Конвертум

www.convertum.ru

# Алексей Мигуцкий

## Руководитель отдела миграции баз данных

- Имею более 8 лет опыта работы в области миграции БД
- Реализовал 20+ масштабных миграционных проектов
- Публикую статьи о миграции
- Увлекаюсь шахматами, рыбалкой
- Создаю кулинарные шедевры
- Постоянно забываю наушники в разных местах

alexey.migutskij@convertum.ru

+7 (980) 160-60-90

Конвертум

www.convertum.ru

# План

1. Проект миграции БД: этапы и сложности
2. Миграция схемы и бизнес-логики – основные сложности, решения и возможности автоматизации
3. Oracle – PostgreSQL: основные сложности и примеры конвертации.
4. SQL Server – PostgreSQL: основные сложности и примеры конвертации.
5. Изменение приложения при миграции баз данных – сложности и возможности автоматизации
6. Автоматизация проекта миграции с Конвертум

Конвертум

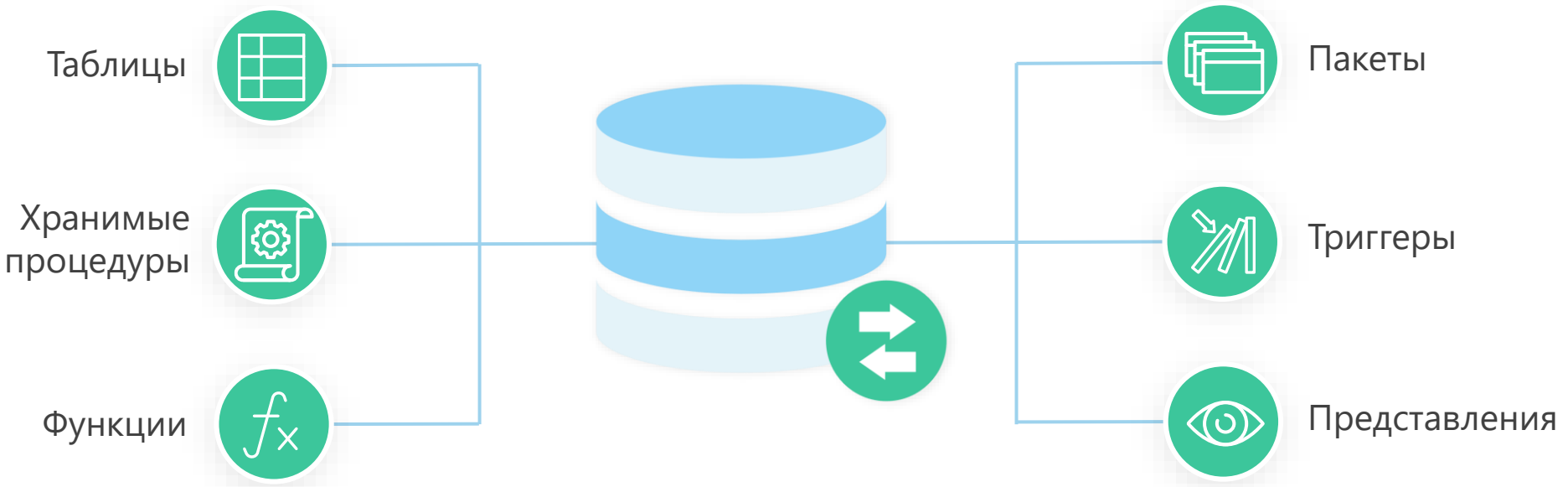
www.convertum.ru

# Проект миграции базы данных на PostgreSQL

Конвертум

www.convertum.ru

# Конвертация схемы и бизнес логики



Конвертум

www.convertum.ru

# Конвертация таблиц, индексов, партишенов, констрейнтов

## Что делаем

- Конвертация структур (таблицы)
- Конвертация констрейнтов (PK, FK, Check, Unique)
- Конвертация и создание индексов в PostgreSQL
- Конвертация дефолтных значений колонок
- Конвертация партишенов

## На что обратить внимание

- Обязательно учитывать зависимости между объектами
- Возможность указать/изменить схемы
- Возможности изменять нейминг объектов/колонок
- Возможность распределить таблицы и индексы по разным тейблспейсам
- Маппинг типов (возможность изменять локально, глобально)
- Возможность отключить/включить ключи, важно учитывать зависимости, иметь возможность накатить констрейнты и индексы на БД

Конвертум

www.convertum.ru

# Конвертация объектов бизнес-логики

## Что делаем

- Конвертация объектов
- Компиляция
- Предварительное тестирование
- Дополнительные изменения
- Финальный сбор схемы без данных

## На что обратить внимание

- Согласованная работа с конвертацией таблиц(типы данных, названия схем)
- Выбор решения под работу приложения (резалтсеты)
- Вынос логики на уровень приложения
- Иметь возможность исправить найденную логическую ошибку  сразу для всех объектов
- Не тянуть ошибки
- Делиться опытом/обучаться во время проекта

Конвертум

www.convertum.ru

# Миграция Oracle в PostgreSQL

## Сложности:

- Динамик SQL
- Коллекции (включая ассоциативные массивы)
- Нативные пакеты
- Системные объекты
- Транзакции (автономные транзакции) и совместная работа с эксепшенами
- Nested процедуры и функции
- Пакеты
- Типы, подтипы, методы типов (member/static functions). Их наследование
- Иерархические запросы
- Non Ansi-SQL Joins (+)
- Синонимы
- GOTO

# Конвертация пакетов

```
CREATE package body accounts is

type typel is record(
account_id  number(10,0),
avail_balance   number);

type account_array is table of typel index by pls_integer;

ACC_LIST account_array;
PREP_ACCOUNTS NUMBER := 15;

PROCEDURE get_list_of_prepared_account
IS
n number;
BEGIN
  for i in 1.. PREP_ACCOUNTS LOOP
    n:=ACC_LIST(i).account_id;
    dbms_output.put_line(ACC_LIST(i).account_id);
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN
   dbms_output.put_line('ERROR');
END;

PROCEDURE ADD_NEW_ACCOUNTS IS
acc_id number;
BEGIN
  for i in 1..3 LOOP
    acc_id := ACC_LIST(i).account_id;
  end loop;
  PREP_ACCOUNTS := 0;
END;

BEGIN
dbms_output.put_line('Start package ACCOUNTS');
END;
```

# Конвертация пакетов

```
CREATE package body accounts is

type typel is record(
account_id  number(10,0),
avail_balance   number);

type account_array is table of typel index by pls_integer;

ACC_LIST account_array;
PREP_ACCOUNTS NUMBER := 15;
```

# Конвертация пакетов

```
PROCEDURE get_list_of_prepared_account
IS
n number;
BEGIN
  for i in 1.. PREP_ACCOUNTS LOOP
    n:=ACC_LIST(i).account_id;
    dbms_output.put_line(ACC_LIST(i).account_id);
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN
   dbms_output.put_line('ERROR');
END;

PROCEDURE ADD_NEW_ACCOUNTS IS
acc_id number;
BEGIN
  for i in 1..3 LOOP
    acc_id := ACC_LIST(i).account_id;
  end loop;
  PREP_ACCOUNTS := 0;
END;
```

# Конвертация пакетов

```
BEGIN
dbms_output.put_line('Start package ACCOUNTS');
END;
```

Конвертум

www.convertum.ru

# Конвертация пакетов

```
CREATE package body accounts is

type type1 is record(
account_id  number(10,0),
avail_balance   number);

type account_array is table of type1 index by pls_integer;
```

```
CREATE SCHEMA  IF NOT EXISTS ACCOUNTS;

CREATE TYPE ACCOUNTS.TYPE1 AS(ACCOUNT_ID BIGINT, AVAIL_BALANCE NUMERIC);
```

Конвертум

www.convertum.ru

# Конвертация пакетов

```
ACC_LIST account_array;
PREP_ACCOUNTS NUMBER := 15;
```

```
BEGIN
dbms_output.put_line('Start package ACCOUNTS');
END;
```

```
CREATE OR REPLACE FUNCTION ACCOUNTS.INIT_GL_VAR()
RETURNS VOID LANGUAGE plpgsql
   AS $$
BEGIN
   CREATE TEMPORARY TABLE  ACCOUNTS_GL_VAR AS
   SELECT(array[]:: ACCOUNTS.TYPE1[],15):: ACCOUNTS.GL_VAR_TYPE AS SWV_GL_VAR_VAL;
   -- begin Initialization Block
   RAISE NOTICE 'Start package ACCOUNTS';
   -- end Initialization Block
RETURN;
   EXCEPTION
   WHEN SQLSTATE '42P07' THEN
      NULL;
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация пакетов

```
ACC_LIST account_array;
PREP_ACCOUNTS NUMBER := 15;
```

```
BEGIN
dbms_output.put_line('Start package ACCOUNTS');
END;
```

```
CREATE OR REPLACE FUNCTION ACCOUNTS.GET_GL_VAR()
RETURNS ACCOUNTS.GL_VAR_TYPE LANGUAGE plpgsql
   AS $$
   DECLARE
   SWV_GL_VAR  ACCOUNTS.GL_VAR_TYPE;
BEGIN
   RETURN(select SWV_GL_VAR_VAL:: ACCOUNTS.GL_VAR_TYPE from ACCOUNTS_GL_VAR);
   EXCEPTION
   WHEN OTHERS THEN
      PERFORM ACCOUNTS.INIT_GL_VAR();
      RETURN(select SWV_GL_VAR_VAL:: ACCOUNTS.GL_VAR_TYPE from ACCOUNTS_GL_VAR);
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация пакетов

```
ACC_LIST account_array;
PREP_ACCOUNTS NUMBER := 15;
```

```
BEGIN
dbms_output.put_line('Start package ACCOUNTS');
END;
```

```
CREATE OR REPLACE PROCEDURE ACCOUNTS.SET_GL_VAR(SWP_GLVAR ACCOUNTS.GL_VAR_TYPE)
LANGUAGE plpgsql
   AS $$
BEGIN
   UPDATE ACCOUNTS_GL_VAR SET SWV_GL_VAR_VAL = SWP_GLVAR;
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация пакетов

```
PROCEDURE get_list_of_prepared_account
IS
n number;
BEGIN
  for i in 1.. PREP_ACCOUNTS LOOP
    n:=ACC_LIST(i).account_id;
    dbms_output.put_line(ACC_LIST(i).account_id);
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN
   dbms_output.put_line('ERROR');
END;

PROCEDURE ADD_NEW_ACCOUNTS IS
acc_id number;
BEGIN
  for i in 1..3 LOOP
    acc_id := ACC_LIST(i).account_id;
  end loop;
  PREP_ACCOUNTS := 0;
END;
```

```
CREATE OR REPLACE PROCEDURE ACCOUNTS.GET_LIST_OF_PREPARED_ACCOUNT()
LANGUAGE plpgsql
   AS $$
   DECLARE
   SWV_GL_VAR  ACCOUNTS.GL_VAR_TYPE DEFAULT ACCOUNTS.GET_GL_VAR();
   n  NUMERIC;
BEGIN
   for i in 1 .. SWV_GL_VAR.PREP_ACCOUNTS
   LOOP
      n := SWV_GL_VAR.ACC_LIST[i].account_id;
      RAISE NOTICE '%',SWV_GL_VAR.ACC_LIST[i].account_id;
   END LOOP;
   EXCEPTION
   WHEN OTHERS THEN
      RAISE NOTICE 'ERROR';
END; $$;

CREATE OR REPLACE PROCEDURE ACCOUNTS.ADD_NEW_ACCOUNTS()
LANGUAGE plpgsql
   AS $$
   DECLARE
   SWV_GL_VAR  ACCOUNTS.GL_VAR_TYPE DEFAULT ACCOUNTS.GET_GL_VAR();
   acc_id  NUMERIC;
BEGIN
   for i in 1 .. 3
   loop
      acc_id := SWV_GL_VAR.ACC_LIST[i].account_id;
   end loop;
   SWV_GL_VAR.PREP_ACCOUNTS := 0;
   CALL ACCOUNTS.SET_GL_VAR(SWV_GL_VAR);
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация пакетов

```
get_list_of_prepared_account

PREP_ACCOUNTS
ACC_LIST(i).account_id;

ADD_NEW_ACCOUNTS
```

```
ACCOUNTS
SWV_GL_VAR  ACCOUNTS.GL_VAR_TYPE DEFAULT ACCOUNTS.GET_GL_VAR();

SWV_GL_VAR.PREP_ACCOUNTS
SWV_GL_VAR.ACC_LIST[i].account_id;

ACCOUNTS
SWV_GL_VAR  ACCOUNTS.GL_VAR_TYPE DEFAULT ACCOUNTS.GET_GL_VAR();
```

# Конвертация коллекций

```
CREATE TYPE employee AS OBJECT (
   id NUMBER,
   Name VARCHAR(300)
   );
CREATE TYPE employees_tab IS TABLE OF employee;

CREATE OR REPLACE PROCEDURE hire
(EMPLOYEES in out employees_tab, id NUMBER, Name VARCHAR) AS
NEW_EMPLOYEES employees_tab := employees_tab();
BEGIN
   EMPLOYEES.Extend(1);
   EMPLOYEES(EMPLOYEES.count) := employee(id, Name);
   FOR i IN EMPLOYEES.first..EMPLOYEES.last
   LOOP
      INSERT INTO emp_tab values (EMPLOYEES(i).id, EMPLOYEES(i).Name);
   END LOOP;
   INSERT INTO EMP_TAB SELECT * FROM TABLE(EMPLOYEES);
   NEW_EMPLOYEES.Extend(EMPLOYEES.count);
   NEW_EMPLOYEES.Delete;
END;
```

```
CREATE TYPE employee AS(id NUMERIC,Name VARCHAR(300));

-- CREATE TYPE employees_tab IS TABLE OF employee;

CREATE OR REPLACE PROCEDURE hire
(INOUT EMPLOYEES employee[] , id NUMERIC, Name VARCHAR)
LANGUAGE plpgsql
   AS $$
   DECLARE
   NEW_EMPLOYEES  employee[] default array[]:: employee[];
BEGIN
   EMPLOYEES[coalesce(array_length(EMPLOYEES,1),0)+1] := null;
   EMPLOYEES[coalesce(array_length(EMPLOYEES,1),0)] := row(id,Name);
   FOR i IN array_lower(EMPLOYEES,1) .. array_upper(EMPLOYEES,1)
   LOOP
       INSERT INTO emp_tab  values(EMPLOYEES[i].id, EMPLOYEES[i].Name);
   END LOOP;
   INSERT INTO EMP_TAB  SELECT * FROM UNNEST(EMPLOYEES);
   for i in 1 .. coalesce(array_length(EMPLOYEES,1),0) loop
      NEW_EMPLOYEES[coalesce(array_length(NEW_EMPLOYEES,1),0)+1] := null;
   end loop;
   NEW_EMPLOYEES := array[]:: employee[];
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация коллекций

```
CREATE TYPE employee AS OBJECT (
   id NUMBER,
   Name VARCHAR(300)
   );
CREATE TYPE employees_tab IS TABLE OF employee;
```

```
CREATE TYPE employee AS(id NUMERIC,Name VARCHAR(300));

-- CREATE TYPE employees_tab IS TABLE OF employee;
```

Конвертум

www.convertum.ru

# Конвертация коллекций

```
(EMPLOYEES in out employees_tab,
NEW_EMPLOYEES employees_tab := employees_tab();
```

```
(INOUT EMPLOYEES employee[]

NEW_EMPLOYEES  employee[] default array[]:: employee[];
```

Конвертум

www.convertum.ru

# Конвертация коллекций

```
EMPLOYEES.Extend(1);
EMPLOYEES(EMPLOYEES.count) := employee(id, Name);
```

```
EMPLOYEES[coalesce(array_length(EMPLOYEES,1),0)+1] := null;
EMPLOYEES[coalesce(array_length(EMPLOYEES,1),0)] := row(id,Name);
```

# Конвертация коллекций

```
EMPLOYEES.first..EMPLOYEES.last
```

```
array_lower(EMPLOYEES,1) .. array_upper(EMPLOYEES,1)
```

Конвертум

www.convertum.ru

# Конвертация коллекций

```
(EMPLOYEES(i).id, EMPLOYEES(i).Name);
```

```
(EMPLOYEES[i].id, EMPLOYEES[i].Name);
```

Конвертум

www.convertum.ru

# Конвертация коллекций

```
SELECT * FROM TABLE(EMPLOYEES);
```

```
SELECT * FROM UNNEST(EMPLOYEES);
```

# Конвертация коллекций

```
NEW_EMPLOYEES.Delete;
```

```
NEW_EMPLOYEES := array[]:: employee[];
```

# Иерархические запросы

```
SELECT
   CONNECT_BY_ROOT H.id AS root_id,
   H.parent_id,
   H.id,
   HA.Name,
   LEVEL,
   LTRIM(SYS_CONNECT_BY_PATH( H.id, '*'), '*') AS path
FROM  tabl_Hier H
JOIN  tabl_Hier_Aux HA on HA.id = H.id
START WITH h.parent_id IS NULL
CONNECT BY H.parent_id = PRIOR H.id
ORDER BY id;
```

```
WITH RECURSIVE
TabAl_cte AS
(SELECT H.id AS root_id , H.parent_id, H.id, HA.Name, 1 as LEVEL,
LTRIM(CONCAT('*',H.id),'*') AS path
FROM  tabl_Hier H
JOIN  tabl_Hier_Aux HA on HA.id = H.id
WHERE h.parent_id IS NULL
UNION ALL
SELECT TabAl_cte.root_id AS root_id, H_prev.parent_id, H_prev.id,
HA_prev.Name, TabAl_cte.LEVEL+1 as LEVEL,
LTRIM(CONCAT(TabAl_cte.path,'*',H_prev.id),'*') AS path
FROM tabl_Hier H_prev
JOIN tabl_Hier_Aux HA_prev
on HA_prev.id = H_prev.id
JOIN TabAl_cte ON H_prev.parent_id = TabAl_cte.id)
SELECT  root_id,parent_id,id,Name,LEVEL,path
FROM TabAl_cte
ORDER BY id;
```

Конвертум

www.convertum.ru

# Иерархические запросы

```
CONNECT BY H.parent_id = PRIOR H.id
```

```
TabAl_cte AS
```

```
JOIN TabAl_cte ON H_prev.parent_id = TabAl_cte.id)
```

```
FROM TabAl_cte
```

Конвертум

www.convertum.ru

# Иерархические запросы

```
LEVEL,
```

```
1 as LEVEL,
TabAl cte.LEVEL+1 as LEVEL,
LEVEL,
```

# Иерархические запросы

```
LTRIM(SYS_CONNECT_BY_PATH( H.id, '*'), '*') AS path
```

```
LTRIM(CONCAT('*',H.id),'*') AS path
```

```
LTRIM(CONCAT(TabAl cte.path,'*',H prev.id),'*') AS path
```

```
path
```

Конвертум

www.convertum.ru

# Иерархические запросы

```
CONNECT_BY_ROOT H.id AS root_id,
```

```
(SELECT H.id AS root_id
SELECT TabAl_cte.root_id AS root_id
root_id,
```

# Миграция SQL Server в PostgreSQL

## Сложности:

- Динамик
- Системные объекты (процедуры, функции, таблицы, вью)
- Резалт сеты
- Case – insensitive сравнение
- Табличные параметры и переменные
- Ретурн из процедуры
- Работа с XML/JSON
- Pivot/Unpivot
- Конвертация Дблинков
- Транзакции и TRY/CATCH

# Конвертация резалтсетов

```
CREATE TABLE result_set_table
( C1 INTEGER, C2 VARCHAR(22) )

CREATE PROCEDURE prc_result_set AS
SELECT * FROM result_set_table
```

```
--Option1 = Setof

DROP TYPE IF EXISTS prc_result_set_rs CASCADE;
CREATE TYPE prc_result_set_rs AS(c1 INTEGER, c2 VARCHAR(22));
CREATE OR REPLACE FUNCTION prc_result_set()
RETURNS SETOF prc_result_set_rs LANGUAGE plpgsql
   AS $$
BEGIN
   return query SELECT * FROM result_set_table;
END; $$;


-- Option2 = Table

CREATE OR REPLACE FUNCTION prc_result_set()
RETURNS table
(
   c1 INTEGER,
   c2 VARCHAR(22)
) LANGUAGE plpgsql
   AS $$
BEGIN
   return query SELECT * FROM result_set_table;
END; $$;


-- Option3=Refcursor

CREATE OR REPLACE PROCEDURE prc_result_set(INOUT SWV_RefCur refcursor)
LANGUAGE plpgsql
   AS $$
BEGIN
   open SWV_RefCur for
   SELECT * FROM result_set_table;
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация резалтсетов и ретурна

```
CREATE PROCEDURE [InsertProduct]
    @ProductId int,
    @Count int
    AS
BEGIN

    DECLARE @PurchPrice money;
    DECLARE @Prod_Count int, @Prod_PurchPrice money, @Spent money;
    DECLARE cur_Product CURSOR LOCAL READ_ONLY FOR
    SELECT Count, PurchPrice FROM Product WHERE Count>0

    BEGIN TRY
    SET @Spent=0;
    IF EXISTS(SELECT * from Product where ProductId=@ProductId)
         UPDATE Product SET [Count]=@Count WHERE ProductId=@ProductId;
    ELSE
         Select 'You try to add unknown Product.';

    OPEN cur_Product

    FETCH NEXT FROM cur_Product INTO @Prod_Count, @Prod_PurchPrice

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @Spent=@Spent+@Prod_Count*@Prod_PurchPrice;
        FETCH NEXT FROM cur_Product INTO @Prod_Count, @Prod_PurchPrice
      END

    CLOSE cur_Product

    SELECT * FROM Product;

    END TRY
    BEGIN CATCH
        Select 'An error occurred in procedure InsertProduct: ';
        RETURN -1;
    END CATCH
END;
```

```
CREATE OR REPLACE PROCEDURE InsertProduct(v_ProductId INTEGER, v_Count INTEGER,
INOUT SWP_Ret_Value INTEGER  DEFAULT NULL,
INOUT SWV_RefCur refcursor , INOUT SWV_RefCur2 refcursor , INOUT SWV_RefCur3 refcursor)
LANGUAGE plpgsql
   AS $$
   DECLARE
   v_PurchPrice  NUMERIC(19,4);
   v_Prod_Count  INTEGER;
   v_Prod_PurchPrice  NUMERIC(19,4);
   v_Spent  NUMERIC(19,4);
   cur_Product CURSOR  FOR
   SELECT Count, PurchPrice FROM Product WHERE Count > 0;
BEGIN
   v_Spent := 0;
   IF EXISTS(SELECT * from Product where ProductId = v_ProductId) then
      UPDATE Product SET Count = v_Count WHERE ProductId = v_ProductId;
   ELSE
      open SWV_RefCur for
      Select 'You try to add unknown Product.';
   end if;
   OPEN cur_Product;
   FETCH NEXT FROM cur_Product INTO v_Prod_Count,v_Prod_PurchPrice;
   WHILE FOUND LOOP
      v_Spent := CAST(v_Spent+v_Prod_Count*v_Prod_PurchPrice AS NUMERIC(19,4));
      FETCH NEXT FROM cur_Product INTO v_Prod_Count,v_Prod_PurchPrice;
   END LOOP;
   CLOSE cur_Product;
   open SWV_RefCur2 for SELECT * FROM Product;
   EXCEPTION WHEN OTHERS THEN
      open SWV_RefCur3 for
      Select 'An error occurred in procedure InsertProduct: ';
      SWP_Ret_Value := -1;
      RETURN;
END; $$;
```

Конвертум

www.convertum.ru

# Конвертация резалтсетов и ретурна

```
RETURN -1;
```

```
INOUT SWP_Ret_Value INTEGER  DEFAULT NULL,
```

```
SWP_Ret_Value := -1;
RETURN;
```

Конвертум

www.convertum.ru

# Конвертация резалтсетов и ретурна

```
Select 'You try to add unknown Product.';
```

```
SELECT * FROM Product;
```

```
Select 'An error occurred in procedure InsertProduct: ';
```

```
INOUT SWV_RefCur refcursor , INOUT SWV_RefCur2 refcursor , INOUT SWV_RefCur3 refcursor)
```

```
open SWV_RefCur for
Select 'You try to add unknown Product.';
```

```
open SWV_RefCur2 for SELECT * FROM Product;
```

```
open SWV_RefCur3 for
Select 'An error occurred in procedure InsertProduct: ';
```

# Табличные типы и переменные

```
create type customer as table (
    first_name varchar(64),
    last_name varchar(64),
    years_old int
)

create procedure check_customers
as
begin
declare @customers customer
declare @younger_customers customer

insert into @customers values ('John', 'Le', 48)
insert into @customers values ('Stive', 'Maison', 17)
insert into @customers values ('JOHN', 'SMITH', 25)
EXEC get_younger_customer @customers
end


create procedure get_younger_customer @custom customer readonly
as
begin
declare @cust_name varchar(150)
select top(1) @cust_name = first_name + ' ' + last_name + ' - '
        + convert(varchar, years_old) from @custom order by years_old
print @cust_name
end
```

```
CREATE TYPE customer_tt AS
(first_name VARCHAR(64)
, last_name VARCHAR(64)
, years_old INTEGER);

create or replace PROCEDURE check_customer_tts()
LANGUAGE plpgsql
   AS $$
   DECLARE
   v_customer_tts  customer_tt[];
   v_younger_customer_tts  customer_tt[];
BEGIN
  v_customer_tts := v_customer_tts || ('John','Le',48):: customer_tt;
  v_customer_tts := v_customer_tts || ('Stive','Maison',17):: customer_tt;
  v_customer_tts := v_customer_tts || ('JOHN','SMITH',25):: customer_tt;

    CALL get_younger_customer_tt(v_customer_tts);
END; $$;

create or replace PROCEDURE get_younger_customer_tt(p_customer_tts customer_tt[])
LANGUAGE plpgsql
   AS $$
   DECLARE
   v_cust_name  VARCHAR(150);
BEGIN
   select  first_name || ' ' || last_name || ' - '
   || CAST(years_old AS VARCHAR(30)) INTO v_cust_name
   from UNNEST(p_customer_tts) order by years_old LIMIT 1;
   RAISE NOTICE '%',v_cust_name;
END; $$;
```

Конвертум

www.convertum.ru

# Табличные типы и переменные

```
create type customer as table (
    first_name varchar(64),
    last_name varchar(64),
    years_old int
)
```

```
declare @customers customer
declare @younger_customers customer
```

```
@custom customer
```

```
CREATE TYPE customer_tt AS
(first_name VARCHAR(64)
, last_name VARCHAR(64)
, years_old INTEGER);
```

```
v_customer_tts  customer_tt[];
v_younger_customer_tts  customer_tt[];
```

```
(p_customer_tts customer_tt[]
```

Конвертум

www.convertum.ru

# Табличные типы и переменные

```
insert into @customers values ('John', 'Le', 48)
insert into @customers values ('Stive', 'Maison', 17)
insert into @customers values ('JOHN', 'SMITH', 25)
```

```
from @custom
```

```
v_customer_tts := v_customer_tts || ('John','Le',48):: customer_tt;
v_customer_tts := v_customer_tts || ('Stive','Maison',17):: customer_tt;
v_customer_tts := v_customer_tts || ('JOHN','SMITH',25):: customer_tt;
```

```
from UNNEST(p_customer_tts)
```

# XML-методы (exists)

```
DECLARE @xml_for_exists XML;
DECLARE @result_for_exists BIT;
SET @xml_for_exists = '<towns><town>Toronto</town><town>Ottawa</town></towns>';
SET @result_for_exists = @xml_for_exists.exist('//town[text() = ''Toronto'']')
print 'EXIST METHOD: ' + convert(varchar(30),@result_for_exists)
```

```
DECLARE
v_xml_for_exists  XML;
v_result_for_exists  BOOLEAN;
v_xml_for_exists := '<towns><town>Toronto</town><town>Ottawa</town></towns>';
v_result_for_exists := xmlexists('//town[text() = ''Toronto'']'
PASSING BY VALUE v_xml_for_exists);
RAISE NOTICE 'EXIST METHOD: %',CAST(v_result_for_exists AS VARCHAR(30));
```

# XML-методы (query)

```
DECLARE @xml_for_query XML
DECLARE @result_for_query XML
SET @xml_for_query = '<Root>
<ProductDescription ProductID="1" ProductName="Road Bike">
<Features>
  <Warranty>1 year parts and labor</Warranty>
  <Maintenance>3 year parts and labor extended maintenance is available</Maintenance>
</Features>
</ProductDescription>
</Root>'
SET @result_for_query = @xml_for_query.query('/Root/ProductDescription/Features')
print 'QUERY METHOD: ' + convert(varchar(1000),@result_for_query)
```

```
DECLARE
v_xml_for_query  XML;
v_result_for_query  XML;

v_xml_for_query := '<Root>
<ProductDescription ProductID="1" ProductName="Road Bike">
<Features>
  <Warranty>1 year parts and labor</Warranty>
  <Maintenance>3 year parts and labor extended maintenance is available</Maintenance>
</Features>
</ProductDescription>
</Root>';
v_result_for_query := XMLPARSE(content array_to_string(xpath
('/Root/ProductDescription/Features',v_xml_for_query),''));
RAISE NOTICE 'QUERY METHOD: %',CAST(v_result_for_query AS VARCHAR(1000));
```

# XML-методы (nodes)

```
DECLARE @xml_for_value_nodes xml
DECLARE @to varchar(40)
DECLARE @from varchar(40)
DECLARE @heading varchar(40)
DECLARE @messagebody varchar(40)
DECLARE @isread bit
set @xml_for_value_nodes = '<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don''t forget me this weekend!</body>
  <flag>1</flag>
</note>'

select @to = tl.[to], @from = tl.[from], @heading = tl.[heading],
@messagebody = tl.[messagebody], @isread = isread
from
(select
reqData.value('(to)[1]', 'VARCHAR(40)') as 'to',
reqData.value('(from)[1]', 'VARCHAR(40)') as 'from',
reqData.value('(heading)[1]', 'VARCHAR(40)') as 'heading',
reqData.value('(body)[1]', 'VARCHAR(40)') as 'messagebody',
reqData.value('(flag)[1]', 'BIT') as 'isread'
FROM @xml_for_value_nodes.nodes('/note') as Req(reqData)
) tl
```

```
DECLARE
v_xml_for_value_nodes  XML;
v_to  VARCHAR(40);
v_from  VARCHAR(40);
v_heading  VARCHAR(40);
v_messagebody  VARCHAR(40);
v_isread  BOOLEAN;
v_xml_for_value_nodes := '<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don''t forget me this weekend!</body>
  <flag>1</flag>
</note>';

select   tl."to", tl."from", tl.heading,
tl.messagebody, isread
INTO v_to,v_from,v_heading,v_messagebody,v_isread
from
(select "(to)" as "to",
"(from)" as "from",
"(heading)" as heading,
"(body)" as messagebody,
"(flag)" as isread
FROM XMLTABLE('/note'
PASSING v_xml_for_value_nodes
COLUMNS "(to)" VARCHAR(40) PATH '(to)[1]',
"(from)" VARCHAR(40) PATH '(from)[1]',
"(heading)" VARCHAR(40) PATH '(heading)[1]',
"(body)" VARCHAR(40) PATH '(body)[1]',
"(flag)" BOOLEAN PATH '(flag)[1]')
as Req) tl;
```

# GOTO

```
create procedure sp_GOTO_LABEL
as
declare @Counter int
    set @Counter = 1
    while @Counter < 10
        begin
            set @Counter = @Counter + 1
            IF @Counter = 4 GOTO Branch_One
            IF @Counter = 5 GOTO Branch_Two
        end
    Branch_One:
        print 'Jumping To Branch One'
        GOTO Branch_Three
    Branch_Two:
        print 'Jumping To Branch Two'
    Branch_Three:
        print 'Jumping To Branch Three'
```

```
create or replace PROCEDURE sp_GOTO_LABEL()
LANGUAGE plpgsql
   AS $$
   DECLARE
   v_Counter  INTEGER;
BEGIN
   << Branch_Three >>
   BEGIN
      << Branch_Two >>
      BEGIN
         << Branch_One >>
         BEGIN
            v_Counter := 1;
            while v_Counter < 10 LOOP
               v_Counter := v_Counter+1;
               IF v_Counter = 4 then
                  EXIT Branch_One;
               end if;
               IF v_Counter = 5 then
                  EXIT Branch_Two;
               end if;
            END LOOP;
         END;
         RAISE NOTICE 'Jumping To Branch One';
         EXIT Branch_Three;
      END;
      RAISE NOTICE 'Jumping To Branch Two';
   END;
   RAISE NOTICE 'Jumping To Branch Three';
END; $$;
```

# Case-insensitive сравнение

```
create table tlike
(id int, title varchar(20), title2 varchar(20));

select * from tlike where title = title2;
select * from tlike where title like title2;
select * from tlike where title in (title2);
```

```
-- use the citext type (case-insensitive text).
  create extension citext;
  create table tlike
  (
     id integer,
     title citext,
     title2 citext
  );

  select * from tlike where title = title2;
  select * from tlike where title ilike title2;
  select * from tlike where title in (title2);

-- You can also add "lower" for each string comparison (variables, table fields, co
  create table tlike
  (
     id integer,
     title varchar(20),
     title2 varchar(20)
  );

  select * from tlike where lower(title) = lower(title2);
  select * from tlike where title ilike title2;
  select * from tlike where lower(title) in (lower(title2));

-- Alternatively, you can also create your own COLLATIONs and use them as follows:
  CREATE COLLATION IF NOT EXISTS swcol_ci_nondet
  (provider = icu, locale = 'und-u-ks-level2', deterministic = false);
  CREATE COLLATION IF NOT EXISTS swcol_ci_det
  (provider = icu, locale = 'und-u-ks-level2', deterministic = true);

  create table tlike
  (
     id integer,
     title varchar(20)  COLLATE swcol_ci_nondet,
     title2 varchar(20) COLLATE swcol_ci_nondet
  );

  select * from tlike where title = title2;
  select * from tlike where title ilike title2 COLLATE swcol_ci_det;
  select * from tlike where title in (title2);
```

Конвертум

www.convertum.ru

# Изменение приложений при миграции баз данных

## Что делаем

- Embedded SQL конвертация
- Изменение подключений/API
- Изменения ETL-процесса
- Конвертация запросов для репорт систем
- Вынос логики с уровня БД на уровень приложения

## На что обратить внимание

- Обязательно связанная работа с конвертацией схемы
- Составление тесткейсов лучше привязывать к изменению приложения
- Обязательно иметь связь между АПП и БД командами
- Начинать с максимально сложных моментов

# Встроенный SQL в Java

```
protected String getStatement(SQLTool sql) throws SQLException {
    return "SELECT t1.c1, t1.c2 " +
        " FROM " + sql.tableName("table_b ") + " t1 " +
        " WHERE t1.c1 = ? " +
        " AND sysdate <= t1.c2 " +
        " AND t1.c2 = (SELECT MIN(t3.c2) " +
        " FROM " + sql.tableName("table_b ") + " t3, " + sql.tableName("TABLE_A ") + " t2 " +
        " WHERE t2.c1 = t3.table_a_c1 " +
        " AND t3.table_a_c1" +
        " = " +
        " (select i.table_a_c1 from table_b i where i.c1 = ?)" +
        " AND t3.c4 > ?)";
    }
```

```
protected String getStatement(SQLTool sql) throws SQLException {
        return "SELECT t1.c1, t1.c2   " +
                " FROM   " + sql.tableName("table_b ") + "   t1   " +
                " WHERE t1.c1 = ?   " +
                " AND LOCALTIMESTAMP  <= t1.c2   " +
                " AND t1.c2 =(SELECT MIN(t3.c2)   " +
                " FROM   " + sql.tableName("table_b ") + "   t3,   " + sql.tableName("TABLE_A ") + "   t2   " +
                " WHERE t2.c1 = t3.table_a_c1   " +
                " AND t3.table_a_c1  " +
                " =(select i.table_a_c1 from table_b i where i.c1 = ?)  " +
                " AND t3.c4 > ?)";
    }
```

Конвертум

www.convertum.ru

# Встроенный SQL в Java

```
tringBuilder sql = new StringBuilder();

verifyTableName(this.getTableName());
sql.append("INSERT INTO " + this.getTableName() + " (");
sql.append("id, ");
sql.append("host, ");
sql.append("add_date, ");
sql.append("mod_user, ");
sql.append("mod_date");
sql.append(") VALUES (?, ?, SYSTIMESTAMP + 3, ?, SYSTIMESTAMP)");

connection.setAutoCommit(false);
statement = connection.prepareStatement(sql.toString());

statement.setInt(1, id);
statement.setString(2, host);
statement.setString(3, this.getUser());

statement.execute();
connection.commit();
```

```
StringBuilder sql = new StringBuilder();

verifyTableName(this.getTableName());
sql.append("INSERT INTO " + this.getTableName() + " (");
sql.append("id, ");
sql.append("host, ");
sql.append("add_date, ");
sql.append("mod_user, ");
sql.append("mod_date");
sql.append(") VALUES (?, ?, LOCALTIMESTAMP+ INTERVAL ''3 day'', ?, LOCALTIMESTAMP)");

connection.setAutoCommit(false);
statement = connection.prepareStatement(sql.toString());

statement.setInt(1, id);
statement.setString(2, host);
statement.setString(3, this.getUser());

statement.execute();
connection.commit();
```

Конвертум

www.convertum.ru

# Встроенный SQL в Java

```
SYSTIMESTAMP + 3,
```

```
LOCALTIMESTAMP+ INTERVAL ''3 day'',
```

# Oracle в Java (PostgreSQL)

```
CREATE procedure sp_test_dbms_sql is
     v_cur_id    PLS_INTEGER;
     v_sql   CONSTANT       varchar2(500)
             := 'select FIRSTANAME, sysdate from EMPLOYEE where ID = :eid';
     v_num_col  PLS_INTEGER;
     v_desc_tab     dbms_sql.desc_tab;
     v_rowcount     number;
     emp_id number;
begin
    select ID into emp_id from HIRE_INFO where HIRE_DATE = TRUNC(SYSDATE);

    v_cur_id := DBMS_SQL.open_cursor;
    dbms_sql.parse             ( v_cur_id,  v_sql,     dbms_sql.native   );
    dbms_sql.describe_columns  ( v_cur_id,  v_num_col, v_desc_tab        );

    dbms_sql.bind_variable     (v_cur_id,   ':eid',    emp_id            );

    v_rowcount := dbms_sql.execute(v_cur_id);
    v_rowcount := DBMS_SQL.FETCH_ROWS (v_cur_id);

    DBMS_SQL.close_cursor(v_cur_id);
end;
```

```
Integer vCurId  = null;
final String vSql  = "select FIRSTANAME, LOCALTIMESTAMP from EMPLOYEE where ID = 
Integer vNumCol  = null;
ResultSetMetaData vDescTab  = null;
BigDecimal vRowcount  = null;
BigDecimal empId  = null;

try(PreparedStatement mStmt = mConn.prepareStatement("select ID from HIRE_INFO whe
    try(ResultSet rs = mStmt.executeQuery();) {
        if (rs.next()) {
            SQLCursorHandler.getInstance().setFound();
            empId = rs.getBigDecimal(1);
        }
        else {
            SQLCursorHandler.getInstance().setNotFound();
        }
    }
}
catch (SQLException se) {
    SQLCursorHandler.getInstance().handleExeption(se);
    throw se;
}

cstVCurId = mConn.prepareCall(vSql);
vDescTab = cstVCurId.getMetaData();
vNumCol = vDescTab.getColumnCount();
cstVCurId.setBigDecimal(":eid", empId);
cstVCurId.execute();
vRowcount = BigDecimal.valueOf(cstVCurId.getUpdateCount());
rsVCurId = cstVCurId.getResultSet();
vRowcount = BigDecimal.valueOf((rsVCurId.next() ? 1 : 0));
rsVCurId.close();
cstVCurId.close();
    select * from tlike where title ilike title2 COLLATE swcol_ci_det;
    select * from tlike where title in (title2);
```

# Oracle в Java (PostgreSQL)

```
CREATE procedure sp_test_dbms_sql is
     v_cur_id    PLS_INTEGER;
     v_sql   CONSTANT       varchar2(500)
             := 'select FIRSTANAME, sysdate from EMPLOYEE where ID = :eid';
     v_num_col  PLS_INTEGER;
     v_desc_tab     dbms_sql.desc_tab;
     v_rowcount     number;
     emp_id number;
begin
    select ID into emp_id from HIRE_INFO where HIRE_DATE = TRUNC(SYSDATE);

    v_cur_id := DBMS_SQL.open_cursor;
    dbms_sql.parse            ( v_cur_id,  v_sql,     dbms_sql.native   );
    dbms_sql.describe_columns ( v_cur_id,  v_num_col, v_desc_tab        );

    dbms_sql.bind_variable    (v_cur_id,   ':eid',    emp_id            );

    v_rowcount := dbms_sql.execute(v_cur_id);
    v_rowcount := DBMS_SQL.FETCH_ROWS (v_cur_id);

    DBMS_SQL.close_cursor(v_cur_id);
end;
```

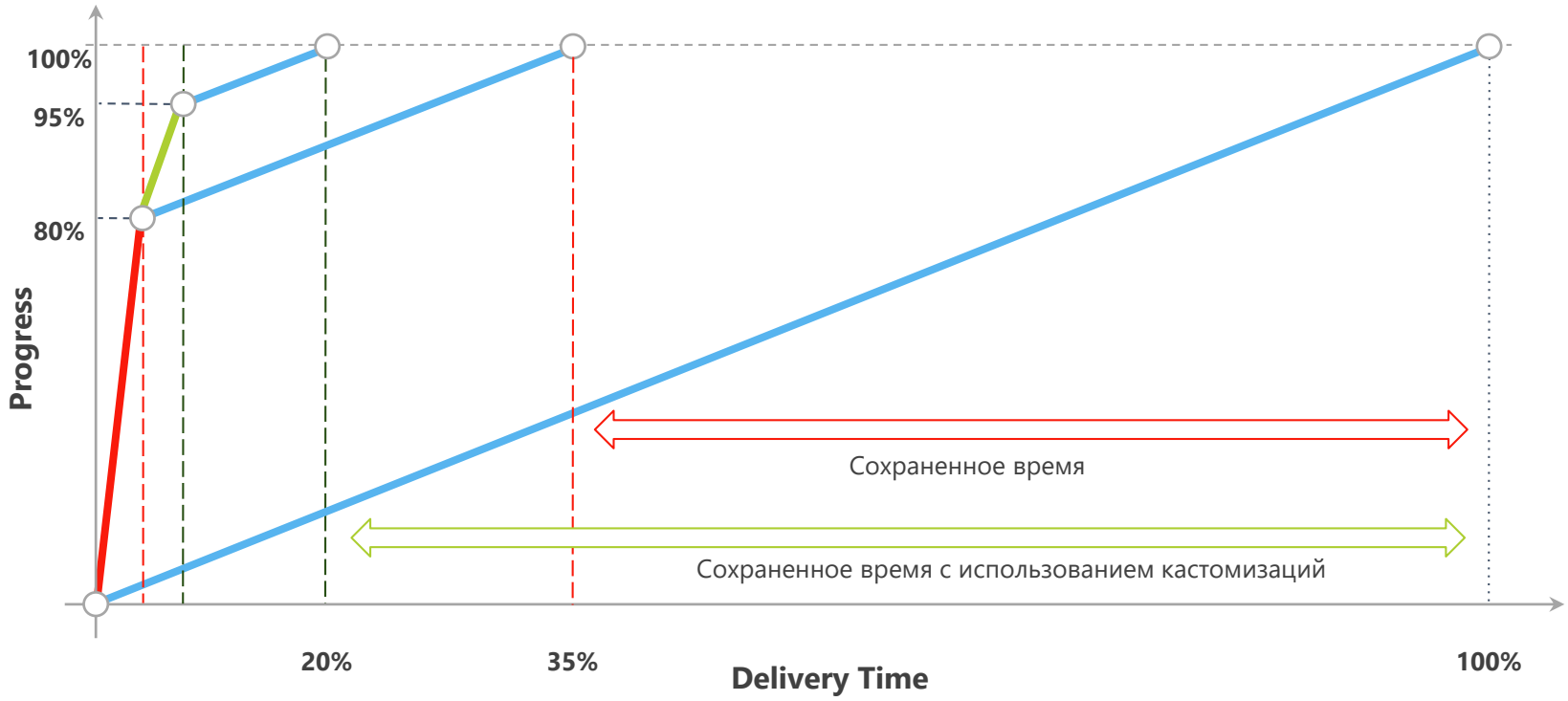
# Oracle в Java (PostgreSQL)

```
Integer vCurId  = null;
final String vSql  = "select FIRSTANAME, LOCALTIMESTAMP from EMPLOYEE where ID = :eid";
Integer vNumCol  = null;
ResultSetMetaData vDescTab  = null;
BigDecimal vRowcount  = null;
BigDecimal empId  = null;

try(PreparedStatement mStmt = mConn.prepareStatement("select ID from HIRE_INFO where HIRE_DATE = TRUNC(LOCALTIMESTAMP)");) {
    try(ResultSet rs = mStmt.executeQuery();) {
        if (rs.next()) {
            SQLCursorHandler.getInstance().setFound();
            empId = rs.getBigDecimal(1);
        }
        else {
            SQLCursorHandler.getInstance().setNotFound();
        }
    }
}
catch (SQLException se) {
    SQLCursorHandler.getInstance().handleExeption(se);
    throw se;
}

cstVCurId = mConn.prepareCall(vSql);
vDescTab = cstVCurId.getMetaData();
vNumCol = vDescTab.getColumnCount();
cstVCurId.setBigDecimal(":eid", empId);
cstVCurId.execute();
vRowcount = BigDecimal.valueOf(cstVCurId.getUpdateCount());
rsVCurId = cstVCurId.getResultSet();
vRowcount = BigDecimal.valueOf((rsVCurId.next() ? 1 : 0));
rsVCurId.close();
cstVCurId.close();
```

# Автоматизация миграции схемы и бизнес-логики




Ручная конвертация

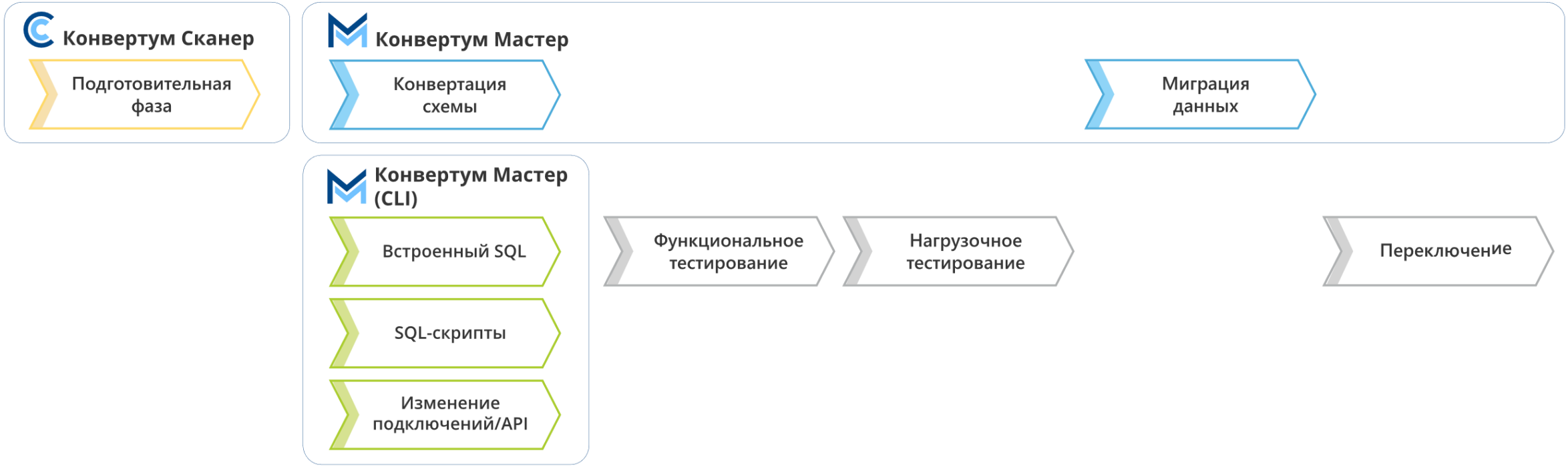
Использование Конвертум

Использование катомизаций

Скрость ручной конвертации - **300-400** строк в день

Скорость конвертации после использования инструмента - **3000-5000** строк в день

# Автоматизация процесса миграции с Конвертумом



Конвертум

www.convertum.ru

# Гибкая модель взаимодействия

## Лицензия Конвертум

- Бесплатная демолицензия на 30 дней
- Бесплатное ПО для оценки
- Бесплатная консультация специалиста

Попробовать бесплатно!

## Услуги по миграции

- Полный цикл услуг (от оценки исходной БД до введения в эксплуатацию целевой)
- Минимизация рисков
- Качественный результат
- **Сопутствующие услуги**
  Рекомендации по целевой технологии, аудит баз данных, аудит приложений, рефакторинг