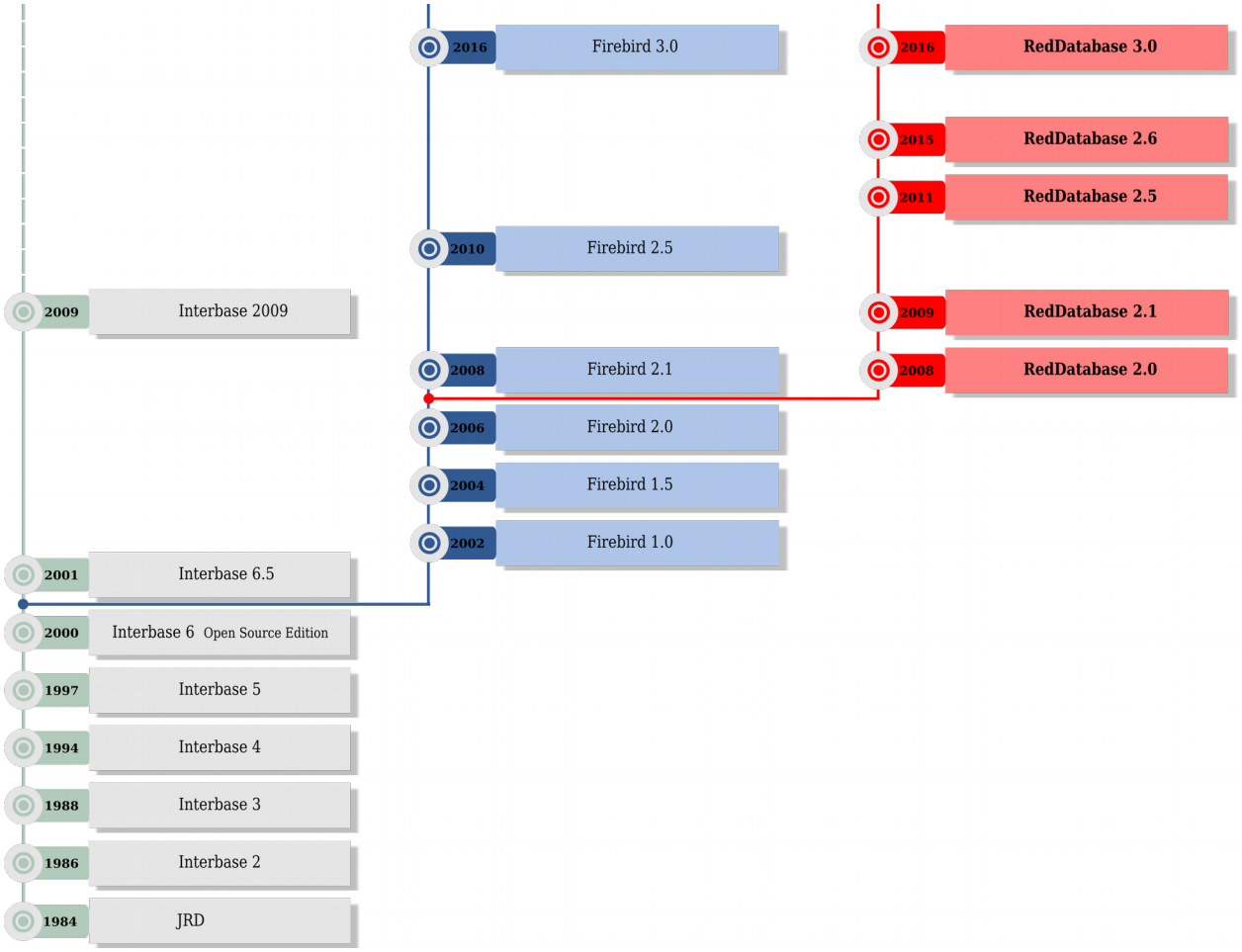
New features of Firebird ported from RedDatabase

Roman Simakov, director of system development department

REDSOFT

History of Red Database



REDSOFT

Security features

- Cryptographic plugin
- Multi-factor authentication
- Cumulative roles
- DML access control
- DDL access control
- Service access control
- Record filtering

Functional features

- Java Stored Procedures
- Full Text Search
- LDAP/AD integration
- StandBy cluster (engine-level replication)

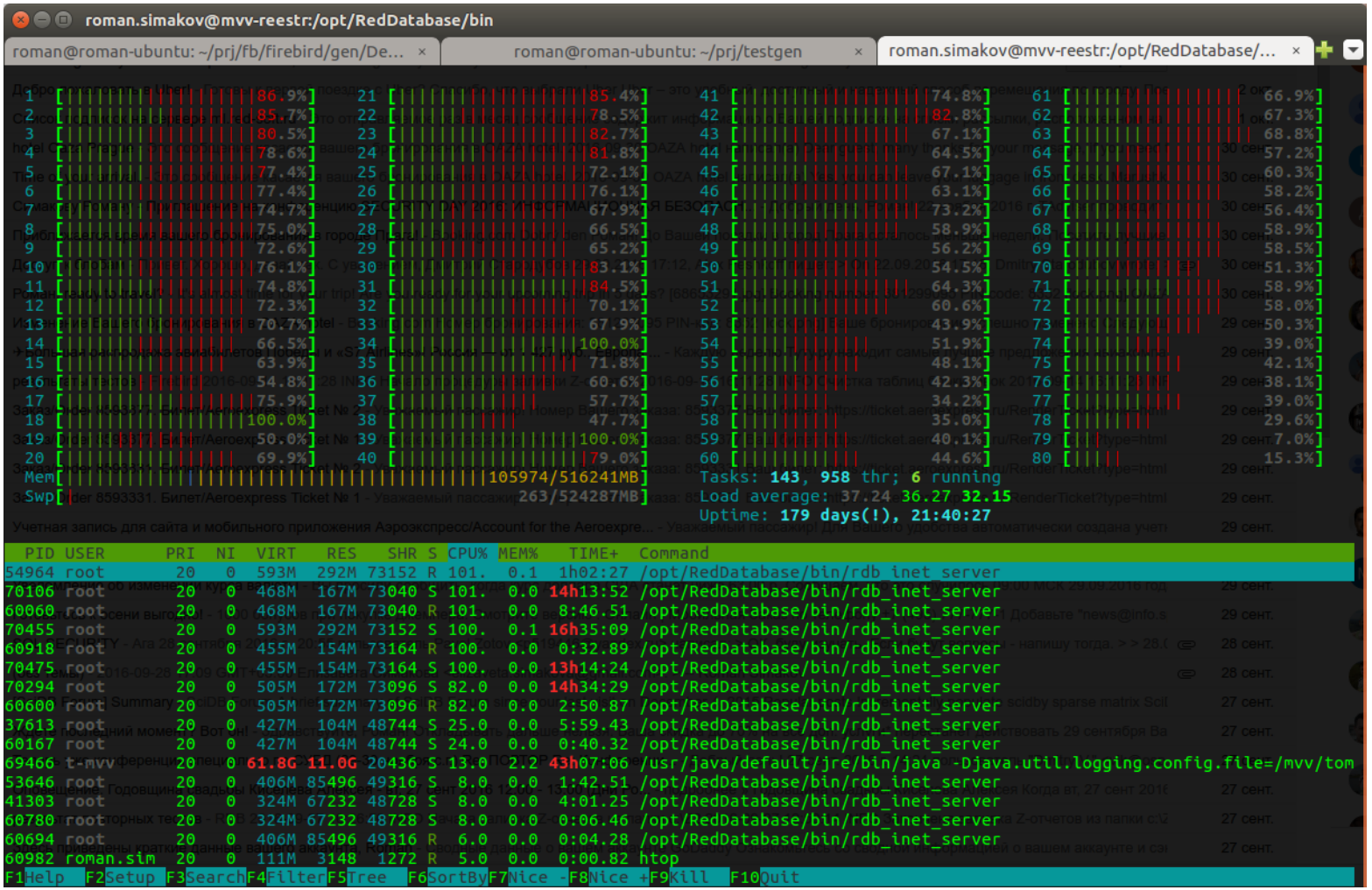
REDSOFT

Automated Information System of Federal Service for Officers of Justice of Russia

- AIS is installed and work in 85 regional departments and in the main office of FSOJ of Russia
- Total amount of Red Database installations are about 2720, i.e. every city of Russia has one or several Red Database servers
- AIS handles more than 10^9 documents per year
- AIS works in 24/7 mode
- Some databases more than 1TB and a lot of data goes to archived set of database files
- 100x of concurrent connections
- 100 000x transactions per hour

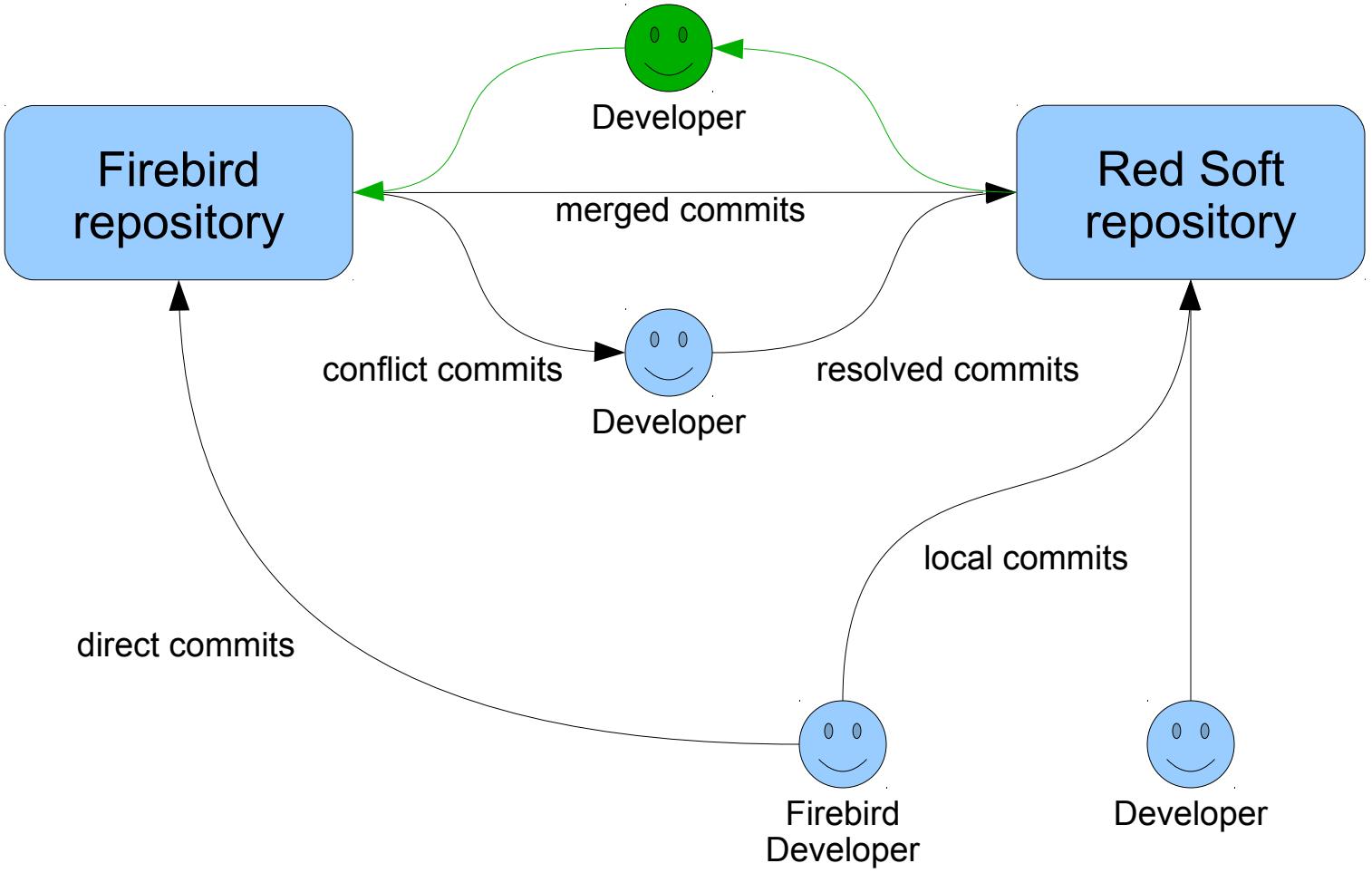
REDSOFT

Example of load



REDSOFT

Development process (schema)



REDSOFT

DDL access control (CORE-735 since 2003)

Problem: Previously **anyone** who connected to database could create a table

Syntax

```
GRANT CREATE <OBJECT> TO [USER | ROLE] <user/role name> [WITH GRANT OPTION]
GRANT ALTER ANY <OBJECT> TO [USER | ROLE] <user/role name> [WITH GRANT OPTION]
GRANT DROP ANY <OBJECT> TO [USER | ROLE] <user/role name> [WITH GRANT OPTION]
REVOKE [GRANT OPTION FOR] CREATE <OBJECT> FROM [USER | ROLE] <user/role name>
REVOKE [GRANT OPTION FOR] ALTER ANY <OBJECT> FROM [USER | ROLE] <user/role name>
REVOKE [GRANT OPTION FOR] DROP ANY <OBJECT> FROM [USER | ROLE] <user/role name>
```

Where OBJECT can be:

TABLE, VIEW, PROCEDURE, FUNCTION, PACKAGE, GENERATOR, SEQUENCE, DOMAIN, EXCEPTION, ROLE, CHARACTER SET, COLLATION, FILTER

REDSOFT

DDL access control: Simple example

```
GRANT CREATE TABLE TO Joe
```

*ALTER possible because of Joe is owner of JoeT

```
GRANT ALTER ANY TABLE TO Joe
```

*Triggers and indices re-use table privileges

```
REVOKE CREATE TABLE FROM Joe
```

```
CREATE TABLE JoeT (I INTEGER)
```

```
ALTER TABLE JoeT ...
```

```
ALTER TABLE BobT ...
```

```
CREATE INDEX ON JoeT ...

CREATE INDEX ON BobT ...
```

```
CREATE TRIGGER ON T ...
```

```
ALTER TRIGGER ON T ...
```

REDSOFT

DDL access control: special form for managing database

```
GRANT CREATE DATABASE TO [USER | ROLE] <user/role name>

GRANT ALTER DATABASE TO [USER | ROLE] <user/role name> [WITH GRANT OPTION]

GRANT DROP DATABASE TO [USER | ROLE] <user/role name> [WITH GRANT OPTION]

REVOKE CREATE DATABASE FROM [USER | ROLE] <user/role name>

REVOKE [GRANT OPTION FOR] ALTER DATABASE FROM [USER | ROLE] <user/role name>

REVOKE [GRANT OPTION FOR] DROP DATABASE FROM [USER | ROLE] <user/role name>
```

ALTER DATABASE permissions is used to check following actions:

1) Altering database itself
2) Commenting on database or db level triggers
3) Managing db level triggers
4) Managing shadows
5) Direct editing RDB$FILES system table

REDSOFT

Cumulative roles (CORE-1815 since 2008)

Now you can grant role to another role except circle references

Syntax:

```
GRANT [DEFAULT] <role name> TO [USER | ROLE] <user/role name> [WITH ADMIN OPTION]

REVOKE [DEFAULT] <role name> FROM [USER | ROLE] <user/role name> [WITH ADMIN OPTION]
```

ADMIN OPTION allows grantee to grant the role to another user or role

1) WORKER->MANAGER->Joe

2) WORKER->MANAGER=>Joe

3) WORKER=>MANAGER->Joe

4) WORKER=>MANAGER=>Joe

Syntax:
RDB$ROLE_IN_USE(role_name varchar(32)) RETURNS BOOLEAN

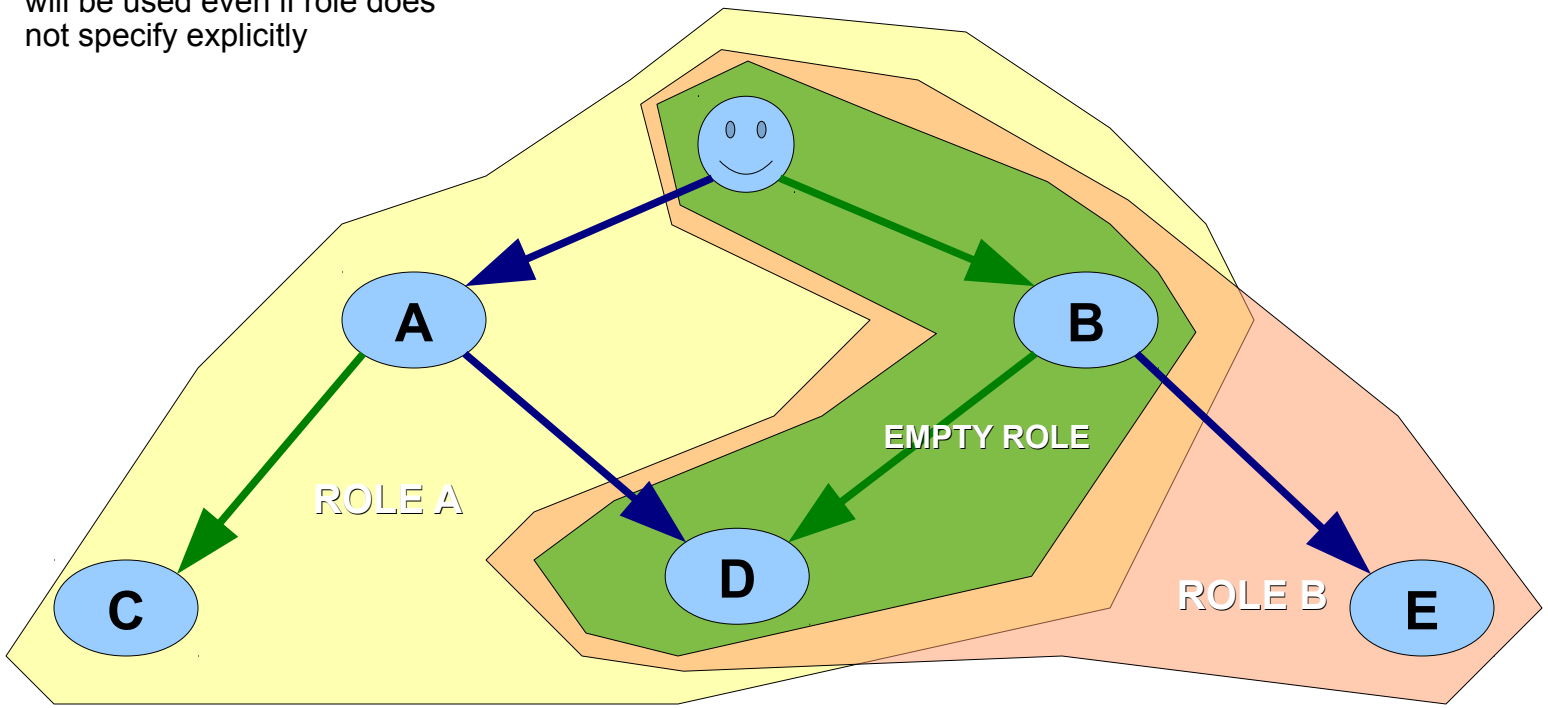
To get a list of currently active roles you can run:

SELECT * **FROM** RDB$ROLES **WHERE** RDB$ROLE_IN_USE(RDB$ROLE_NAME)

REDSOFT

Cumulative roles: DEFAULT explanation

DEFAULT means that this role will be used even if role does not specify explicitly



DEFAULT

WITHOUT DEFAULT

REDSOFT

SQL SECURITY (SQL STANDARD 2003, 2011)

Syntax

```
CREATE TABLE <TABLENAME> (...) [SQL SECURITY {DEFINER | INVOKER}]

ALTER TABLE <TABLENAME> ... [{ALTER SQL SECURITY {DEFINER | INVOKER} | DROP SQL SECURITY}]

CREATE [OR ALTER] TRIGGER <TRIGGERNAME> ... [SQL SECURITY {DEFINER | INVOKER} | DROP SQL SECURITY]
[AS ...]

CREATE [OR ALTER] FUNCTION <FUNCTIONNAME> ... [SQL SECURITY {DEFINER | INVOKER}] AS ...

CREATE [OR ALTER] PROCEDURE <PROCEDURENAME> ... [SQL SECURITY {DEFINER | INVOKER}] AS ...

CREATE [OR ALTER] PACKAGE <PACKAGENAME> [SQL SECURITY {DEFINER | INVOKER}] AS ...
```

REDSOFT

SQL SECURITY (Calculated columns)

```
connect 'localhost:/tmp/db.fdb' user sysdba password 'masterkey';
set term ^;
create function f() returns int
as
begin
    return 3;
end^
set term ;^

create table t (i integer, c computed by (i + f())) sql security definer;
insert into t values (2);
grant select on table t to user us;
grant execute on function f to user us;
commit;

connect 'localhost:/tmp/db.fdb' user us password 'pas';
select * from t;
```

REDSOFT

SQL SECURITY (Functions)

```
connect 'localhost:/tmp/db.fdb' user sysdba password 'masterkey';
set term ^;
create function f (i integer) returns int sql security definer
as
begin
  insert into t values (:i);
  return i + 1;
end^
set term ;^
grant execute on function f to user us;
grant insert on table t to user us;
commit;

connect 'localhost:/tmp/db.fdb' user us password 'pas';
select f(3) from rdb$database;
```

REDSOFT

SQL SECURITY (Stored procedures)

```
connect 'localhost:/tmp/db.fdb' user sysdba password 'masterkey';
set term ^;
create procedure p (i integer) sql security definer
as
begin
  insert into t values (:i);
end^
set term ;^
grant execute on procedure p to user us;
grant insert on table t to user us;
grant insert on table t to procedure p;
commit;

connect 'localhost:/tmp/db.fdb' user us password 'pas';
execute procedure p(1);
```

REDSOFT

SQL SECURITY (Triggers)

```
connect 'localhost:/tmp/db.fdb' user sysdba password 'masterkey';
create table tr (i integer) sql security definer;
create table t (i integer);
set term ^;
create trigger tr_ins for tr after insert sql security definer
as
begin
  insert into t values (NEW.i);
end^
set term ;^
grant insert on table tr to user us;
grant insert on table t to user us;
commit;

connect 'localhost:/tmp/db.fdb' user us password 'pas';
insert into tr values(2);
```

REDSOFT

SQL SECURITY (Packages)

```
connect 'localhost:/tmp/db.fdb' user sysdba password 'masterkey';
create table t (i integer);
set term ^;
create package pk sql security definer
as
begin
    function f(i integer) returns int;
end^
create package body pk
as
begin
    function f(i integer) returns int
    as
    begin
      insert into t values (:i);
      return i + 1;
    end
end^
set term ;^
grant execute on package pk to user us;
grant insert on table t to user us;
commit;

connect 'localhost:/tmp/db.fdb' user us password 'pas';
select pk.f(3) from rdb$database;
```

REDSOFT

Thanks!

visit: www.red-soft.ru
roman.simakov@red-soft.ru