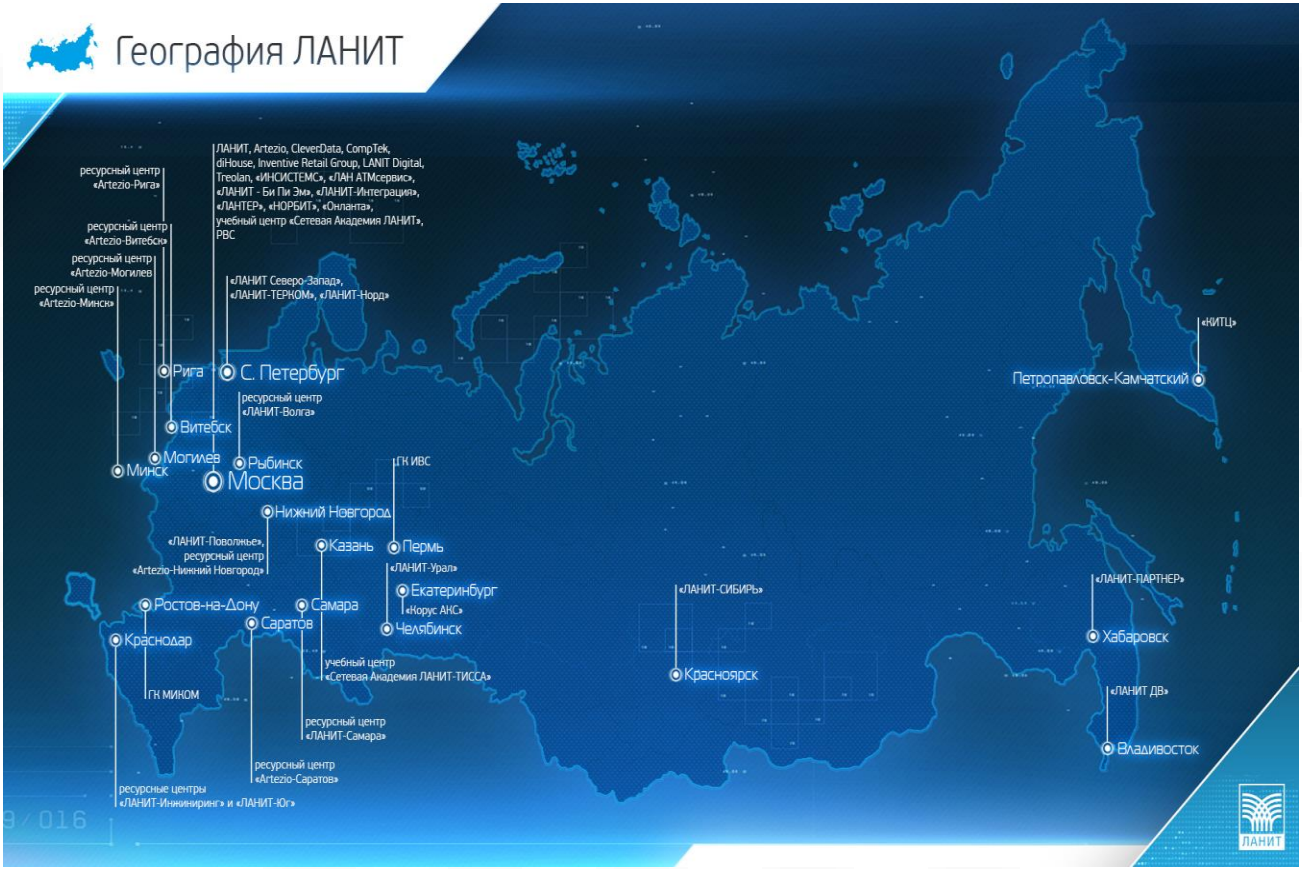
Один базист и сто разработчиков

Михаил Балаян
Руководитель сектора управления данными

ЛАНИТ, 2016

География ЛАНИТ

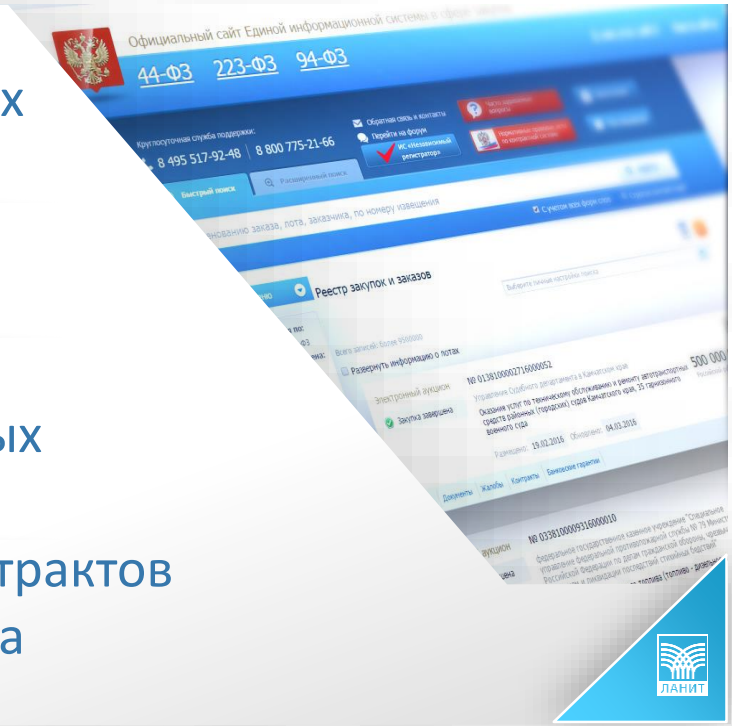


Партнеры


acer
Adobe
APC
by Schneider Electric
ASUS
AVAYA
Canon
Check Point
SOFTWARE TECHNOLOGIES LTD.
CISCO
CITRIX
DAIKIN
DELL
EDB
ENTERPRISEDB
EMC²
EPSON
EXCEED YOUR VISION
IRBIS
FG WILSON
FUJITSU
Hewlett Packard
Enterprise
HUAWEI
intel
legrand
Lenovo
LG
Microsoft
MITSUBISHI
ELECTRIC
ORACLE
Panasonic
PHILIPS
SAMSUNG
SAP
Schneider
Electric
SIEMENS
SONY
Symantec.
TOSHIBA
UNIFY
VERITAS
vmware
xerox

Единая система в сфере закупок

- 30 функциональных подсистем
- 100 млн запросов ежедневно
- более 1 млн зарегистрированных пользователей
- свыше 3,3 млн контрактов по итогам 2015 года

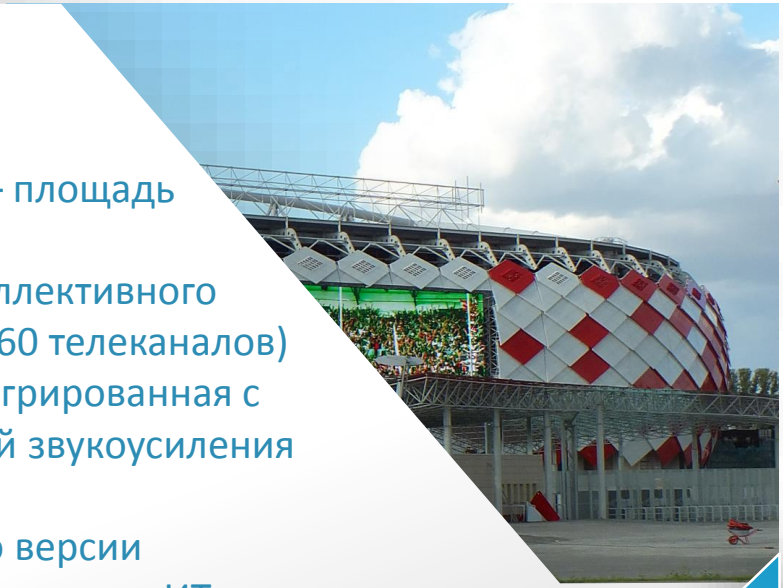


Космодром «Восточный»

- Более 100 тыс. кв. м – площадь помещений, оснащенных инженерными системами
- Более 3 км шинопроводных линий
- Более 800 силовых шкафов и шкафов автоматик
- 1200 км кабельных линий и 40 км труб
- Спроектирован и построен холодильный центр мощностью 8500 кВт

Стадион «Открытие Арена»

- 45 тыс. зрителей
- Свыше 127 тыс. кв. м – площадь объекта
- Построена система коллективного приема телевидения (60 телеканалов) и видеорекламы, интегрированная с видеотабло и системой звукоусиления трибун стадиона
- «Проект 2015 года» по версии ведущего российского портала ИТ-директоров Global CIO



Государственная Информационная Система ЖКХ

- Зарегистрировано около 57 тыс. организаций
- Внесена информация о 637 тыс. многоквартирных и 2,5 млн. жилых домов
- Управляющие организации разместили данные 400 тыс. договоров управления

План

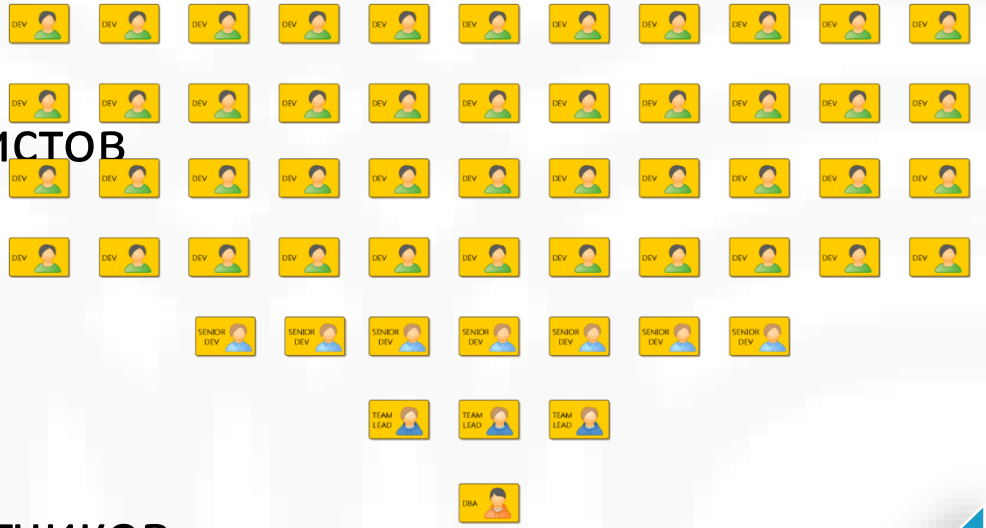
- Проблема
- Требования
- Liquibase изнутри
- Свод правил

Проблема #1. Много разработчиков

- мало специалистов по базам данных

Проблема #1. Много разработчиков

- мало специалистов по базам данных
- много разработчиков

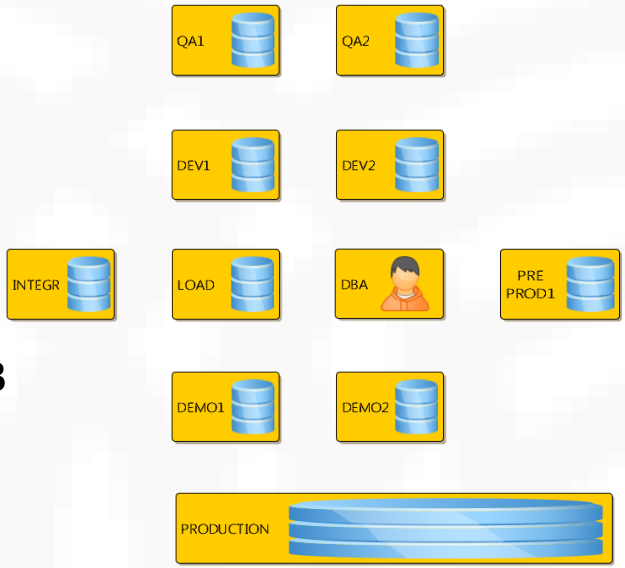


Проблема #2. Много стендов

- мало специалистов по базам

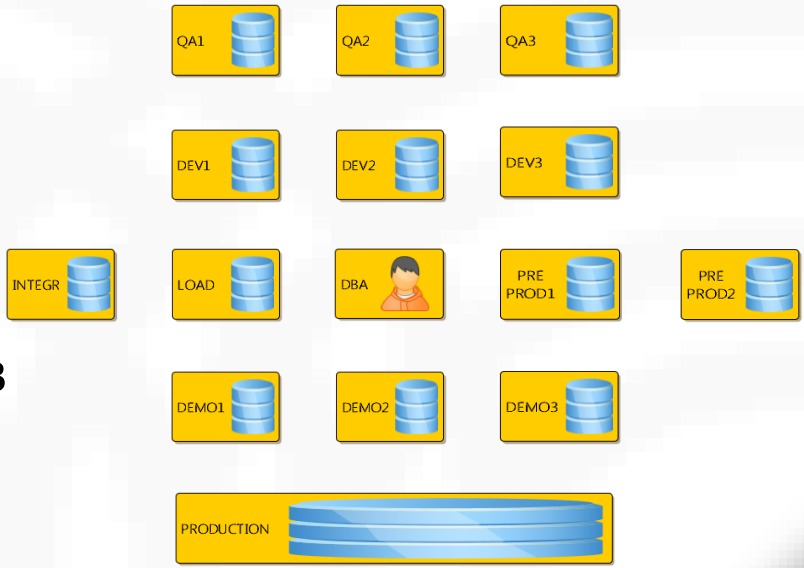
Проблема #2. Много стендов

- мало специалистов по базам
- много стендов

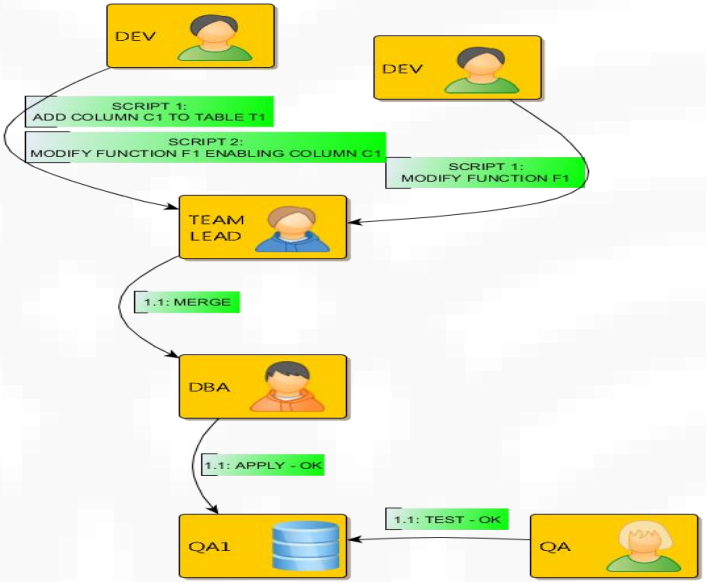


Проблема #2. Много стендов

- мало специалистов по базам
- много стендов
- с разными версиями

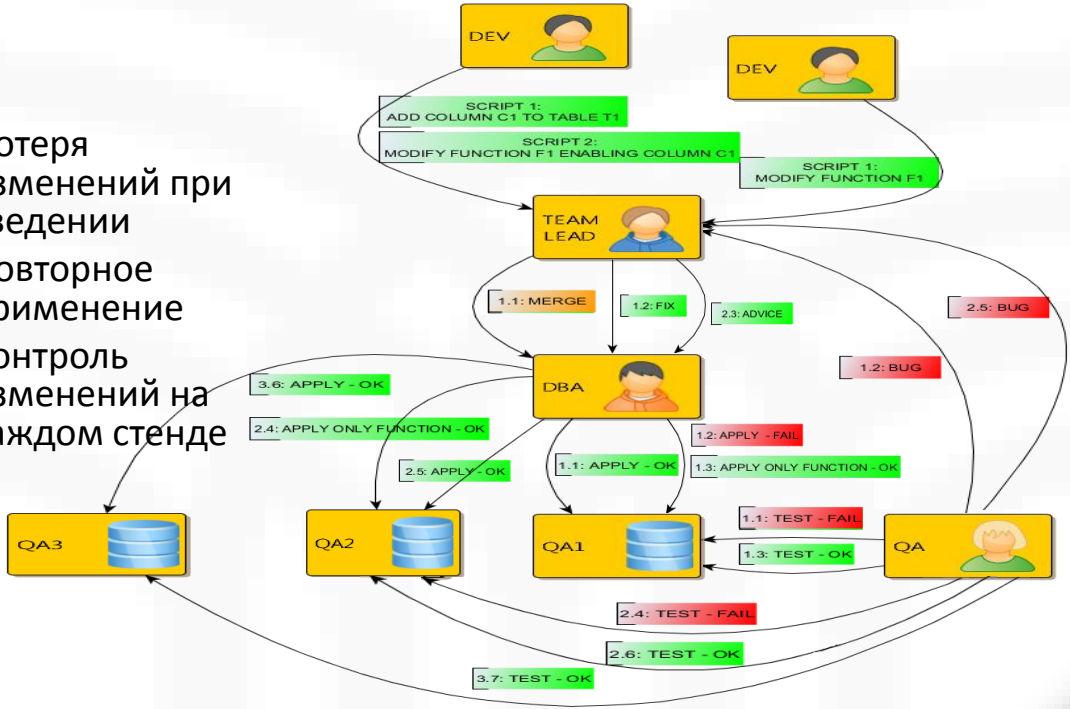


Проблема #3. Человеческий фактор



Проблема #3. Человеческий фактор

- Потеря изменений при сведении
- Повторное применение
- Контроль изменений на каждом стенде

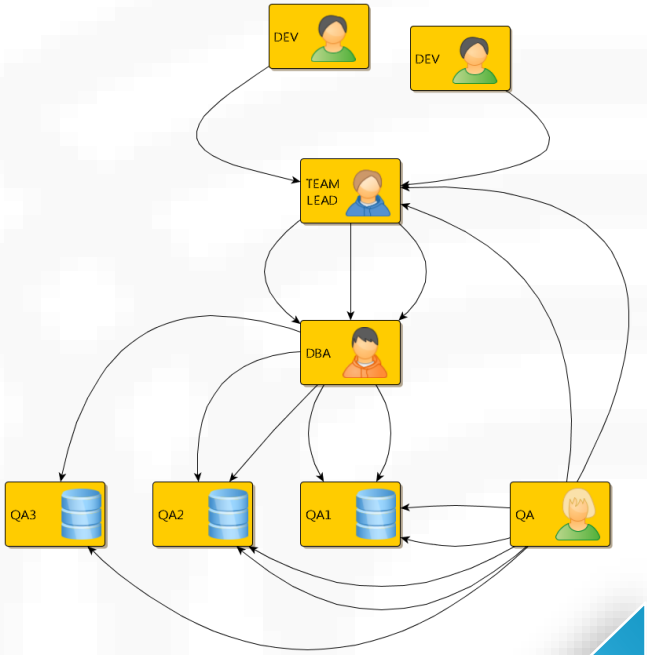


Проблема #4. Версионность

- Версия приложения == версия базы данных
- Код приложения можно деплоить многократно (stateless)
- Скрипты базы данных деплоить многократно нельзя (stateful)
- ~~Can you make a build in one step?~~ NO (Joel Spolsky)

А есть ли проблема?

- ✘ Небольшая команда
- ✘ Мало изменений в базе данных
- ✓ А можно по-другому?!
- ✓ А как?
- ✓ Написали своё



Почему не своё

- Заточено под конкретный проект
- Заточено под конкретную базу данных
- Сопровождение и развитие своими силами

Проблемы. Итого

- Много разработчиков
- Много стендов и баз данных
- Сложность ведения версий системы
- Сложность автоматизации

Необходимые требования

- Скрипты базы – это часть версии
- Применение скриптов автоматически
- Идемпотентность*

* не ругательство

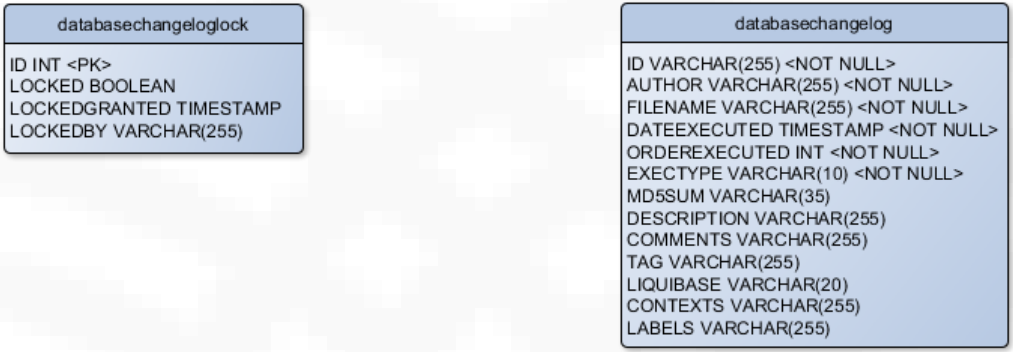
Достаточные требования

- Создание «с нуля» для любой версии системы (миграции никто не отменял)
- Защита от одновременного запуска обновлений

Почему liquibase

- По описанию удовлетворял всем требованиям
- Поддерживает чистый sql (xml, yaml, json)
- Имеет возможность обратной разработки (reverse engineering)
- Есть возможность выполнения diff баз данных
- Open-source, Java
- Самостоятельный jar-файл
- Альтернатив было не много

Внутреннее устройство




```
INSERT INTO public.databasechangeloglock (ID, LOCKED)
VALUES (1, FALSE);
```

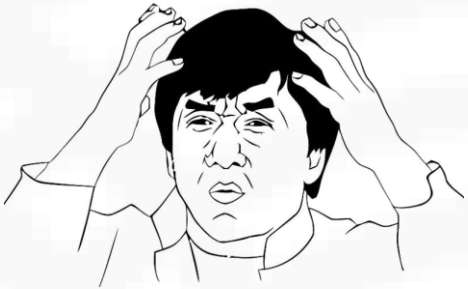
defaultSchemaName

Основные понятия Liquibase

- Changelog
- Changeset
- Preconditions

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog" xmlns:e

<preConditions>
    <dbms type="postgresql" />
    <runningAs username="postgres" />
</preConditions>

<changeSet author="balayan" id="1455019189854-1">
    <preConditions onFail="MARK_RAN">
        <not>
            <tableExists schemaName="public" tableName="pgbench_accounts" />
        </not>
    </preConditions>
    <createTable tableName="pgbench_accounts">
        <column name="aid" type="INT">
            <constraints nullable="false"/>
        </column>
        <column name="bid" type="INT"/>
        <column name="abalance" type="INT"/>
        <column name="filler" type="CHAR(84)"/>
    </createTable>
</changeSet>
```

Liquibase XML [sql]



```
<changeSet author="balayan" id="1455019189854-1">
<preConditions onFail="MARK_RAN" onFailMessage="table pgbench_accounts already exists">
    <sqlCheck expectedResult="0">select count(*) from pg_tables where tablename = 'pgbench_accounts' and schemaname = 'public'</sqlCheck>
</preConditions>
    <sql>CREATE TABLE public.pgbench_accounts (aid INT NOT NULL, bid INT, abalance INT, filler CHAR(84))</sql>
</changeSet>
```

<sql> считает “;” в конце строки как разделитель выражения

Liquibase XML [createProcedure]

```
<changeSet author="ivanov" id="20140829-03" runOnChange="true">
    <comment>task_number</comment>
    <createProcedure>
    <![CDATA[
        CREATE OR REPLACE FUNCTION public.func_get_positive_delta(a int, b int)
          RETURNS int AS
        $BODY$
        DECLARE
          vRes int;
        BEGIN
          IF a < b THEN
            vRes := b - a;
          ELSE
            vRes := a - b;
          END IF;

          RETURN vRes;
        END;
        $BODY$
        LANGUAGE plpgsql IMMUTABLE;
    ]]>
    </createProcedure>
    <sql>ALTER FUNCTION public.func_get_sum(character) OWNER TO ivanov</sql>
    <rollback>drop function func_get_sum(int, int)</rollback>
</changeSet>
```

Liquibase XML [runAlways]

```xml
<changeSet author="balayan" id="ro_grants_script-01"  runAlways="true">
    <comment>task_number</comment>
    <sql>grant connect on database liquitest to testuser</sql>
    <sql>grant usage on schema testschema to testuser</sql>
    <sql>grant select on all tables in schema testschema to testuser</sql>
    <rollback>revoke select on all tables in schema testschema from testuser</rollback>
    <rollback>revoke usage on schema testschema from testuser</rollback>
    <rollback>revoke connect on database liquitest from testuser</rollback>
</changeSet>
```

Защита от одновременного запуска

```
UPDATE public.databasechangeloglock
   SET LOCKED = TRUE,
   LOCKEDBY = 'Balayan (77.88.55.66)',
   LOCKGRANTED = '2016-02-12 19:20:21.223'
 WHERE ID = 1 AND LOCKED = FALSE;
```

INFO 12.02.16 19:32: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:33: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:33: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:33: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:33: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:33: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:33: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:34: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:34: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:34: liquibase: Waiting for changelog lock....
INFO 12.02.16 19:34: liquibase: Waiting for changelog lock....
Unexpected error running Liquibase: Could not acquire change log lock. Currently locked by Balayan (77.88.55.66) since 12.02.16 19:20

Идемпотентность

```
INSERT INTO public.databasechangelog
        (ID, AUTHOR, FILENAME,
         DATEEXECUTED, ORDEREXECUTED, MD5SUM,...)
VALUES ('1455019189854-1', 'balayan', 'mychangelog.xml',
         NOW(), 1, '7:cc5b99990f0ab9b2c25cde7b16502f6e',...);
```

```
<changeSet author="balayan" id="1455019189854-1">
    <preConditions onFail="MARK_RAN">
        <not>
            <tableExists schemaName="public" tableName="pgbench_accounts" />
        </not>
    </preConditions>
    <createTable tableName="pgbench_accounts">
        <column name="aid" type="INT">
            <constraints nullable="false"/>
        </column>
        <column name="bid" type="INT"/>
        <column name="abalance" type="INT"/>
        <column name="filler" type="CHAR(84)"/>
    </createTable>
</changeSet>
```

Once committed never changes

Идемпотентность и Обычная БД

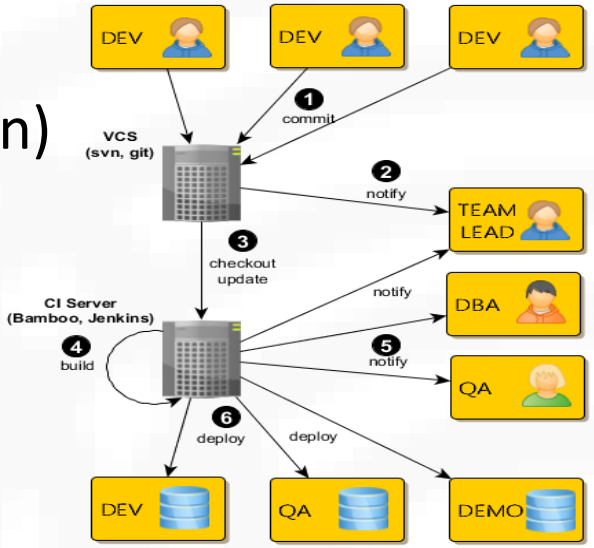
```
<changeSet id="20140902-1" author="ivanov">
    <comment>task_number</comment>
    <sql>
        ALTER TABLE SCOTT.TIGER ADD (ETYPE VARCHAR2(10));
        UPDATE SCOTT.TIGER SET ETYPE = 'N';
        COMMENT ON COLUMN SCOTT.TIGER.ETIPE IS 'Тип сущности';
    </sql>
</changeSet>
```

```
<changeSet id="20140902-1" author="ivanov">
    <comment>task_number</comment>
    <sql>ALTER TABLE SCOTT.TIGER ADD (ETYPE VARCHAR2(10));</sql>
</changeSet>

<changeSet id="20140902-2" author="ivanov">
    <comment>task_number</comment>
    <sql>UPDATE SCOTT.TIGER SET ETYPE = 'N';</sql>
</changeSet>

<changeSet id="20140902-3" author="ivanov">
    <comment>task_number</comment>
    <sql>COMMENT ON COLUMN SCOTT.TIGER.ETIPE IS 'Тип сущности';</sql>
</changeSet>
```

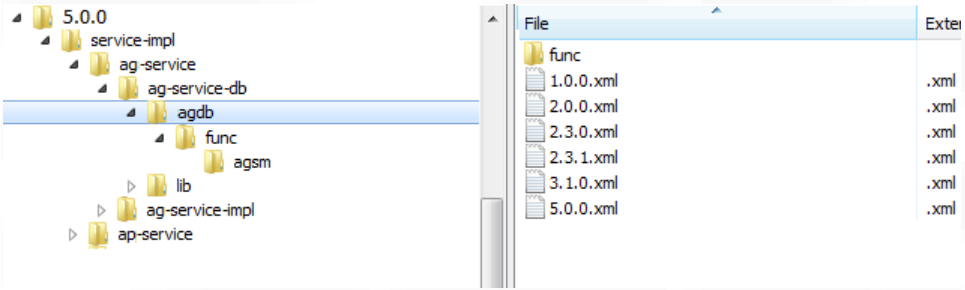
Автоматизация

- Bamboo (jenkins, hudson)
- Shell / bat

```
./liquibase   --driver=org.postgresql.Driver
                    --username=$USR
                    --
changeLogFile=master_changelog.xml
                    --
url=$URL/dbname?tcpKeepAlive=true
                    --password=$PWD
                    --logLevel=info
                    update
if [ $? -ne 0 ]
then
exit 1
fi
```



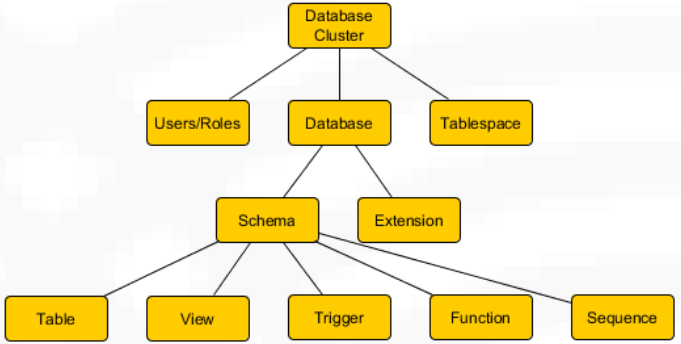
Часть версии



- Ченджлоги и исходники приложения хранятся в ветке версии
- Для каждой версии свой ченджлог
- Каждая следующая версия содержит ченджлоги всех предыдущих

Создание с нуля

- Initial_changelog (табличное пространство, база данных, пользователи) *// postgres*
- Prerun_changelog (схемы, расширения, владельцы, значения по-умолчанию)
- Master_changelog (сами изменения) *// правят разработчики*
- Postrun_changelog (права на объекты, ro-пользователи)




```
<changeSet author="balayan" id="subsystem-dbname-05" runInTransaction="false">
    <preConditions onFail="MARK_RAN" onFailMessage="Database dbname already exists">
        <sqlCheck expectedResult="0">SELECT COUNT(1) FROM pg_database
                                     WHERE datname = 'dbname'</sqlCheck>
    </preConditions>
    <comment>создание базы данных dbname</comment>
    <sql>create database dbname owner hcs_srch tablespace srch_tbs</sql>
    <rollback>DROP database dbname</rollback>
</changeSet>
```

Это ещё не всё

- Как обеспечить уникальность ченджсетов
- Как облегчить последующий анализ изменений в версии
- Проблема пересоздаваемых объектов и проблема зависимостей

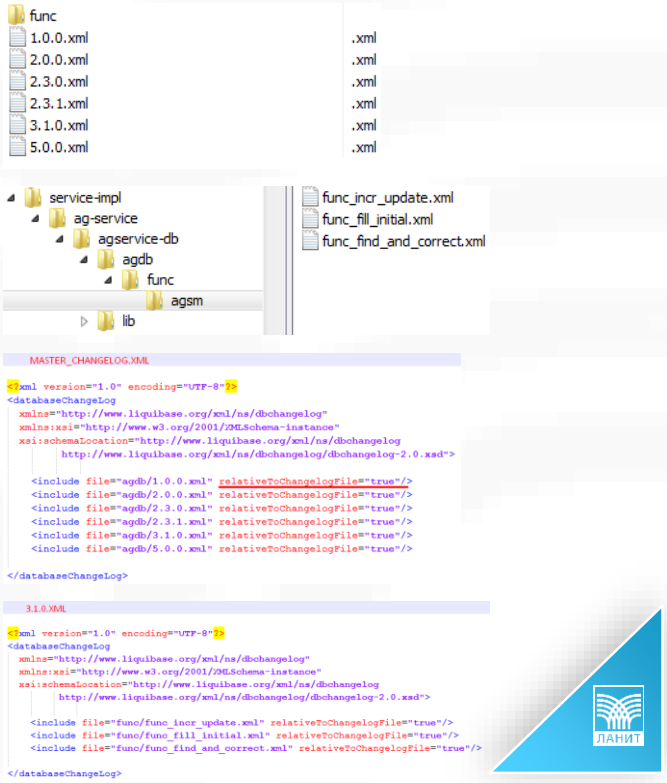
Уникальность ченджсетов

- Уникальность: author + id + filename
- Id = <date+TodayChangeNumber>

```
<changeSet id="20140902-1" author="ivanov">
    <comment>task_number</comment>
    <sql>ALTER TABLE SCOTT.TIGER ADD (ETYPE VARCHAR2(10));</sql>
</changeSet>

<changeSet id="20140902-2" author="ivanov">
    <comment>task_number</comment>
    <sql>UPDATE SCOTT.TIGER SET ETYPE = 'N';</sql>
</changeSet>

<changeSet id="20140902-3" author="ivanov">
    <comment>task_number</comment>
    <sql>COMMENT ON COLUMN SCOTT.TIGER.ETIPE IS 'Тип сущности';</sql>
</changeSet>
```

Удобство анализа (include)

- Для каждой версии создаём свой ченджлог
- Все пересоздаваемые объекты базы в отдельных ченджлогах (include)



Функции и представления

- Изменение пересоздаваемых объектов
  - Function return type
  - View column drop
- Зависимости объектов в БД

```
sql> create table t1 (f1 text);
sql> create view v1 as select * from t1;
sql> alter table t1 alter column f1 type char(5);
```

ERROR: cannot alter type of a column used by a view or rule
DETAIL: rule _RETURN on view v1 depends on column "f1"

Наш этикет

- Используем только теги <sql>, <createProcedure>
- Если есть служебные символы XML, обрамляем конструкцией <![CDATA[…]]>
- Если нужно применение ченджсета при его изменениях, используем runOnChange=“true”
- Если нужно постоянное применение ченджсета, используем runAlways=“true”
- Каждое изменение пишем в отдельном ченджсете
- Сохраненные в хранилище ченджсеты не правятся (разработчиками)
- Для каждой версии создаём свой ченджлог
- Каждая следующая версия содержит ченджлоги всех предыдущих
- Есть разделение на служебные ченджлоги и на те, которые относятся к приложению (master_changelog.xml)
- Id формируем в формате <текущая_дата>-<сегодняшний_номер_изменения>
- Все пересоздаваемые объекты в отдельных ченджлогах
- Представления перед деплоем удаляем, после – восстанавливаем
- Функции удаляем и создаём (drop and ctreate), а не пересоздаем (create or replace)

Quick start

- Скачать liquibase http://www.liquibase.org/download/index.html
- Распаковать <lbdir>
- Скачать требуемый jdbc-драйвер https://jdbc.postgresql.org/download.html
- Скопировать драйвер в каталог <lbdir>/lib
- Запустить liquibase --driver=org.postgresql.Driver --url=jdbc:postgresql://77.88.55.66:5454/liquitest --username=postgres --password=postgres --changeLogFile=generated_log.xml generateChangeLog
- updateSQL, update

Контакты

Михаил Балаян

Руководитель сектора управления данными

E-mail: balayan@lanit.ru

Skype: m.balayan