RedDatabase 4.0
Overview

Berlin, 2019

Firebird Conference 2019

Berlin, 17-19 October

IBSurgeon

About Red Soft

- Founded in 2006

- Has several laboratories located in different Russian cities:
  - Moscow
  - Murom
  - Dubna
  - Tver
  - a partner of Skolkovo

- The main products:
  - RedDatabase - a relational DBMS
  - RedOS - an rpm-based operating system
  - RedPlatform - an electronic document management platform
  - Several Information Systems of the federal scale (The service of bailiffs)

History of RedDatabase

Year	1984	2000	2001	2002	2006	2008	2009	2010	2011	2015	2016	2017	2019
Interbase	JRD	6.0	6.5				2009						
Firebird				1.0	2.0	2.1		2.5			3.0		4.0
RedDatabase						2.0	2.1		2.5	2.6		3.0	4.0

- Versions that share the same code base are marked by the same color

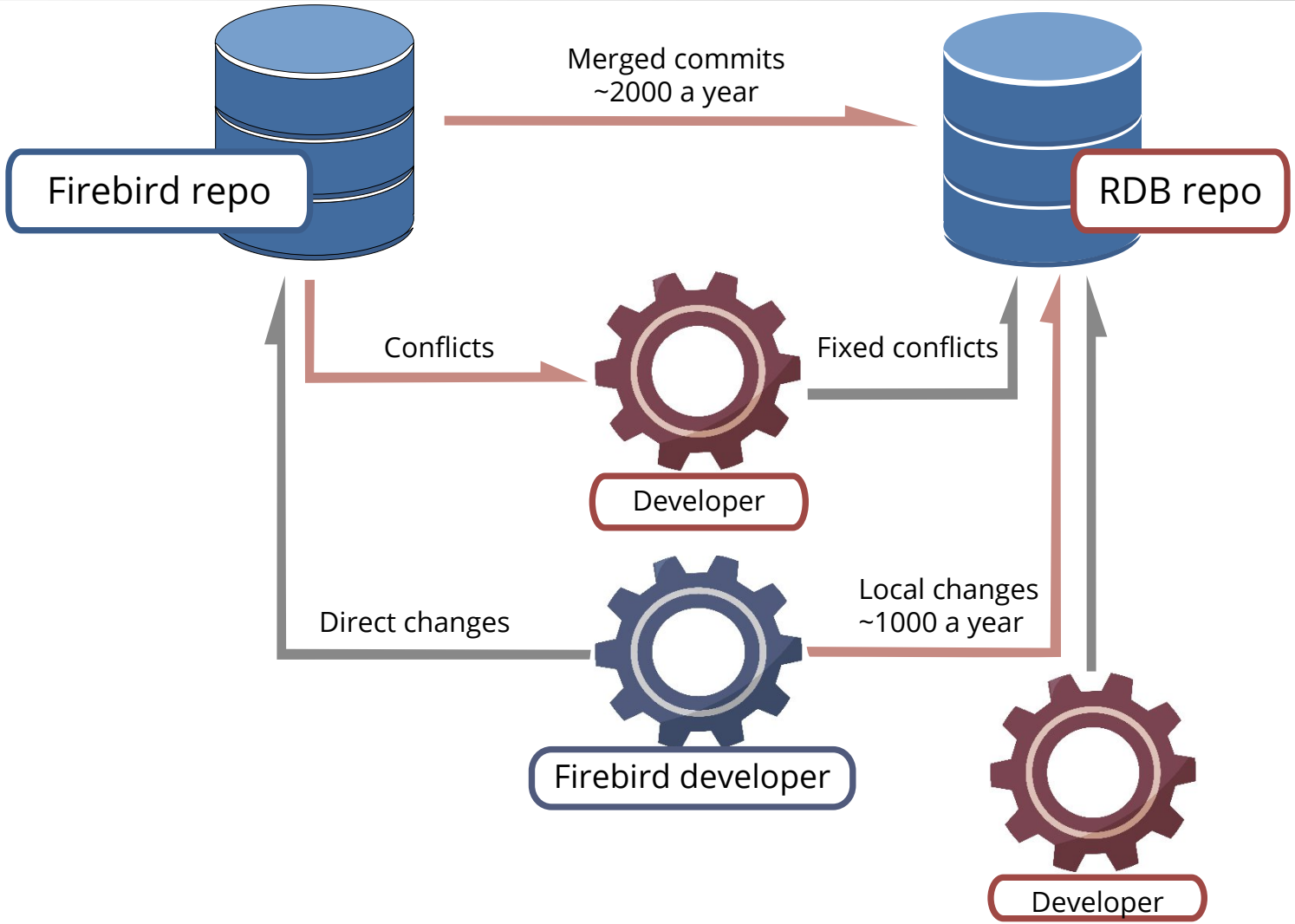
RedDatabase -> Firebird

Feature	Merge year
nbackup (later improved by Vlad)	2007
Trace	2009
DDL access control (CORE-735 since 2003)	2014
Cumulative roles (CORE-1815 since 2008)	2016
SQL SECURITY (SQL STANDARD 2003, 2011)	2016
Blob access	2018
Sync and async replication	2019
Snapshot consistency and intermediate garbage collection (Nikolay Samofatov (Red Soft) + Vlad Khorsun (Firebird))	2019

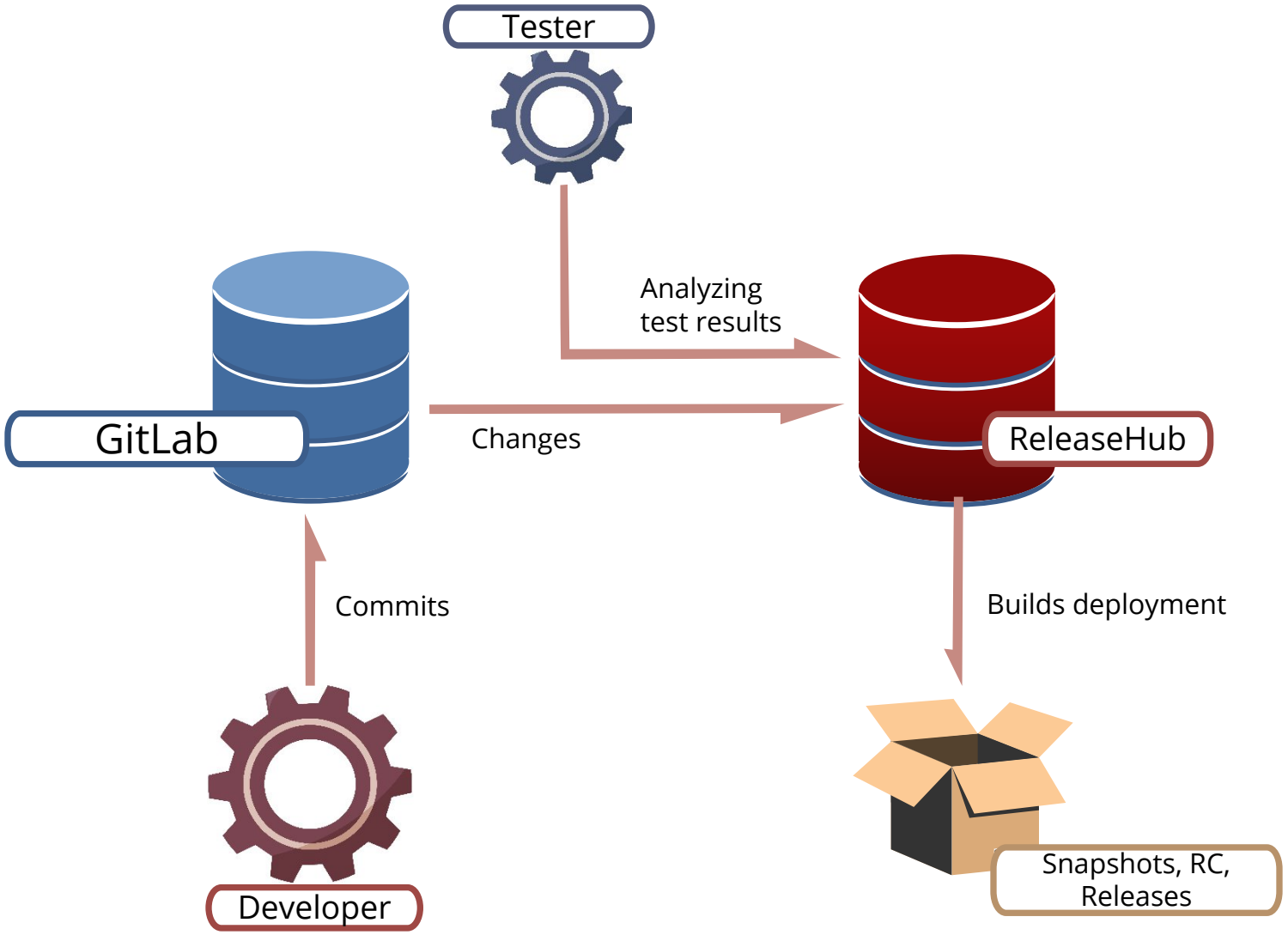
The main Features (except supported by Firebird)

- Russian standard certificate of security
- Supports:
  - sync replication
  - async replication
  - HA cluster with pacemaker
- A lot of additional authentication plugins:
  - LDAP/AD
  - GSS API
  - Certificates
  - Based on Russian GOST cryptography
- Java procedures, functions and triggers out of the box

Development process



DevOps


Tester
Analyzing
test results
GitLab
Changes
ReleaseHub
Commits
Builds deployment
Developer
Snapshots, RC,
Releases

Technical Support

- **Standard**
  - Consulting by phone or email
  - Handling requests from 10:00 till 18:00 MSK every working day
  - Free updates to the new versions of the RedDatabase
  - Post-install configuring of the customer server based on an application features

- **Extended**
  - Time of availability:
    - by e-mail – round-the-clock
    - by phone – from 9:00 till 19:00 in working day, from 10:00 till 17:00 in holidays
  - Shorten time of reaction on a request
  - A dedicated manager
  - Administration of the customer services

Migration

- Firebird/Interbase -> RedDatabase - no problems
- A tool which converts the Oracle's metadata into the RedDatabase ones
  - Take into account SQL differences between Oracle and RedDatabase
  - Minimize manual labor of a programmer
  - Mark and comment code if cannot convert it
- RedReplicator - ETL tool for Firebird/RedDatabase
- In the plans:
  - MS SQL Server
  - IBM DB2

Courses

- Database administration
- Application software development
- Possible platforms:
  - on Red Soft site
  - on customer site
  - in form of webinar
- There is a test in the end of education
- If the test results are acceptable we issue a certificate

What's new in RedDatabase 4.0

- Except Firebird features

- Job scheduler
- Multifactor authentication based on plugins
- Security policies
- Alter procedure using BLR
- CSV external tables
- Tablespaces

Job scheduler

- Job descriptions are stored in **scheduler.fdb**
- Setting in **scheduler.conf**

```
CREATE OR ALTER JOB <job_name>
[ACTIVE | INACTIVE]
<cron time> | '@reboot'
[START DATE '<timestamp>' | NULL]
[END DATE '<timestamp>' | NULL]
{AS
<declaration of variables>
BEGIN
<SQL statements block>
END
| COMMAND '<command>' }

<cron time> ::= '<Minutes> <Hours> <Days of Month> <Months> <Days of Week>'
```

Job scheduler

- @reboot - run a job on the next server start
- COMMAND - to run a command of OS (Only DB admins can create the job)
- schedule.conf:JobCommandAccess - a list of available programs or their directories
- A job uses owner privileges

Notifications:

- schedule.conf:JobNotificationCommand - a command of OS to perform a notification
- Available variables:
  - $(timestamp) - date and time of the event
  - $(job_name) - a name of the job
  - $(job_id) - ID of the job
  - $(event) - a type of the event: RUN_START, RUN_FINISH or RUN_ERROR
  - $(message) - an error message

Job scheduler. Example.

- **Job to recompute index statistics in 00:00 every Sunday**

```
create job RECOMPUTE_IDX_STAT '0 0 * * 7' -- Sunday 00:00
as
   declare variable index_name varchar(31);
begin
   for select trim(rdb$index_name) from rdb$indices into :index_name do
     execute statement 'set statistics index ' || :index_name || ';';
end^
```

- **Job to make a backup of the database in 01:00 every Sunday**

```
create job DB_BACKUP '0 1 * * 7' -- Sunday 01:00
command '/opt/RedDatabase/bin/gbak -b mydb /var/backups/mydb.fbk -user SYSDBA -password
masterkey';
```

Multifactor authentication based on plugins

- Old multifactor authentication could combine password and certificate and use Russian standard algorithms (GOST, Government standard)
- Firebird 3+ has **separated** authentication from engine and supports a lot of methods.
- RedDatabase 3+ implements the old multifactor authentication in standalone plugin to keep backward compatibility
- RedDatabase 4+ implements every factor in **separated** plugins
  - It's more simple to implement and maintain
- There are two new plugins:
  - GostPassword - RS CHAP protocol
  - Certificate - to authenticate user by certificate
  - Both use cryptoprovider CryptoPro to perform operations like crypt, decrypt, certificate validation, etc.

Security policies

- A special plugin - Policy. The last in the AuthServer.
- When it present:
  - Success of authentication by a plugin doesn't stop authentication process
  - Auth data are collected in AuthBlock
  - Policy plugin analyzes collected data at the end of authentication and decides if it's possible to attach the user to the database.
- Security policies allow to control:
  - Password complexity
  - A number of previous password that must be distinct to the new one
  - Password validity period
  - A number of fail attempts to pass authentication
  - A set of factors (plugins) which a user must pass to be authenticated (allow to require several factors like certificate + password)

Security policies. Example. Syntax.

```
CREATE POLICY TestPolicy AS
        AUTH_FACTORS = (CERTIFICATE, GOSTPASSWORD),
        PSWD_NEED_CHAR = 5,
        PSWD_NEED_DIGIT = 3,
        PSWD_MIN_LEN = 8,
        PSWD_NEED_DIFF_CASE = true,
        PSWD_VALID_DAYS = 15,
        PSWD_UNIQUE_COUNT = 5,
        MAX_FAILED_COUNT = 5;
```

Alter procedure using BLR

- CREATE/ALTER PROCEDURE <NAME>(<PARAMETERS>) RETURNS (<COLUMNS>) AS 'BLR_STRING';
- CREATE/ALTER FUNCTION <NAME>(<PARAMETERS>) RETURNS <DATATYPE> AS 'BLR_STRING';
- To get BLR_STRING use:

SELECT **base64_encode(rdb$procedure_blr)** FROM rdb$procedures WHERE rdb$procedure_name = '<PROCEDURE NAME>';

- Allows developer to update procedure/function without opening source code

CSV external tables

- Every record of CSV file is a record of a table
- Empty strings are ignored
- Separator is ','
- 'Extra' fields will be ignored
- Absent values will be NULL
- Skipped values will be NULL - "1,2**,**3"

- Creation of an external table

```
create table CSV_EXT external 'C:\RDB\Tasks\38786\table.csv' adapter 'CSV'(<fields>);
```

- Import of data

```
insert into IMP select * from CSV_EXT;
```

- Export of date

```
delete from CSV_EXT;
insert into CSV_EXT select * from IMP;
```

Tablespaces. Goals.

1. Extend current database size limits even with small data page size.
2. Move not so active or archive part of database to slow volumes of big size.
3. Share read-only tablespaces.
4. Separate BLOBs and index storage.
5. Partial backup, export or import of database.

Tablespaces. Notes.

1. Header page (**pag_header**) in tablespace file is reserved.
2. Every tablespace has its own SCNs (**pag_scns**) and PIPs (**pag_pages**).
3. Index root pages (**pag_root**) are located in the relation's page space.
4. Current limit is **254** tablespaces (total DB size limit is **31PB**!!!).
5. By default CREATE **INDEX** uses **TABLESPACE** of **table**!
6. ALTER TABLE <TABLE NAME> ALTER TABLESPACE ... requires **EX lock** on the database.

Tablespaces. Syntax.

CREATE TABLESPACE <TS NAME> FILE </path/to/file> [{OFFLINE | ONLINE}]
[{READ ONLY | READ WRITE}]

ALTER TABLESPACE <TS NAME> FILE /path/to/file {OFFLINE | ONLINE} |
{READ ONLY | READ WRITE} | {BEGIN | END} BACKUP

DROP TABLESPACE <TS NAME> [INCLUDING CONTENTS]

CREATE TABLE ...
(... <column> BLOB ... TABLESPACE <TS NAME> ...)
TABLESPACE <TS NAME> [<other table options>]

Tablespaces. Syntax.

ALTER TABLE <TABLE NAME> ALTER TABLESPACE <TS NAME>

ALTER TABLE <TABLE NAME> DROP TABLESPACE

ALTER TABLE <TABLE NAME> ALTER <blob column> ALTER TABLESPACE <TABLESPACE>

CREATE INDEX ... TABLESPACE <TABLESPACE>

ALTER INDEX <INDEX NAME> ALTER TABLESPACE <TS NAME>

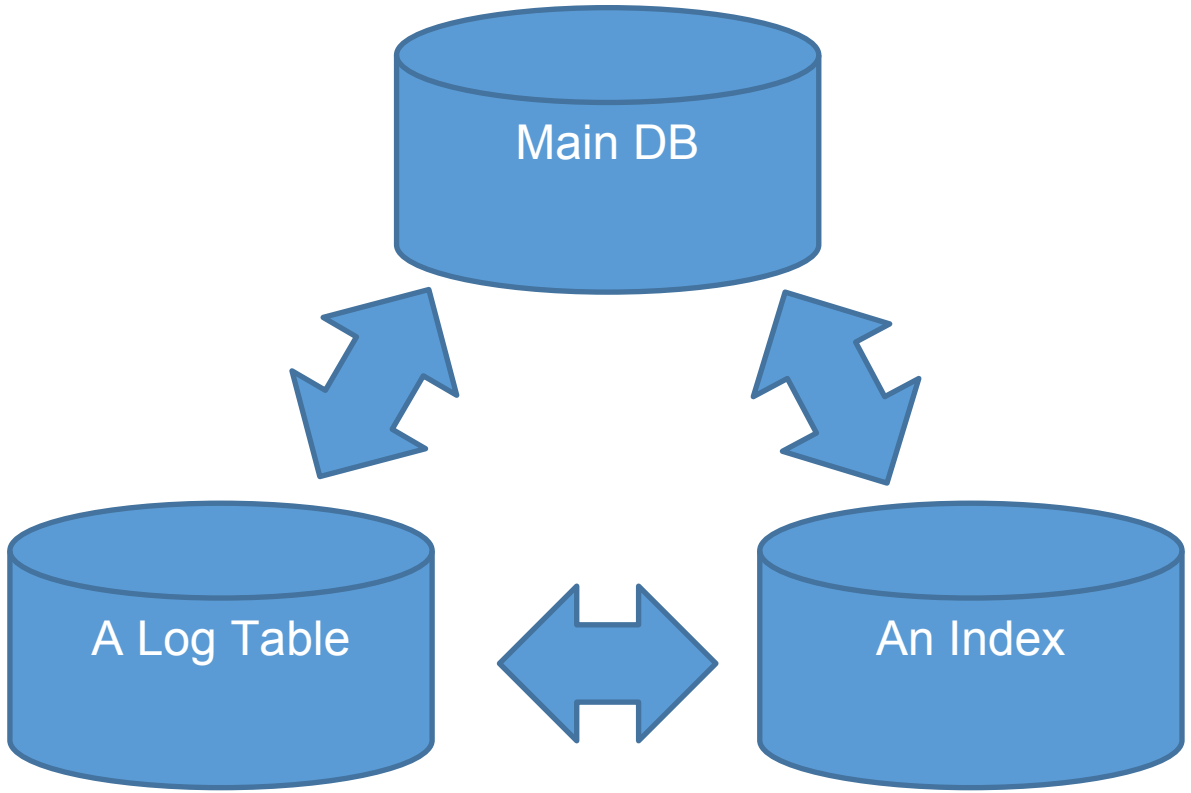
ALTER INDEX <INDEX NAME> DROP TABLESPACE

Tablespaces. Example.

```
CREATE DATABASE '/var/db/main.fdb';
CREATE TABLESPACE LOGTS FILE '/var/db/log.fdb';
CREATE TABLESPACE IDXTS FILE '/var/db/idx.fdb';
...
CREATE TABLE PRICE (
        PRODUCT_ID INTEGER PRIMARY KEY,
        PRODUCT_NAME VARCHAR(256), PRICE
NUMERIC(18, 4)
);          -- MAIN DB

CREATE TABLE PRICE_LOG(     PRODUCT_ID
INTEGER,   OLD_PRICE NUMERIC(18, 4),
        NEW_PRICE NUMERIC(18, 4),
        CHANGE_DATE TIMESTAMP
) TABLESPACE LOGTS;

CREATE INDEX PRICE_PRODUCT_NAME_IDX ON
        PRICE (PRODUCT_NAME)
        TABLESPACE IDXTS;
```



Tablespaces. Blob problem.

- We do not know a table which a blob is related to
- Blob linked with table at the time of materialization
- Possible solutions:
  - Create a BLOB in the main DB space and then copy it into correct tablespace
  - Guess a tablespace for a BLOB and move later in case of miss
  - Add an argument in API of BLOB creation **Can be combined**
  - Use a special tablespace for non-materialized BLOBs
  - **Your ideas???**

Roadmap of RedDatabase 5 (draft)

- Improved tablespaces (BLOB problem, tools, gray parts)
- Partitions
- Point in time recovery
- Support of JSON
- Built-in PSQL debugger
- Performance optimization
- Scalability

Questions?

www.red-soft.ru
www.reddatabase.ru