Новые возможности Firebird 5.0

Дмитрий Еманов
dimitr@firebirdsql.org

Firebird Project
https://www.firebirdsql.org/

РЕД СОФТ
https://www.red-soft.ru/

Текущий статус

v5.0 Beta 1

- Выпущен в марте 2023
- Доступен для Windows / Linux / MacOS / Android
- Около десятка новых фич, более 50 улучшений
- Исправлено более 30 ошибок
- Выложена бета-версия Firebird 5.0 Language Reference

Новые возможности v5.0 Beta 1

Самое важное

- Параллельные (многопоточные) бекап/рестор, свип и создание индексов
- Поддержка частичных индексов
- `SKIP LOCKED` выражение для команд `SELECT WITH LOCK / UPDATE / DELETE`
- Поддержка множества записей в выражении `RETURNING`
- Возможность минорного обновления ODS без бекап-рестора
- Профайлер SQL/PSQL
- Более эффективное сжатие длинных строк
- Кеш компилированных запросов
- Новые встроенные функции и пакеты

Совместимость и миграция

Особенности

- Новая минорная ODS 13.1
- Ядро может работать с базами в ODS 13.0 (созданными в Firebird 4.0), но некоторые фичи будут недоступны
- Единственная несовместимость с Firebird 4.0 — новое поведение `RETURNING`
- Удалены утилита QLI и поддержка протокола WNET (он же NamedPipes, он же NetBEUI)

Совместимость и миграция

Особенности

- Новая минорная ODS 13.1
- Ядро может работать с базами в ODS 13.0 (созданными в Firebird 4.0), но некоторые фичи будут недоступны
- Единственная несовместимость с Firebird 4.0 — новое поведение `RETURNING`
- Удалены утилита QLI и поддержка протокола WNET (он же NamedPipes, он же NetBEUI)

Миграция

- С любой версии — backup/restore через `gbak`
- С Firebird 4.0 — через `gfix -upgrade` (опционально)

Совместимость и миграция

Обновление ODS «на месте»

- Новая команда: `gfix -upgrade <database>`
- Доступна SYSDBA, владельцу БД или пользователю с привилегией `USE_GFIX_UTILITY`
- Работает только для минорного апгрейда ODS (13.0 → 13.1 и т. п.), в этих случаях можно использовать вместо традиционного цикла backup/restore
- Требует эксклюзивного доступа к базе данных
- Полностью транзакционна, при любой ошибке состояние ODS базы данных остается прежним
- Даунгрейд через backup/restore

Многопоточные операции

В ядре Firebird

- Поддержка концепции worker thread / worker attachment
- Параметры конфига `ParallelWorkers` и `MaxParallelWorkers`, управляющие рабочими потоками
- Используется пока только для свипа (как ручного, так и автоматического) и для команды `CREATE INDEX`
- `ParallelWorkers` можно переопределить через параметр коннекта `isc_dpb_parallel_workers`
- `gfix -sweep -parallel 4 <database>`
- В будущем будет использоваться и другими операциями

Многопоточные операции

В утилите gbak

- Поддержка многопоточности и чтения/записи в нескольких коннектах
- Бекап использует концепцию разделяемого снапшота транзакции (`fb_info_tra_snapshot_number`, `isc_tpb_at_snapshot_number`) для получения целостного образа данных из разных параллельных коннектов
- Параллельные чтение/запись управляются через параметр `-parallel`, он же передается в ядро через `isc_dpb_parallel_workers` для параллельного создания индексов с теми же параметрами
- `gbak -b -parallel 4 <dbname> <backupname>`
  `gbak -r -parallel 4 <backupname> <dbname>`

Поддержка частичных индексов

Синтаксис

```
CREATE [UNIQUE] [{ASC[ENDING] | DESC[ENDING]}]
INDEX <index_name> ON <table_name>
{ (<column_list>) | COMPUTED [BY] ( <value_expression> ) }
WHERE <search_condition>
```

Семантика

- Индексирует заданное подмножество записей
- Позволяет сделать это подмножество уникальным
- Условие индекса должно совпадать с условием в WHERE/ON-предикате (*)
- Индексируемые поля не обязаны присутствовать в условии индекса

Выражение SKIP LOCKED

Синтаксис

- ```
  SELECT … WITH LOCK SKIP LOCKED
  UPDATE … SKIP LOCKED
  DELETE … SKIP LOCKED
  ```

Семантика

- Позволяет пропускать при чтении записи, заблокированные другими коннектами
- Полезно для организации параллельных очередей обработки
- FIRST-фильтры работают после блокировки, как и прежде
  ```

Курсоры в RETURNING

Синтаксис

- `INSERT … SELECT, UPDATE, DELETE, MERGE, UPDATE OR INSERT RETURNING <что-то-там>`

Семантика

- Возвращают курсор вместо одной записи
- Описываются в API как `isc_info_sql_stmt_select` вместо `isc_info_sql_stmt_exec_procedure`
- `INSERT … VALUES (), UPDATE/DELETE … WHERE CURRENT OF` работают по старым правилам
- Внутри PSQL все также работает по старым правилам

Улучшенное RLE-сжатие записей

Проблема

- Неэффективное сжатие длинных последовательностей, особенно актуально для длинных UTF8-строк
- Степень сжатия не более 64x (каждые 128 байт в 2 байта)
- Большое количество блоков «сжатый-несжатый» тормозит распаковку

Улучшенное RLE-сжатие записей

Проблема

- Неэффективное сжатие длинных последовательностей, особенно актуально для длинных UTF8-строк
- Степень сжатия не более 64x (каждые 128 байт в 2 байта)
- Большое количество блоков «сжатый-несжатый» тормозит распаковку

Решение

- Счетчики длины блока переменной длины → теперь любую последовательность одинаковых байт сжимаем в один блок
- Не пытаемся сжимать короткие последовательности
- Добавили возможность хранить несжатые записи

Профайлер SQL / PSQL

Архитектура

- Системный пакет RDB$PROFILER, который собирает данные и передает их в плагин
- Есть плагин по умолчанию, доступен «из коробки»
- Работает как для SQL-запросов, так и для PSQL-кода
- Собирает число вызовов каждой команды, ведет статистику минимального, максимального и общего времени выполнения команды
- Для явных и неявных курсоров собирает статистику по операциям открытия курсора и фетчей из него, а также умеет собирать статистику в разрезе узлов плана выполнения
- Требует наличия привилегии `PROFILE_ANY_ATTACHMENT`

Профайлер SQL / PSQL

Использование

```
rdb$profiler.start_session('Profile Session 1');


set term !;
execute block
as
begin
  execute procedure ins;
  delete from tab;
end!
set term ;!


rdb$profiler.finish_session(true);
```

Профайлер SQL / PSQL

Анализ

```
select * from plg$prof_sessions;
select * from plg$prof_psql_stats_view;
select * from plg$prof_record_source_stats_view;
select preq.*
  from plg$prof_requests preq
  join plg$prof_sessions pses
    on pses.profile_id = preq.profile_id and
       pses.description = 'Profile Session 1';

и т.п.
```

После Beta 1

Реализовано на текущий момент

- Добавлена поддержка QUARTER в функции EXTRACT / FIRST_DAY / LAST_DAY
- Актуальное число параллельных воркеров вынесено в MON$ATTACHMENTS, а также доступно через контекстную переменную SYSTEM.PARALLEL_WORKERS и через API
- Команда SHOW DATABASE в ISQL теперь выводит статус реплики и статус публикации, команда SHOW TABLE также показывает участие таблицы в репликации
- Улучшена (на порядок) скорость создания вычисляемых и частичных индексов
- Оптимизировано формирование грантов при извлечении метаданных в ISQL

После Beta 1

Пул-риквесты в очереди

- Трассировка событий COMPILE для процедур/функций/триггеров с планами выполнения их внутренних запросов и временем парсинга
- Рекурсивное отображение планов процедур
- Поддержка именованных аргументов при вызове хранимых процедур/функций
- Снятие лимита в 64КБ на длину записи

После Beta 1

Пул-риквесты в очереди

- Трассировка событий COMPILE для процедур/функций/триггеров с планами выполнения их внутренних запросов и временем парсинга
- Рекурсивное отображение планов процедур
- Поддержка именованных аргументов при вызове хранимых процедур/функций
- Снятие лимита в 64КБ на длину записи

Что дальше

- Release Candidate 1

Трассировка COMPILE-событий

```
2022-12-15T12:18:34.0900 (2236042:0x7ffff024a050) COMPILE_PROCEDURE
    /data/tpc-c/tpcc-fb50.fdb (ATT_28, SYSDBA:NONE, NONE, TCPv4:127.0.0.1/60476)
    /work/firebird/gen/Debug/firebird/bin/isql:2236126

Procedure NEWORD1:
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

-- line 23, column 2
PLAN (DISTRICT INDEX (DISTRICT_PK))

-- line 28, column 2
PLAN JOIN (CUSTOMER INDEX (CUSTOMER_PK), WAREHOUSE INDEX(WAREHOUSE_PK))

      3 ms
```

Трассировка COMPILE-событий

```
2022-12-15T12:18:34.0900 (2236042:0x7ffff024a050) COMPILE_PROCEDURE
    /data/tpc-c/tpcc-fb50.fdb (ATT_28, SYSDBA:NONE, NONE, TCPv4:127.0.0.1/60476)
    /work/firebird/gen/Debug/firebird/bin/isql:2236126

Procedure NEWORD1:
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Select Expression at line 23, column 2
    -> Singularity Check
        -> Filter
            -> Table "DISTRICT" Access By ID
                -> Bitmap
                    -> Index "DISTRICT_PK" Unique Scan

Select Expression at line 28, column 2
    -> Singularity Check
        -> Nested Loop Join (inner)
            -> Filter
                -> Table "CUSTOMER" Access By ID
                    -> Bitmap
                        -> Index "CUSTOMER_PK" Unique Scan
            -> Filter
                -> Table "WAREHOUSE" Access By ID
                    -> Bitmap
                        -> Index "WAREHOUSE_PK" Unique Scan

      3 ms
```

Рекурсивный вывод планов

```
create or alter procedure p1 returns (a int)
as
begin
  select null from rdb$database into :a;
  suspend;

  for select rdb$relation_id from rdb$relations into :a do suspend;
end^

create or alter procedure p2 returns (a int)
as
begin
  select null from rdb$database into :a;
  suspend;

  for select rdb$relation_id from rdb$relations into :a do suspend;

  for select a from p1 into :a do suspend;
end^
```

Рекурсивный вывод планов

```
SQL> select * from p1;

Select Expression
    -> Procedure "P1" Scan
        -> Select Expression
            -> Singularity Check
                -> Table "RDB$DATABASE" Full Scan
        -> Select Expression
            -> Table "RDB$RELATIONS" Full Scan

SQL> select * from p2;

Select Expression
   -> Procedure "P2" Scan
        -> Select Expression
            -> Singularity Check
                -> Table "RDB$DATABASE" Full Scan
        -> Select Expression
            -> Table "RDB$RELATIONS" Full Scan
        -> Select Expression
            -> Procedure "P1" Scan
                -> Select Expression
                    -> Singularity Check
                        -> Table "RDB$DATABASE" Full Scan
                -> Select Expression
                    -> Table "RDB$RELATIONS" Full Scan
```

Именованные аргументы при вызове PSQL

```
select function_name(parameter2 => 'Two', parameter1 => 1)
from rdb$database;

select function_name(1, parameter2 => 'Two')
from rdb$database;

execute procedure insert_customer(
  last_name => 'SCHUMACHER',
  first_name => 'MICHAEL');

select *
from get_customers(city_id => 10, last_name => 'SCHUMACHER');
```

Firebird 6.0

Надо доделать

- Команда `TRUNCATE TABLE`
- Хранимая статистика оптимизатора, гистограммы

В работе

- Общий кеш метаданных в суперсервере

Может быть портировано

- Поддержка `JSON`-функций и `JSON_TABLE` по SQL-стандарту
- Табличные пространства
- `fb_repl_print` — текстовый вывод журналов репликации

Firebird 6.0

Топ-10 голосования в трекере

1. Job / Task Scheduler
2. Local temporary tables
3. Database Links
4. Add support for INTERSECT and EXCEPT data set operators
5. Create Table as Select ....
6. GIS implementation (opengis)
7. Full-Text indexing
8. Support native JSON datatype for columns as MySQL / PostgreSql
9. Add support for SQL Schemas
10. New database object - Constants

Вопросы?