

Новые возможности языка SQL в Firebird



Вспомним Firebird 2.1

- **Общий SQL**
 - COMMON TABLE EXPRESSIONS
 - INSERT OR UPDATE
 - MERGE
 - RETURNING
 - Встроенные функции
- **Процедурный SQL**
 - Домены в процедурах и триггерах
 - *TYPE OF <domain name>*
- **DDL**
 - DATABASE TRIGGER's
 - *CONNECT | DISCONNECT*
 - *TRANSACTION START | COMMIT | ROLLBACK*
 - GLOBAL TEMPORARY TABLE
- **Мониторинг**
 - Таблицы мониторинга
 - Прерывание пользовательского запроса



Что нового в Firebird 2.5

- **Общий SQL**

- SIMILAR TO
- SQLSTATE
- 16-тиричные константы
- Преобразования UUID между двоичным и строковым представлениями

- **Процедурный SQL**

- Автономные транзакции
- Новые возможности EXECUTE STATEMENT
- TYPE OF COLUMN

- **DDL**

- ALTER VIEW
- ALTER computed fields
- CREATE\ALTER\DROP user
- ALTER ROLE
- GRANTED BY

- **Мониторинг**

- Новые таблицы мониторинга
- Прерывание пользовательского соединения



Общий SQL : SIMILAR TO

Поддержка регулярных выражений согласно стандарту SQL

Новый предикат **SIMILAR TO**

Более мощная версия **LIKE** с синтаксисом регулярных выражений

Пример : является ли строка числом ?

```
Value SIMILAR TO '[\+|-]?[0-9]*([0-9].|. [0-9])?[[[:DIGIT:]]*'  
        ESCAPE '\'
```

<code>[\+ -]</code>	+ или -
<code>?</code>	0 или 1 раз
<code>[0-9]</code>	любая цифра
<code>*</code>	0 или много раз
<code>([0-9]. . [0-9])</code>	<цифра и точка> или <точка и цифра>
<code>?</code>	0 или 1 раз
<code>[[[:DIGIT:]]</code>	любая цифра
<code>*</code>	0 или много раз



Общий SQL : SQLSTATE

SQLSTATE стандартный код ошибки, удобен для универсальных приложений

SQLCODE устарел (но поддерживается). Рекомендуем пользоваться **SQLSTATE**

Для получения SQLSTATE используйте новую функцию API : **fb_sqlstate**

В Firebird 2.5, isql показывает SQLSTATE в сообщениях об ошибках :

```
FB25>isql
```

```
SQL> connect 'not_exists';  
Statement failed, SQLSTATE = 08001  
I/O error during "CreateFile (open)" operation for file "not_exists"  
-Error while trying to open file  
-The system cannot find the file specified.
```

```
FB21>isql
```

```
SQL> connect 'not_exists';  
Statement failed, SQLCODE = -902  
I/O error for file "...\\not_exists"  
-Error while trying to open file  
-The system cannot find the file specified.
```



Общий SQL : HEX константы

Шестнадцатиричные числовые и строковые константы

Числовые константы : *0xNNNNNNNN*

Префикс *0x* или *0X*

До 16 шестнадцатиричных цифр

Если задано от 1 до 8 цифр, тип константы - *signed integer*

Если задано от 9 до 16 цифр, тип константы - *signed bigint*

```
SQL> SELECT 0xF0000000, 0x0F000000 FROM RDB$DATABASE;
```

CONSTANT	CONSTANT
=====	=====
-268435456	4026531840



Общий SQL : HEX константы

Шестнадцатичные числовые и строковые константы

Строковые константы : *x'HH...H'*

Префикс *x* или *X*

Тип данных - **CHAR(N / 2) CHARACTER SET OCTETS**, где N - количество цифр в строке

```
SQL> SELECT 'First line' || _ASCII x'0D0A09' || 'Second line'  
CON> FROM RDB$DATABASE;
```

CONCATENATION

=====

First line

Second line



Общий SQL : UUID <-> CHAR

Преобразования UUID между двоичным и строковым видами

CHAR_TO_UUID : преобразует строковое (CHAR(32) ASCII) представление UUID
(XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX)
в двоичное (CHAR(16) OCTETS), более оптимизированное для хранения.

```
SQL> SELECT CHAR_TO_UUID('A96B285B-4629-45A1-9A86-A8ECCF6561F4')  
      FROM RDB$DATABASE;
```

```
CHAR_TO_UUID  
=====  
A96B285B462945A19A86A8ECCF6561F4
```



Общий SQL : UUID <-> CHAR

Преобразования UUID между двоичным и строковым видами

UUID_TO_CHAR

Преобразует двоичный (CHAR(16) OCTETS) UUID в строковый (CHAR(32) ASCII)

```
SQL> SELECT UUID_TO_CHAR(x' A96B285B462945A19A86A8ECCF6561F4' )  
      FROM RDB$DATABASE ;
```

```
UUID_TO_CHAR
```

```
=====
```

```
A96B285B-4629-45A1-9A86-A8ECCF6561F4
```



PSQL : автономные транзакции

Синтаксис

```
IN AUTONOMOUS TRANSACTION DO  
  <simple statement | compound statement>
```

Параметры транзакции ?

Такие же, как и у внешней транзакции (уровень изоляции, read\write, wait, и т.д.)
Настаивается ? Пока что нет... может быть позже

Как и когда завершается автономная тр-ция ?

```
if (statement executed ok)  
then commit  
else rollback
```

Примечания

Автономная транзакция и её внешняя транзакция полностью независимы и изолированы друг от друга, как и любые другие две транзакции.



PSQL : EXECUTE STATEMENT

- Полностью переписанный **EXECUTE STATEMENT** :
 - Запросы с параметрами
 - Выполнение с правами вызывающего PSQL объекта
 - Автономные транзакции
 - Запросы к другим БД Firebird
 - Полная обратная совместимость

- **Синтаксис**

```
[FOR] EXECUTE STATEMENT <query_text> [( <input_parameters> )]  
  [ON EXTERNAL [DATA SOURCE] <connection_string>]  
  [WITH AUTONOMOUS | COMMON TRANSACTION]  
  [AS USER <user_name>]  
  [PASSWORD <password>]  
  [ROLE <role_name>]  
  [WITH CALLER PRIVILEGES]  
  [INTO <variables>]
```



PSQL : EXECUTE STATEMENT

Запросы с параметрами

Именованные параметры

```
EXECUTE STATEMENT
```

```
( 'INSERT INTO TABLE VALUES (:a, :b, :a) '
```

```
(a := 100, b := CURRENT_CONNECTION)
```

Не именованные параметры

```
EXECUTE STATEMENT
```

```
( 'INSERT INTO TABLE VALUES (?, ?, ?) '
```

```
(100, CURRENT_CONNECTION, 100)
```



PSQL : EXECUTE STATEMENT

Права вызывающего объекта

```
-- logon as SYSDBA
CREATE TABLE A (ID INT);
CREATE USER VLAD PASSWORD 'vlad';

CREATE PROCEDURE P1 RETURNS (CNT INT)
AS
BEGIN
    EXECUTE STATEMENT 'SELECT COUNT(*) FROM A' INTO :CNT;
    SUSPEND;
END;

GRANT SELECT ON TABLE A TO PROCEDURE P1;
GRANT EXECUTE ON PROCEDURE P1 TO USER VLAD;

-- logon as VLAD
SELECT * FROM P1;

Statement failed, SQLSTATE = 42000
Execute statement error at jrd8_prepare :
335544352 : no permission for read/select access to TABLE A
Statement : SELECT COUNT(*) FROM A
Data source : Internal::
-At procedure 'P1'
```



PSQL : EXECUTE STATEMENT

Права вызывающего объекта

```
-- logon as SYSDBA
CREATE PROCEDURE P2 RETURNS (CNT INT)
AS
BEGIN
    EXECUTE STATEMENT 'SELECT COUNT(*) FROM A'
        WITH CALLER PRIVILEGES
        INTO :CNT;
    SUSPEND;
END;
```

```
GRANT SELECT ON TABLE A TO PROCEDURE P2;
GRANT EXECUTE ON PROCEDURE P2 TO USER VLAD;
```

```
-- logon as VLAD
SELECT * FROM P2;
```

```
      CNT
=====
      0
```

Динамический запрос выполняется с таким набором прав, как если бы он выполнялся непосредственно вызывающим PSQL объектом (процедурой или триггером)



PSQL : EXECUTE STATEMENT

Транзакции

Общая транзакция (по-умолчанию)

запрос выполняется в текущей транзакции

```
EXECUTE STATEMENT '...'  
WITH COMMON TRANSACTION
```

Автономная транзакция

запрос выполняется в отдельной транзакции

```
EXECUTE STATEMENT '...'  
WITH AUTONOMOUS TRANSACTION
```

параметры такие же, как и у текущей транзакции



PSQL : EXECUTE STATEMENT

Запросы к другим БД Firebird

EXTERNAL DATA SOURCE :

<connection_string> обычная строка коннекта Firebird

Запрос к другой БД, используя user\password

```
EXECUTE STATEMENT '...'  
  ON EXTERNAL DATA SOURCE 'host:path'  
  USER 'VLAD' PASSWORD 'dontKnow'
```

Если user\password не заданы

Trusted authentication (только для Windows) : имя учётной записи процесса Firebird будет именем пользователя на удалённой БД (если TA поддерживается на удалённой стороне)

```
EXECUTE STATEMENT '...'  
  ON EXTERNAL DATA SOURCE 'host:path'
```

CURRENT_USER будет именем пользователя на локальной БД

```
EXECUTE STATEMENT '...'  
  ON EXTERNAL DATA SOURCE 'path_to_the_current_database'
```



PSQL : TYPE OF COLUMN

Синтаксис

```
data_type ::= <builtin_data_type>
            | <domain_name>
            | TYPE OF <domain_name>
            | TYPE OF COLUMN <table_name>.<column_name>
```

Использование

- В параметрах процедур

```
CREATE PROCEDURE
  MY_PROC (IN_PARAM TYPE OF COLUMN <table_name>.<column_name>)
  RETURNS (OUT_PARAM TYPE OF COLUMN <table_name>.<column_name>)
```

- В объявлениях переменных

```
DECLARE VARIABLE VAR1 TYPE OF COLUMN <table_name>.<column_name>
```

- В приведениях типа

```
OUT_PARAM = CAST (VAR1 AS TYPE OF COLUMN <table_name>.<column_name>)
```



ALTER VIEW \ COMPUTED FIELD

Синтаксис

```
{CREATE OR ALTER | ALTER} VIEW <view_name> [( <field list> )]  
    AS <select statement>  
  
ALTER TABLE <table_name>  
    ALTER <field_name> [TYPE <data type>] COMPUTED BY (<expression>);
```

Назначение

изменить определение view (или computed field)

До Firebird 2.5

drop и create (или alter два раза) все зависимые объекты

восстановить все права, выданные ранее всем пересозданным объектам

В Firebird 2.5

просто выполните **ALTER VIEW** и всё :-)



ALTER VIEW \ COMPUTED FIELD

Пример

```
CREATE TABLE T (NAME CHAR(20), CF COMPUTED BY (LEFT(NAME, 10)) );
CREATE VIEW V AS SELECT CF FROM T;
CREATE PROCEDURE P ... SELECT ... FROM V ...; -- зависимая процедура
```

Изменим определение view и computed field

До Firebird 2.5 :

```
ALTER PROCEDURE P AS BEGIN END; -- пустое тело для ВСЕХ зависимых
DROP VIEW V;
ALTER TABLE T DROP CF;
ALTER TABLE T
  ADD CF COMPUTED BY (SUBSTRING(NAME FOR 15));
CREATE VIEW V AS
  SELECT NAME, CF FROM T;
GRANT SELECT ON TABLE T TO VIEW V; -- восстановить ВСЕ зависимые
ALTER PROCEDURE P ... SELECT ... FROM V ...; -- объекты и права
```

А если есть сотни зависимых объектов ?



ALTER VIEW \ COMPUTED FIELD

Пример

```
CREATE TABLE T (NAME CHAR(20), CF COMPUTED BY (LEFT(NAME, 10)) );  
CREATE VIEW V AS SELECT CF FROM T;  
CREATE PROCEDURE P ... SELECT ... FROM V ...; -- зависимая процедура
```

Изменим определение view и computed field

В Firebird 2.5 :

```
ALTER TABLE T  
  ALTER CF COMPUTED BY (RIGHT(NAME, 10));  
  
ALTER VIEW V AS  
  SELECT NAME, CF FROM T;
```



DDL : управление пользователями

Синтаксис

```
CREATE USER name PASSWORD 'password'  
  [FIRSTNAME 'firstname']  
  [MIDDLENAME 'middlename']  
  [LASTNAME 'lastname']
```

```
ALTER USER name PASSWORD 'password'  
  [FIRSTNAME 'firstname']  
  [MIDDLENAME 'middlename']  
  [LASTNAME 'lastname']
```

```
DROP USER name
```



DDL : SYSDBA и SYSADMIN

Firebird 2.1

CURRENT_USER вернёт SYSDBA
Соединение обладает полномочиями SYSDBA

Firebird 2.5

CURRENT_USER вернёт имя реальной учётной записи Domain\User

Auto-admin mapping (индивидуально в каждой БД)

```
ALTER ROLE RDB$ADMIN SET|DROP AUTO ADMIN MAPPING;
```

Соединение без указания роли или с указанием роли RDB\$ADMIN

Auto-admin mapping выключено (по умолчанию)
обычные полномочия пользователя

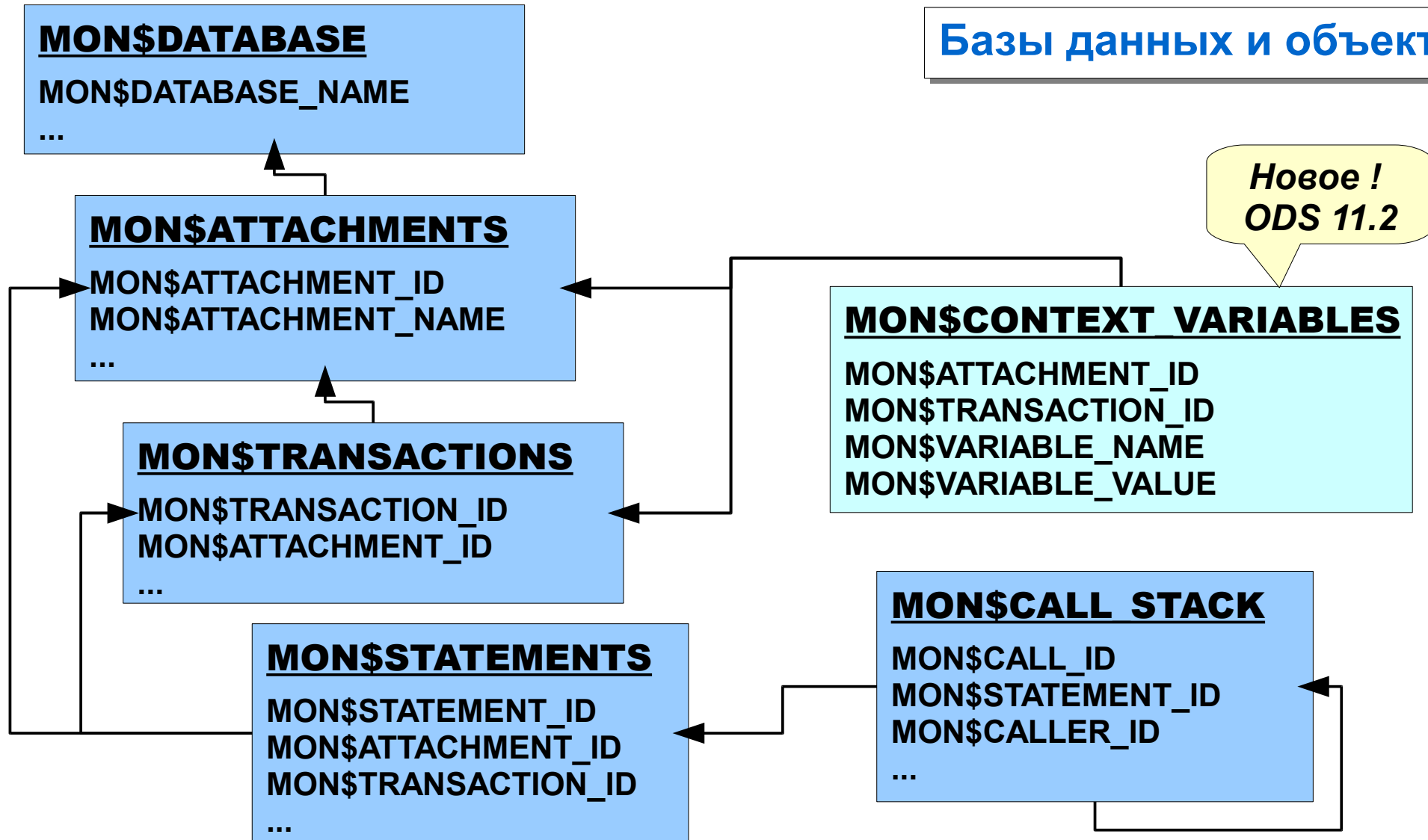
Auto-admin mapping включено
полномочия SYSDBA

Соединение с указанием роли не RDB\$ADMIN
обычные полномочия пользователя



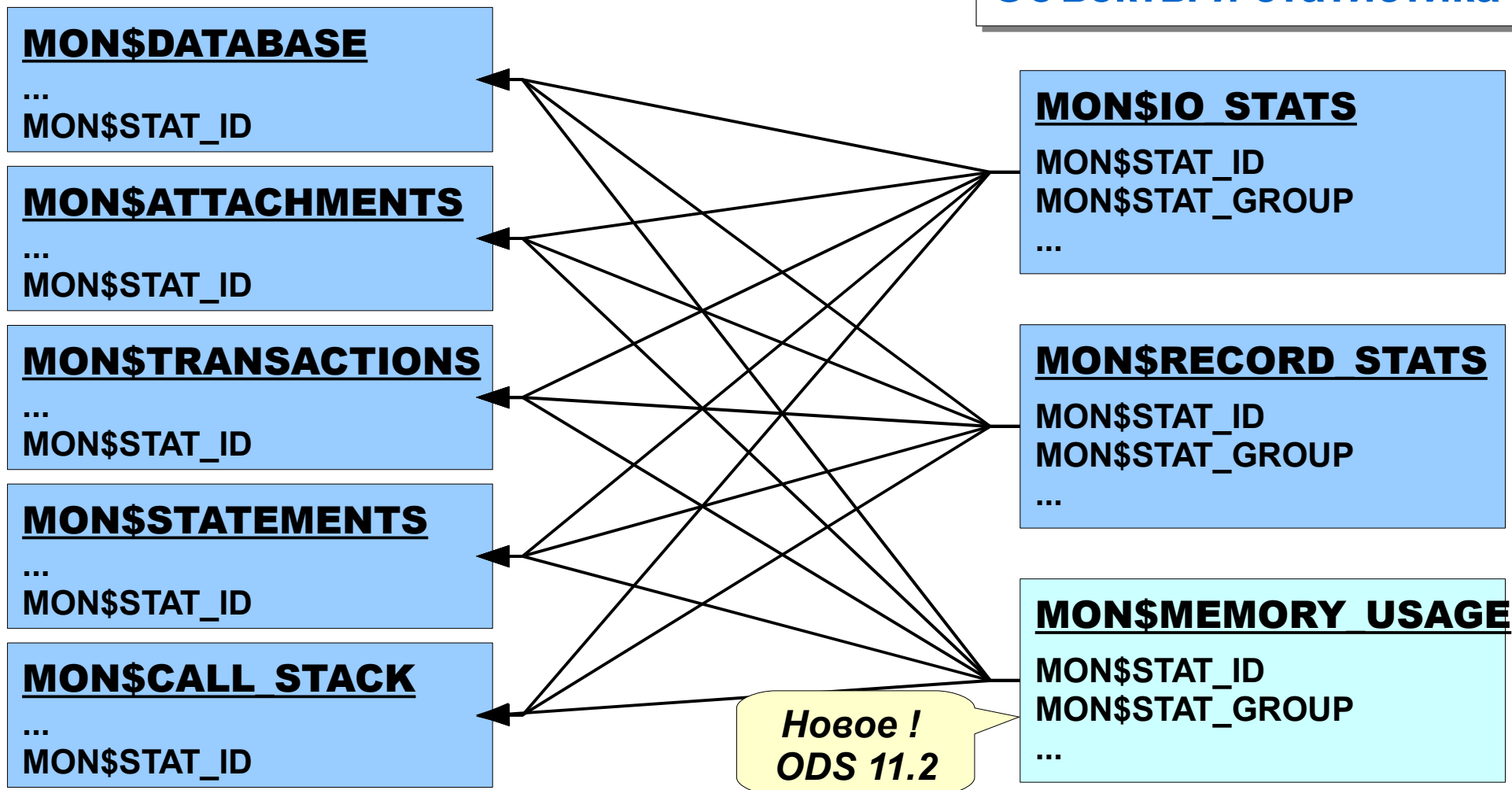
МОНИТОРИНГ

Базы данных и объекты



МОНИТОРИНГ

Объекты и статистика



ТАБЛИЦЫ МОНИТОРИНГА

«Активные» таблицы мониторинга

1) Остановить все запросы, выполняющиеся в соединении № 32:

```
DELETE FROM MON$STATEMENTS  
WHERE MON$ATTACHMENT_ID = 32
```

2) Остановить все запросы, выполняющиеся дольше 20 минут:

```
DELETE FROM MON$STATEMENTS  
WHERE DATEADD(20 MINUTE TO MON$TIMESTAMP) < CURRENT_TIMESTAMP  
AND MON$STATE <> 0;
```

3) Отключить все соединения, кроме своего (новое в Firebird 2.5):

```
DELETE FROM MON$ATTACHMENTS  
WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```



Это не всё !

А что дальше ?



Что нового в Firebird 3

- **Общий SQL**

- Полный синтаксис оператора **MERGE** (SQL 2008)
- **MERGE ... RETURNING**
- Оконные (аналитические) функции (window functions)
- **SUBSTRING** с регулярными выражениями

- **Процедурный SQL**

- Функции на SQL
- Пакеты процедур и функций : PSQL packages
- Внешние ф-ции, процедуры и триггеры на C\C++\Pascal\Java и т.п.
- Исключения с параметрами : **EXCEPTION ... USING (...)**
- **SQLSTATE** в обработчике **WHEN**
- Оператор **CONTINUE** в циклах



Что нового в Firebird 3

- **DDL**

- Управление возможностью хранить NULL значения в колонке

- **ALTER DOMAIN ... {NULL | NOT NULL}**

- **ALTER COLUMN ... {NULL | NOT NULL}**

- **ALTER DATABASE ... SET DEFAULT CHARACTER SET**

- **IDENTITY columns** (*привет мигрантам с MSSQL/MySQL ;)*

- DDL триггеры

- **Более полная система SQL полномочий**

- **GRANT CREATE | ALTER | DROP <object> TO <user> | <role>**

- **GRANT ROLE TO ROLE**

- **Мониторинг**

- Статистика, планы запросов, ...

- **To be continued :-)**



Общий SQL : MERGE

**Полный синтаксис, согласно SQL 2008.
Поддержка RETURNING.**

```
MERGE INTO <table>
  USING <table_or_join>
    ON <search_condition>
  [WHEN MATCHED [AND <search_condition>] THEN
    UPDATE SET col1 = val1, ..., colN = valN
    |
    DELETE]
  [WHEN NOT MATCHED [AND <search_condition>] THEN
    INSERT [(col1, ..., colN)] VALUES (val1, ..., valN)]
  [RETURNING ... [INTO ...]]
```



Общий SQL : WINDOW FUNCTIONS

Синтаксис

```
<window function> ::=  
    <window function type> OVER (<window specification>)
```

```
<window function type> ::=  
    <aggregate function>           -- агрегатные  
    | <rank function type>         -- ранжирующие  
    | ROW_NUMBER                   -- номер строки  
    | <lead or lag function>      -- навигационные
```

```
<aggregate function> ::=  
    AVG | MAX | MIN | SUM | COUNT
```

```
<rank function type> ::=  
    RANK | DENSE_RANK
```

```
<lead or lag function> ::=  
    LEAD | LAG
```



Общий SQL : WINDOW FUNCTIONS

Синтаксис

`<window specification> ::=`

`[PARTITION BY column1, ...]`

`[ORDER BY column1 [ASC|DESC] [NULLS {FIRST|LAST}], ...]`

Пример

`SUM(C) OVER ()`

`SUM(C) OVER (ORDER BY A, B)`

`SUM(C) OVER (PARTITION BY A)`

`SUM(C) OVER (PARTITION BY A, ORDER BY B)`

A	B	C	SUM	SUM1	SUM2	SUM3
1	1	30	141	30	60	30
1	2	20	141	50	60	50
1	3	10	141	60	60	60
2	1	25	141	85	40	25
2	2	15	141	100	40	40
3	1	41	141	141	41	41



Общий SQL : поиск подстроки по шаблону с REGEXP

Синтаксис

`SUBSTRING(<string> SIMILAR <pattern> ESCAPE <char>)`

Правила

- Шаблон

- $R = \langle R1 \rangle \langle E \rangle " \langle R2 \rangle \langle E \rangle " \langle R3 \rangle$

- Поиск

- $S = \langle S1 \rangle \langle S2 \rangle \langle S3 \rangle$

1) $\langle S \rangle \quad \text{SIMILAR TO } \langle R1 \rangle \langle R2 \rangle \langle R3 \rangle \text{ ESCAPE } \langle E \rangle$

2) $\langle S1 \rangle \quad \text{SIMILAR TO } \langle R1 \rangle \quad \text{ESCAPE } \langle E \rangle \text{ AND}$
 $\langle S2 \rangle \langle S3 \rangle \text{ SIMILAR TO } \langle R2 \rangle \langle R3 \rangle \quad \text{ESCAPE } \langle E \rangle$

3) $\langle S2 \rangle \quad \text{SIMILAR TO } \langle R2 \rangle \quad \text{ESCAPE } \langle E \rangle \text{ AND}$
 $\langle S3 \rangle \quad \text{SIMILAR TO } \langle R3 \rangle \quad \text{ESCAPE } \langle E \rangle$

- Результат

- $S2$



Общий SQL : поиск подстроки по шаблону с REGEXP

Синтаксис

```
SUBSTRING(<string> SIMILAR <pattern> ESCAPE <char>)
```

Пример

```
SUBSTRING('абв-12b34xyz' SIMILAR '%\["[\+|-]?[0-9]+\]' ESCAPE '\')
```

R1 = %

R2 = [\+|-]?[0-9]+

R3 = %

1) 'абв-12b34xyz' SIMILAR TO '%[\+|-]?[0-9]+%' ESCAPE '\'

2) 'абв' SIMILAR TO '%' ESCAPE '\'

'-12b34xyz' SIMILAR TO '[\+|-]?[0-9]+%' ESCAPE '\'

3) '-12' SIMILAR TO '[\+|-]?[0-9]+' ESCAPE '\'

'b34xyz' SIMILAR TO '%' ESCAPE '\'

Результат

'-12'



PSQL : SQL-функции

Синтаксис

```
{CREATE [OR ALTER] | ALTER | RECREATE} FUNCTION <name>
  [(param1 [, ...])]
  RETURNS <type>
AS
BEGIN
  ...
END
```

Пример

```
CREATE FUNCTION F(X INT) RETURNS INT
AS
BEGIN
  RETURN X+1;
END;

SELECT F(5) FROM RDB$DATABASE;
```



PSQL : Пакеты

```
-- заголовок пакета, только объявления
CREATE OR ALTER PACKAGE TEST
AS
BEGIN
    PROCEDURE P1(I INT) RETURNS (O INT);    -- публичная процедура
END

-- тело пакета, реализация
RECREATE PACKAGE BODY TEST
AS
BEGIN
    FUNCTION F1(I INT) RETURNS INT;        -- приватная функция

    PROCEDURE P1(I INT) RETURNS (O INT)
    AS
    BEGIN
    END;

    FUNCTION F1(I INT) RETURNS INT
    AS
    BEGIN
        RETURN 0;
    END;
END
```



PSQL : EXCEPTION с параметрами

Синтаксис

```
EXCEPTION <name> USING (param1 [, ...]);
```

Пример

```
CREATE EXCEPTION EX_BAD_SP_NAME
    'Имя процедуры должно начинаться с '@1' : '@2'';

CREATE TRIGGER TRG_SP_CREATE BEFORE CREATE PROCEDURE
AS
DECLARE SP_NAME VARCHAR(255);
BEGIN
    SP_NAME = RDB$GET_CONTEXT('DDL_TRIGGER', 'OBJECT_NAME');

    IF (SP_NAME NOT STARTING 'SP_')
    THEN EXCEPTION EX_BAD_SP_NAME USING ('SP_', SP_NAME);
END;
```



DDL : IDENTITY columns

Синтаксис

```
<column definition> ::=  
    <name> <type> GENERATED BY DEFAULT AS IDENTITY <constraints>
```

Правила

- Тип колонки - целое или **NUMERIC(P, 0)**
- Неявно задаётся **NOT NULL**
- Не гарантирует уникальности
- Не может иметь значения по умолчанию (**DEFAULT**)
- Не может быть вычисляемой (**COMPUTED BY**)



DDL : DDL триггеры

Синтаксис

`<database-trigger> ::=`

```
{CREATE | RECREATE | CREATE OR ALTER} TRIGGER <name>
  [ACTIVE | INACTIVE] {BEFORE | AFTER} <ddl event>
  [POSITION <n>]
```

`<ddl event> ::=`

```
ANY DDL STATEMENT
| <ddl event item> [{OR <ddl event item>}...]
```

`<ddl event item> ::=`

```
{CREATE | ALTER | DROP}
{TABLE | PROCEDURE | FUNCTION | TRIGGER | EXCEPTION |
VIEW | DOMAIN | SEQUENCE | INDEX | ROLE |
USER | COLLATION | PACKAGE | PACKAGE BODY |
CHARACTER SET }
```



DDL : DDL триггеры

Контекстные переменные (**RDB\$GET_CONTEXT**)

- Область имен **DDL_TRIGGER**
- Действительна только при вызове из DDL триггера
- Только чтение
- Предопределённые переменные:
 - **DDL_EVENT** - вид DDL события
 - **OBJECT_NAME** - имя объекта метаданных
 - **SQL_TEXT** - текст SQL запроса



СПАСИБО ЗА ВНИМАНИЕ

Вопросы ?

[Firebird official web site](#)

[Firebird tracker](#)

hvlad@users.sf.net

