344343900681	TABLE1	test text
85000000E5010000	0,298344343900681	TABLE1	test text

Здесь ROW_ID это значение RDB\$DB_KEY найденной записи; SCOPE – соответствие найденной записи условию поиска; RELATION – имя таблицы, в которой была найдена запись; HIGHLIGHT – фрагмент текста, содержащий строку, удовлетворяющую условиям поиска.

По значению RDB\$DB_KEY можно выбрать непосредственно найденные записи из соответствующих таблиц, например:

```
SELECT B.* from FTS$SEARCH ('SIMPLE_INDEX', 'TABLE1',
'test') as A LEFT JOIN TABLE1 as B on A.FTS$DB_KEY =
B.RDB$DB_KEY
```

Результатом такого запроса в отличие от предыдущего примера будет выборка из таблицы TABLE1, например:

F_ID	F_VCHAR
3	test text
6	test text

3.9 Синтаксис запросов поиска

Использование в термах символов “?” и “*” позволяет выполнять поиск по маске. В этом случае символ “?” заменяет один любой символ, а “*” – любое количество символов, например:

```
te?t test* tes*t
```

Примечание: Фразу поиска нельзя начинать с символов “?” или “*”.

Для выполнения нечеткого поиска в конец термина следует добавить тильду “~”. В этом случае будут искажаться все похожие слова, например при поиске “roam~” будут так же найдены слова “foam” и “goams”.

Lucene позволяет изменять значимость термов во фразе поиска. Например, вы ищете фразу “Hello world” и хотите, чтобы слово “world” было более значимым. Значимость термина во фразе поиска можно увеличить, используя символ “^”, после которого указывается коэффициент усиления. В следующем примере значимость слова “world” в четыре раза больше значимости слова “Hello”, которая по умолчанию равна единице.

```
"Hello world^4"
```

Булевы операторы позволяют использовать логические конструкции при задании условий поиска. Lucene поддерживает следующие булевы операторы: AND, “+”, OR, NOT, “-”.

Примечание: Булевы операторы должны указываться заглавными буквами.

3.9.1 Оператор OR.

OR является булевым оператором по умолчанию, это означает, что если между двумя терминами фразы поиска не указан другой булев оператор, то подставляется оператор OR. При этом система поиска находит документ, если одна из указанных во фразе поиска терм в нем присутствует. Альтернативным обозначением оператора OR является “||”.

```
"Hello world" "world"
```

Эквивалентно:

```
"Hello world" OR "world"
```

3.9.2 Оператор AND.

Оператор AND указывает на то, что в тексте должны присутствовать все, объединенные оператором термы поиска. Альтернативным обозначением оператора является “&&”.

```
"Hello" AND "world"
```

3.9.3 Оператор “+”.

Оператор “+” указывает на то, что следующее за ним слово должно обязательно присутствовать в тексте. Например, для поиска записей, которые должны содержать слово “hello” и могут содержать слово “world”, фраза поиска может иметь вид:

```
+Hello world
```

3.9.4 Оператор NOT.

Оператор NOT позволяет исключить из результатов поиска те, в которых встречается терм, следующий за оператором. Вместо слова NOT может использоваться символ “!”. Например, для поиска записей, которые должны содержать слово “hello” и не должны содержать слово “world”, фраза поиска может иметь вид:

```
"Hello" NOT "world"
```

Примечание: Оператор NOT не может использоваться только с одним термом, например поиск с таким условием не вернет результатов, например:

```
NOT 'world'
```

3.9.5 Оператор “-”.

Этот оператор является аналогичным оператору NOT. Пример использования:

```
"Hello" -"world"
```

3.9.6 Группировка булевых операторов.

Анализатор запросов Lucene поддерживает группировку булевых операторов. Допустим, требуется найти либо слово “word” либо слово “dolly” и обязательно слово “hello”, для этого используется такой запрос:

```
"Hello"&&("world"||"dolly")
```

Экранирование специальных символов.

Для включения специальных символов во фразу поиска выполняется их экранирование обратным слешем "\". Ниже приведен список специальных символов, используемых в Lucene на данный момент:

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

Фраза поиска для выражения (1 +1): 2 будет иметь вид:

```
\ (1 \ +1 \) \: 2
```

Более подробное англоязычное описание синтаксиса расположено на официальном сайте Lucene:

http://lucene.apache.org/java/2_3_1/queryparsersyntax.html.