



New SQL Features in Firebird 2.5



Whats New in Firebird 2.5



- **Common SQL**

- SIMILAR TO
- SQLSTATE
- Hexadecimal constants
- UUID binary to/from char conversions

- **Procedural SQL**

- AUTONOMOUS TRANSACTIONS
- EXECUTE STATEMENT
- TYPE OF COLUMN

- **DDL**

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

- **Monitoring**

- New MONITORING TABLES
- Terminate user connection





Regular expressions support per SQL standard specification

New **SIMILAR TO** predicate

Use **vertical bar** to match any of the few expressions :

```
'ab'   SIMILAR TO 'ab|cd|efg'  -- true
'efg'  SIMILAR TO 'ab|cd|efg'  -- true
'a'    SIMILAR TO 'ab|cd|efg'  -- false
```

Use **asterisk** to match **zero or more** occurrences of search string

```
' '     SIMILAR TO 'a*'  -- true
'a'     SIMILAR TO 'a*'  -- true
'aaa'   SIMILAR TO 'a*'  -- true
```

Use **plus sign** to match **one or more** occurrences of search string

```
' '     SIMILAR TO 'a+'  -- false
'a'     SIMILAR TO 'a+'  -- true
'aaa'   SIMILAR TO 'a+'  -- true
```

Use **question mark** to match **zero or one** occurrences of search string

```
' '     SIMILAR TO 'a?'  -- true
'a'     SIMILAR TO 'a?'  -- true
'aaa'   SIMILAR TO 'a?'  -- false
```



Common SQL : SIMILAR TO



Use **{N}** to match **exact N** occurrences of search string

```
' '      SIMILAR TO 'a{2}'  -- false
'a'      SIMILAR TO 'a{2}'  -- false
'aa'     SIMILAR TO 'a{2}'  -- true
'aaa'    SIMILAR TO 'a{2}'  -- false
```

Use **{N,}** to match **N or more** occurrences of search string

```
' '      SIMILAR TO 'a{2,}' -- false
'a'      SIMILAR TO 'a{2,}' -- false
'aa'     SIMILAR TO 'a{2,}' -- true
'aaa'    SIMILAR TO 'a{2,}' -- true
```

Use **{N, M}** to match **N to M** occurrences of search string

```
' '      SIMILAR TO 'a{2,4}' -- false
'a'      SIMILAR TO 'a{2,4}' -- false
'aa'     SIMILAR TO 'a{2,4}' -- true
'aaa'    SIMILAR TO 'a{2,4}' -- true
'aaaa'   SIMILAR TO 'a{2,4}' -- true
'aaaaa'  SIMILAR TO 'a{2,4}' -- false
```

Use **underscore** to match **any (non-empty) character**

```
' '      SIMILAR TO '_'     -- false
'a'      SIMILAR TO '_'     -- true
'1'      SIMILAR TO '_'     -- true
'a1'     SIMILAR TO '_'     -- false
```



Common SQL : SIMILAR TO



Use **percent** sign to match a string **of any length**, including empty strings

```
' '      SIMILAR TO '%'      -- true
'az'     SIMILAR TO 'a%z'    -- true
'a123z'  SIMILAR TO 'a%z'    -- true
'azx'    SIMILAR TO 'a%z'    -- false
```

Use a **round brackets** to **group** a complete expression as single one

```
'ab'     SIMILAR TO '(ab){2}' -- false
'aabb'   SIMILAR TO '(ab){2}' -- false
'abab'   SIMILAR TO '(ab){2}' -- true
```

Use a **square brackets** to match a character identical to one of **character enumeration**

```
'b'     SIMILAR TO '[abc]'    -- true
'd'     SIMILAR TO '[abc]'    -- false
'9'     SIMILAR TO '[0-9]'    -- true
'9'     SIMILAR TO '[0-8]'    -- false
```

Use a square brackets and **circumflex** to match a character **not** identical to one of enumeration

```
'b'     SIMILAR TO '[^abc]'   -- false
'd'     SIMILAR TO '[^abc]'   -- true
```



Common SQL : SIMILAR TO



Match a character identical to one enumeration except of another enumeration

```
'3' SIMILAR TO '[[0-9]^3]' -- false
'4' SIMILAR TO '[[0-9]^1-3]' -- true
```

Match a character identical to one character included in <character class identifier>

```
'4' SIMILAR TO '[:DIGIT:]' -- true
'a' SIMILAR TO '[:DIGIT:]' -- false
'4' SIMILAR TO '[^:DIGIT:]' -- false
'a' SIMILAR TO '[^:DIGIT:]' -- true
```

Character class identifiers:

Identifier	Description
ALPHA	All characters that are simple latin letters (a-z, A-Z).
UPPER	All characters that are simple latin uppercase letters (A-Z). Includes lowercase letters when using case-insensitive collation.
LOWER	All characters that are simple latin lowercase letters (a-z). Includes uppercase letters when using case-insensitive collation.
DIGIT	All characters that are numeric digits (0-9).
SPACE	All characters that are the space character (ASCII 32).
WHITESPACE	All characters that are whitespaces (vertical tab (9), newline (10), horizontal tab (11), carriage return (13), formfeed (12), space (32)).
ALNUM	All characters that are simple latin letters (ALPHA) or numeric digits (DIGIT).



Common SQL : SQLSTATE



SQLSTATE is standard compliant generic error code for use by generic applications

SQLCODE is deprecated (but still supported) and it is recommended to use **SQLSTATE**

To obtain SQLSTATE value use new API function : **fb_sqlstate**

SQLSTATE is not available (yet) for WHEN block of PSQL exception handling

isql since v2.5 used SQLSTATE in error messages :

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.
```





Hexadecimal numeric and binary string literals

Hexadecimal digits :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, a, b, c, d, e, f

Numeric literals : *0xHHHHHHHH*

Prefix *0x* or *0X*

Up to 16 hexadecimal digits

1 - 8 digits : data type is *signed integer*

9 - 16 digits : data type is *signed bigint*

0xF0000000 == -268435456 -- signed integer

0x0F0000000 == 4026531840 -- signed bigint

String literals : *x'HH...H'*

Prefix *x* or *X*

Data type is **CHAR(N / 2) CHARACTER SET OCTETS**, where N is a number of digits in string

```
SQL> SELECT 'First line' || CAST(x'0D0A09' AS CHAR(3)) || 'Second line'
```

```
CON> FROM RDB$DATABASE;
```

CONCATENATION

=====

First line

Second line





UUID binary to/from CHAR conversion

From char to binary : **CHAR_TO_UUID**

Converts the CHAR(32) ASCII representation of an UUID
(XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX) to the CHAR(16) OCTETS
representation (optimized for storage).

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

```
CHAR_TO_UUID  
=====  
6BA95B282946A145869AECA865CFF461
```

From binary to char : **UUID_TO_CHAR**

Converts a CHAR(16) OCTETS UUID to the CHAR(32) ASCII representation

```
SQL> SELECT UUID_TO_CHAR(x' 6BA95B282946A145869AECA865CFF461') FROM  
      RDB$DATABASE;
```

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





Syntax

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

Parameters ?

Same as outer transaction (isolation level, read\write, wait mode, etc)
Configurable ? Not yet... may be later

How it ends ?

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

Notes

Autonomous transaction and its outer transaction fully independent and isolated from each other as any other two transactions.

Temporary BLOBs, created in autonomous transaction, attached to outer transaction. This is done to allow usage of such blobs after autonomous transaction ends. This behavior may be a source of unexpected additional memory consumption.





New implementation of EXECUTE STATEMENT :

- Input parameters
- May run with privileges of caller PSQL object
- Autonomous transactions
- Query another Firebird database
- Full backward compatibility

Syntax

```
[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>]
```





Input parameters

Named input parameters

EXECUTE STATEMENT

```
( 'INSERT INTO TABLE VALUES (:a, :b, :a) ' )  
(a := 100, b := CURRENT_CONNECTION)
```

Not named input parameters

EXECUTE STATEMENT

```
( 'INSERT INTO TABLE VALUES (?, ?, ?) ' )  
(100, CURRENT_CONNECTION, 100)
```





Caller privileges

```
-- 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'
```





Caller privileges

```
-- 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
```

Dynamic statement is executed with that set of privileges as it would have if its executed immediately by caller PSQL object (procedure or trigger)





Transactions

Common transaction (default)

statement executed in the current transaction

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

Autonomous transaction

statement executed in the separate new transaction

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

parameters the same as current transaction, not (yet) configurable





Query another Firebird database

EXTERNAL DATA SOURCE clause :

DATA SOURCE is optional

<connection_string> is usual Firebird's connection string

Query another database using user\password

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

When user\password is not set

Trusted authentication (Windows only) : Firebird's **process account name** is effective user name at **remote database** (if TA is supported by remote side)

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

CURRENT_USER is effective user name at **local database**

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





Query another Firebird database

Internal (local) connection : execution context is context of current connection

- **EXTERNAL DATA SOURCE** clause is omitted, or
- **<connection_string>** is NULL
- and
- **USER** clause is omitted, or
- **<user_name>** value is equal to **CURRENT_USER** value, or
- **<user_name>** is NULL

External (remote) connection : separate connection with remote server is established

- **EXTERNAL DATA SOURCE** clause is specified, and
- **<connection_string>** is not NULL
- or
- **USER** clause is specified, and
- **<user_name>** value is not equal to **CURRENT_USER** value, and
- **<user_name>** is not NULL

WITH CALLER PRIVELEGES is **ignored** for external connections





Query another Firebird database

COMMON TRANSACTION

- If connection is external
 - started new transaction with the same parameters as current transaction, or
 - used already started transaction in this connection
- If connection is internal
 - used current transaction

Transaction lifetime is bound to the lifetime of current transaction so common transaction committed (rolled back) the same way as current transaction

AUTONOMOUS TRANSACTION

Always started new transaction with the same parameters as current transaction. Autonomous transaction will be committed if the statement is executed ok or rolled back if the statement is executed with errors.





Syntax

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

Usage

- Input and output parameters

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

- Variable declaration

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

- CAST statement

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





Syntax

```
{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>);
```

- **Task**
 - change view (or computed field) definition
- **Before Firebird 2.5**
 - drop and create (or alter two times) again all dependent objects
 - restore all privileges granted to all dependent objects which was re-created
- **Firebird 2.5**
 - just issue **ALTER VIEW** and enjoy :-)



ALTER VIEW \ COMPUTED FIELD



EXAMPLE

```
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 ...; -- dependent procedure
```

Try to change both view and computed field definitions

Before Firebird 2.5 :

```
ALTER PROCEDURE P AS BEGIN END; -- empty body for ALL dependent objects  
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; -- restore ALL dependent  
ALTER PROCEDURE P ... SELECT ... FROM V ...; -- objects and privileges
```

What if you have hundreds dependent objects ?

Firebird 2.5 :

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





Syntax

```
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
```





When Windows Administrator is connected using trusted authentication :

Firebird 2.1

**CURRENT_USER is a SYSDBA
SYSDBA privileges**

Firebird 2.5

CURRENT_USER is always a Domain\User

Auto-admin mapping (per database)

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

Connect without a role or with RDB\$ADMIN role

**Auto-admin mapping is off (by default)
ordinary privileges**

**Auto-admin mapping is on
SYSDBA privileges**

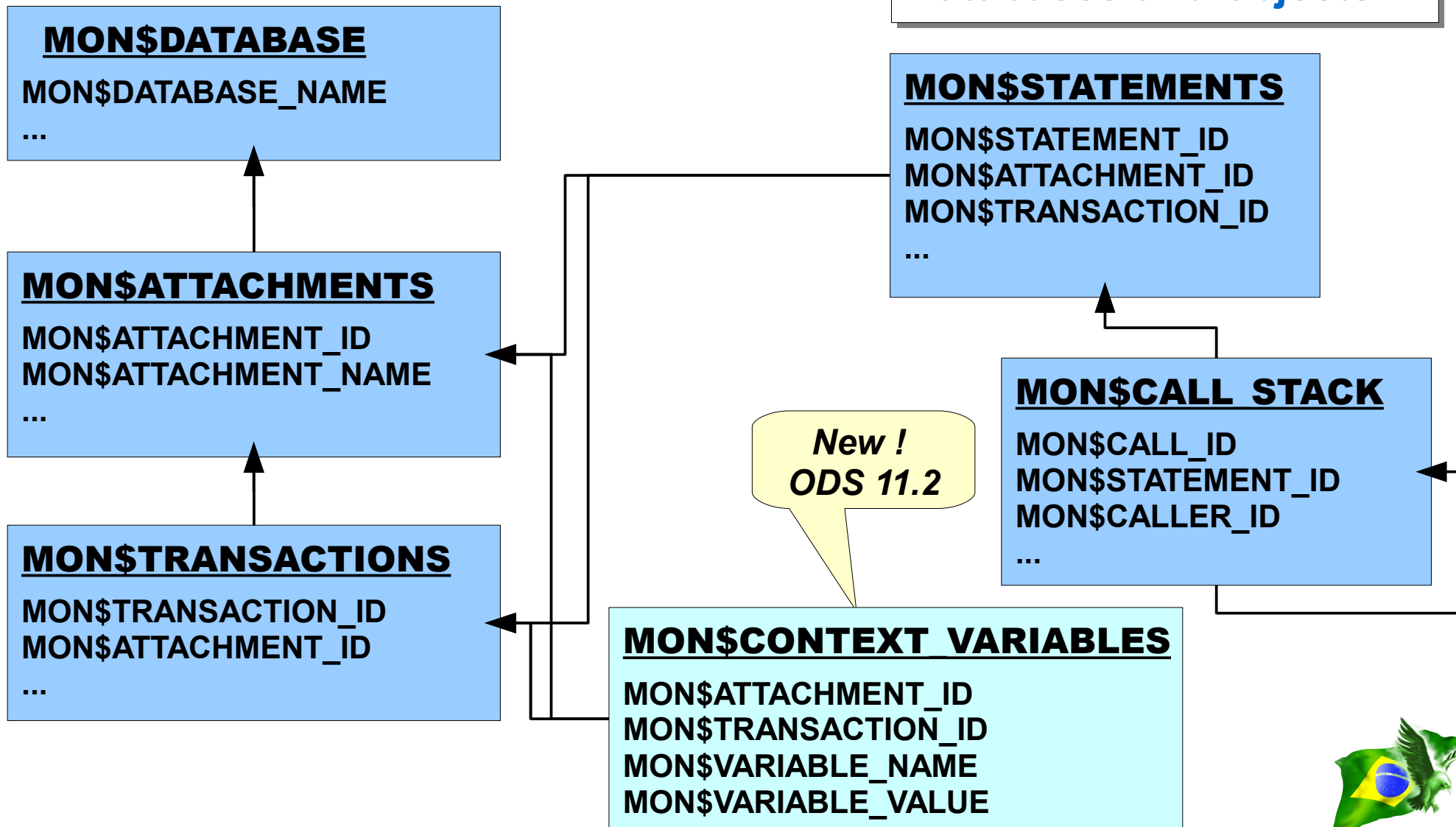
**Connect with a not RDB\$ADMIN role
ordinary privileges**



MORE MONITORING TABLES



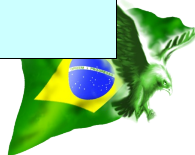
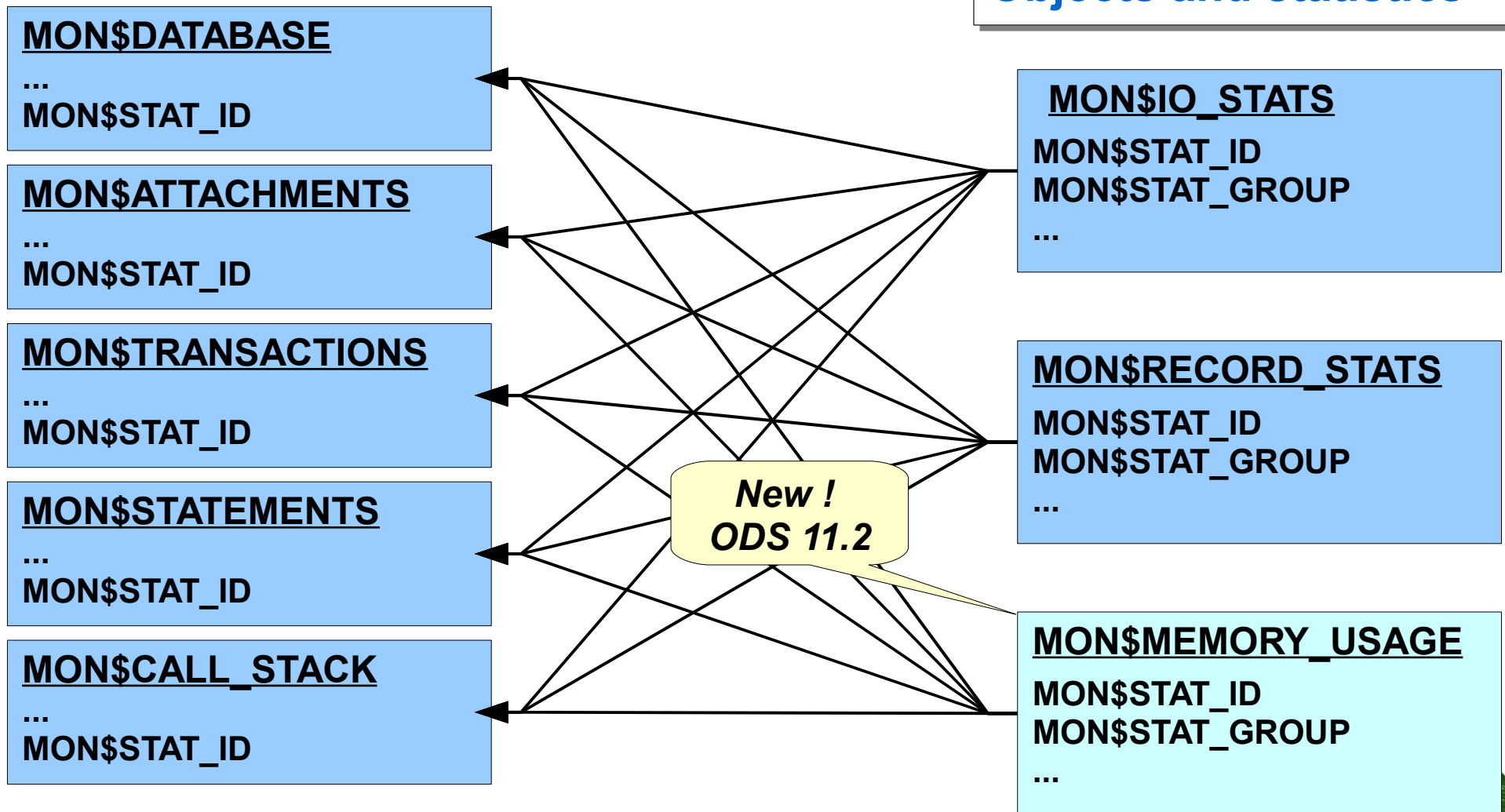
Databases and objects



MORE MONITORING TABLES



Objects and statistics



MORE MONITORING TABLES



- **MON\$MEMORY_USAGE** *current memory usage*
- **MON\$STAT_ID** *statistics ID*
- **MON\$STAT_GROUP** *statistics group*
 - 1 *database*
 - 2 *attachment*
 - 3 *transaction*
 - 4 *statement*
 - 5 *call*
- **MON\$MEMORY_USED** *number of bytes currently in use*
- **MON\$MEMORY_ALLOCATED** *number of bytes currently allocated at the OS level*
- **MON\$MAX_MEMORY_USED** *maximum number of bytes used by this object*
- **MON\$MAX_MEMORY_ALLOCATED** *maximum number of bytes allocated from OS by this object*



MORE MONITORING TABLES



- **MON\$CONTEXT_VARIABLES** *known context variables*
- **MON\$ATTACHMENT_ID** *attachment ID*
- **MON\$TRANSACTION_ID** *transaction ID*
- **MON\$VARIABLE_NAME** *name of context variable*
- **MON\$VARIABLE_VALUE** *value of context variable*

Some monitoring tables is active

1) Cancel current activity of connection № 32:

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

2) Disconnect everybody but ourselves (new in Firebird 2.5):

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



OBRIGADO PELA VOSSA ATENÇÃO !



Questions ?

[Firebird official web site](#)

[Firebird tracker](#)

hvlad@users.sf.net

