FlexSoft JDP

Java Development Platform

Перенос бизнес-логики с Oracle PL/SQL на Java и её исполнение без Oracle

```
for I in C_OP(P_BRANCH, P_OP_NO)
loop
  V_MFO := MGC_ENV.GET_BANK_BIC;
```

JDP

```
while (cOp.fetchInto(i)) {
  vMfo.set(mgcEnv.getBankBic());
```

Включено в реестр российского ПО: запись № 36141 от 17.11.2026

Сколько бизнес-логики в вашей системе живёт в PL/SQL и годами ждёт, пока до неё дойдут руки?

Пакеты, процедуры и триггеры копились годами. Именно этот код держит систему на Oracle.

Переписать PL/SQL вручную кажется простой задачей

Так бывает и со своей разработкой, и с тиражной системой, для которой поставщик не предложил пути миграции

НЕОБОСНОВАННОЕ

Ожидание

«Наберём Java-команду и перепишем процедуры одну за другой»



быстро, линейно, предсказуемо

СУРОВАЯ

Реальность

Меняется не только синтаксис, меняется среда исполнения

- Семантика NULL и неявные преобразования типов
- Состояние пакетов и сессий
- Транзакционное поведение
- Производительность под нагрузкой
- **Регрессионное тестирование**

Без проверки расхождения легко пропустить

JDP автоматизирует перенос кода и даёт среду, где он работает как в Oracle



01

Анализ

До переноса JDP разбирает PL/SQL и SQL и показывает, что переносится автоматически, а что потребует рефакторинга

02

Конвертация

PL/SQL переводится в Java-классы. На офисном компьютере ЗО КИИ (1,1 млн строк) за 15 минут, весь *FXL* (более 4 млн строк) за 26 минут. Схему и данные переносит утилита

03

Исполнение

Java-код работает в среде DAF на любой платформе на базе Apache Ignite

Приложение не нужно дробить на сервисы до переноса: после него оно работает в кластере и масштабируется горизонтально

Вычисления в памяти	Platform V DataGrid, FlexSoft In-Memory Data Platform или другая платформа на базе Apache Ignite
Хранение	Platform V Ocean DB, YDB, Pangolin или FlexSoft In-Memory Data Platform

FlexSoft In-Memory Data Platform закрывает оба слоя

Функции переносятся по одной, перенесённую можно вернуть

На примере АБС: старая и новая система какое-то время работают вместе, каждый этап проверяется и принимается отдельно



Общие данные синхронизированы: обе системы работают с одними и теми же данными

Когда все функции перенесены и приняты, Oracle выключается

Как выглядит результат конвертации

Фрагмент пакета *FXL* в упрощённом виде: выборка ключей в коллекцию и удаление строк по этой коллекции с возвратом ключей. Поведение проверяется автоматически: исполнение в Oracle сравнивается с исполнением Java-кода

PL/SQL

```
select pa_id bulk collect into v_Id_A1
  from MGC_CL_PA p
 where client_id = p_Client_Id;

delete from MGC_CL_PA_ACCOUNTS a
 where a.pa_id member of v_Id_A1
   and a.acct_id member of p_Acct_Id
return pa_id bulk collect into v_Id_A2;
```

Java после JDP

```
QUERY = """
  SELECT pa_id FROM MGC_CL_PA p
  WHERE client_id = :P_CLIENT_ID """;
cm().selectBulkCollectInto(
    new ImplicitSelect8(), vIdA1);
@CursorBind("V_ID_A1") final CorNumberA
    bindVIdA1 = corNumberA().set(vIdA1);
QUERY = """
  SELECT 1 FROM MGC_CL_PA_ACCOUNTS a
  WHERE a.pa_id MEMBER OF :V_ID_A1
    AND a.acct_id MEMBER OF :P_ACCT_ID """;
DmlOperationContext implicitDml9 = cm()
    .executeDeleteSaveContext(
        "MGC_CL_PA_ACCOUNTS", new ImplicitDml9());
implicitDml9.returnInto(" pa_id ", vIdA2);
```

1. Выборка в коллекцию BULK COLLECT стала одним вызовом: selectBulkCollectInto
2. Коллекции PL/SQL уходят в SQL параметрами: MEMBER OF работает как в Oracle
3. Удаление с возвратом ключей в коллекцию сохранено: returnInto

Команда продолжает писать на PL/SQL, а в продуктиве работает Java



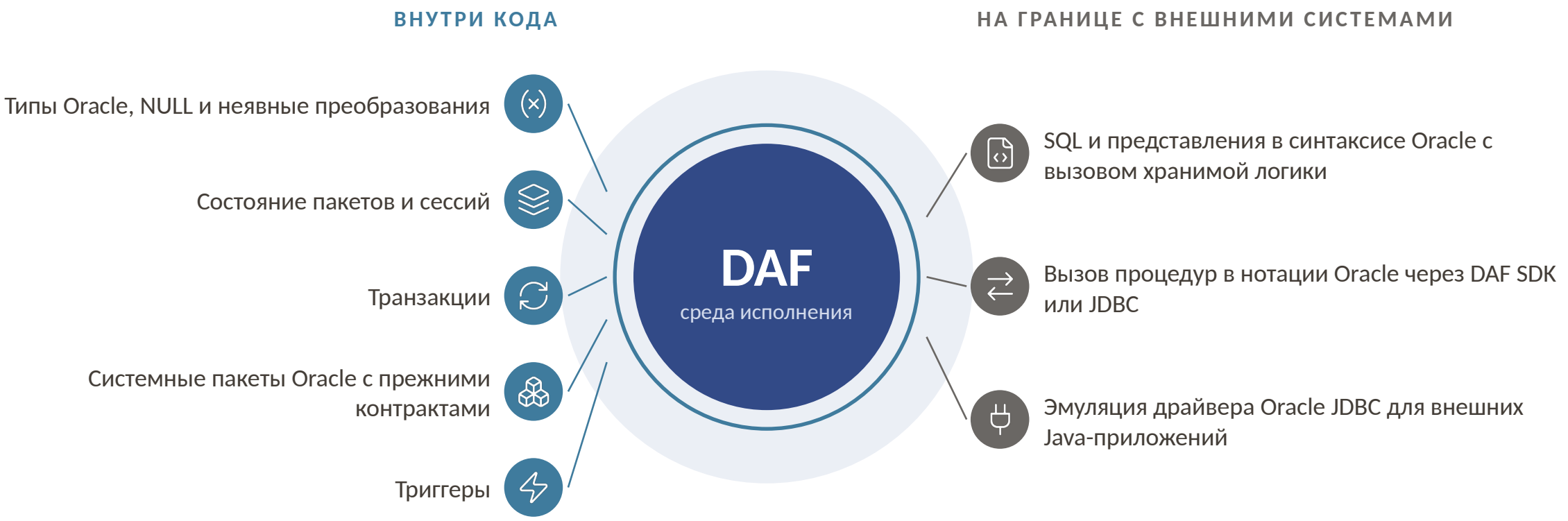
Для команды не меняется

- Язык и привычные приёмы PL/SQL
- Знание бизнес-логики и кода
- Порядок разработки и выпуска

JDP не разовый конвертер

Конвертация и среда исполнения встраиваются в разработку: команда не чувствует смены стека

Поведение Oracle сохраняется внутри кода и на границе с внешними системами



Что в вашем легаси-коде потребует рефакторинга*, видно из отчёта диагностики ещё до переноса

* Поддержка редких конструкций Oracle расширяется с каждой версией JDP. Единичные сложные случаи, например вставку в подзапрос, проще переписать вручную: в ЗО КИИ это меньше тысячи строк из 1,1 млн.

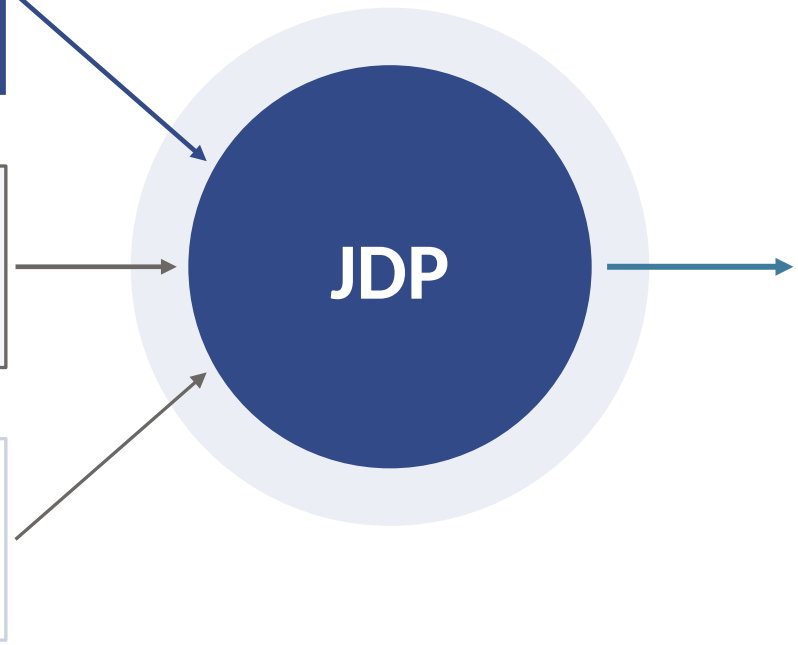
Платформу можно адаптировать под другую легаси-СУБД

Сейчас JDP работает с Oracle PL/SQL. Под другой диалект SQL платформа адаптируется в рамках проекта

Oracle PL/SQL
поддерживается

MS SQL, T-SQL
адаптация под проект

Другие SQL-СУБД
архитектура позволяет, обсуждаем



НА ВЫХОДЕ

Java-код в среде DAF

- Отчёт диагностики до переноса
- Автоматическая конвертация
- Исполнение с поведением исходной СУБД

Мы прошли этот путь на своём продукте

Модули АБС *FXL* переведены на отечественный стек, в версии *FXL* 3.0, с помощью JDP

FXL 2.0

модульная архитектура на Oracle и PL/SQL

JDP

автоматическая конвертация

НОВОЕ ПОКОЛЕНИЕ

FXL 3.0

тот же код на Java в модульной space-based архитектуре

```
if I.CLASS_OP like 'ПК_ПОСТ_%' then
```

→

```
if (i.classOp.like("ПК_ПОСТ_%").isTrue()) {
```

Заказчикам предлагаем тот же инструмент и ту же среду исполнения, на которых работает наш продукт

Независимый нагрузочный тест *FXL* 3.0 после переноса

Кластер масштабируется горизонтально, нагрузка распределяется линейно и обеспечивает высокую производительность на скромном оборудовании. RPO = 0 и RTO = 0. Это показал независимый тест *FXL* 3.0, полученной с использованием JDP.

1 890

документов в секунду в среднем, с проверками и проводками

184 мс

время создания документа, 95-й перцентиль

RPO = 0
RTO = 0

при аварийной потере узлов под нагрузкой: данные не потеряны, обработка продолжилась

УСЛОВИЯ ТЕСТА

- Независимая лаборатория «Перфоманс Лаб» по заказу ФлексСофт. Отчёт ред. 1.4 от 06.02.2026 передаём по запросу
- *FXL* 3.0: три узла Platform V DataGrid и три узла YDB
- Параллельно: выгрузка 115 500 документов в Банк России (17 мин 52 с) и загрузка с обработкой 100 000 документов (25 мин)
- Внутренний тест ФлексСофт: FXL 3.0 на Platform V DataGrid и Platform V Ocean DB, по три узла, около 1 400 документов в секунду; при аварийном отключении сервера данные не потеряны и работа не остановилась

СТРЕССОВЫЕ СЦЕНАРИИ

Узел добавлялся в кластер под нагрузкой без заметной просадки производительности

Жёсткий вывод сервера с узлами вычислений и СУБД под нагрузкой без потери данных. Прерванные в этот момент запросы повторяет вызывающая система

Успешный пилот в РТК

За 3 месяца специалисты «РТК ИТ» перевели модуль учёта трафика выделенных каналов своей биллинговой системы с Oracle на Platform V DataGrid и Platform V Pangolin DB

ЧТО ПЕРЕНЕСЕНО

10 тыс. строк

кода на PL/SQL переведены в Java: 23 объекта (пакеты, функции, типы)

14 млн строк

данных в 38 таблицах и представление

420

последовательностей: все, что были в системе

КАК ШЁЛ ПИЛОТ

1. **Развернули платформу**

   Специалисты заказчика по нашим инструкциям развернули FlexSoft JDP и кластер Platform V DataGrid из трёх узлов с RPO = 0 и RTO = 0

2. **Перенесли данные и код**

   Утилитами FlexSoft JDP перенесли таблицы и последовательности из Oracle и перевели код PL/SQL в Java

3. **Дополнили платформу**

   Редкие конструкции Oracle, найденные в коде модуля, мы добавили в FlexSoft JDP

4. **Проверили модуль**

   Функциональное тестирование по подготовленному сценарию прошло успешно

Исходный код заказчику дорабатывать не пришлось: недостающие конструкции вошли в платформу

Срок зависит от сложности кода, поэтому начинаем с диагностики

01

Обследование

Код, окружение, целевая конфигурация; отчёт диагностики JDP

02

Пилотный перенос

Фрагмент выбираем по итогам обследования

03

Проверка

Поведение, интеграции и нагрузка на фрагменте

04

План перехода

Дальнейший перенос, запуск и сопровождение

◆ решение заказчика после каждого этапа

Команда РТК справилась своими силами. Проверьте JDP на своём коде в пилоте, поможем на каждом шаге

Следующий шаг: архитектурная сессия по вашей системе

ОТ ВАС

- Описание системы и где в ней бизнес-логика
- Ключевые интеграции
- Текущая нагрузка и требования к доступности
- Эксперты по приложению, Oracle и эксплуатации

РЕЗУЛЬТАТ СЕССИИ

- Оценка применимости JDP к вашему коду
- Список открытых вопросов
- Предложение по фрагменту для пилотного переноса

Исходные коды JDP могут предоставляться в рамках отдельного соглашения. Передавать всю кодовую базу до сессии не нужно.

Приложение. Пример 1: триггер на изменение поля

Строчный триггер before update по одному полю. Фрагмент в упрощённом виде: без полей класса

PL/SQL

```
create or replace trigger
  SCS_DM_DOCUMENTS_BUR_VAL_DATE
  before update of value_date on scs_dm_documents
  for each row
```

Java после JDP

```
@Trigger(tableName = "SCS_DM_DOCUMENTS",
  eventTypes = { Trigger.EventType.UPDATE },
  columnNames = { "VALUE_DATE" },
  whenType = Trigger.WhenType.BEFORE)
public class ScsDmDocumentsBurValDate
    extends Service
    implements
      TriggerRowLevel<ScsDmDocumentsRowType> {
  @Override
  public void onEvent(
      ScsDmDocumentsRowType oldValue,
      ScsDmDocumentsRowType newValue,
      TriggerContext triggerContext) {
```

1. Триггер становится классом Java
2. Таблица, событие, поле и момент срабатывания описаны аннотацией @Trigger
3. Строчный триггер получает старое и новое значение строки как типизированные объекты

Приложение. Пример 2: обновление с возвратом значения

UPDATE с RETURNING INTO и проверкой SQL%ROWCOUNT. Фрагмент сокращён: показаны не все параметры и условия

PL/SQL

```
update PCF_CB_CALC_H
   set rstr_h_no = v_rstr_h_no
      ,rstr_e_no = v_rstr_e_no
 where agr_no = rec.agr_no
   and prod_id = rec.prod_id
   and start_date = v_start_date
   and end_date = v_end_date
   and calc_amount is null
returning id into v_calc_id;
if sql%rowcount = 0
```

Java после JDP

```
@CursorBind("V_RSTR_H_NO")
final NumberVar bindVRstrHNo =
    number().set(vRstrHNo);
// так же ещё 5 параметров запроса
@CursorQuery
public final static String QUERY = """
  SELECT :V_RSTR_H_NO new_rstr_h_no,
         :V_RSTR_E_NO new_rstr_e_no
  FROM PCF_CB_CALC_H WHERE agr_no = :REC_AGR_NO
    AND ... AND calc_amount IS NULL """;
DmlOperationContext implicitDml2 = cm()
    .executeUpdateSaveContext("PCF_CB_CALC_H",
        new ImplicitDml2());
implicitDml2.returnInto(" id ", vCalcId);
if (cm().rowCount().equal(0).isTrue()) {
```

1. SET и WHERE переносятся запросом с именованными параметрами
2. RETURNING INTO сохранён: id возвращается в переменную
3. SQL%ROWCOUNT доступен как cm().rowCount() с той же проверкой

Приложение. Пример 3: массовая вставка FORALL

FORALL с INSERT по коллекциям p_Cache. Фрагмент сокращён: из 11 полей показаны два

PL/SQL

```
forall i in 1 .. p_Cache.In_Cache
  insert into DPS_AGR_CALC_INT
    (Agreement_Id
    ,Calc_Amount
    , ...)
  values
    (p_Cache.Agreement_Id(i)
    ,nvl(p_Cache.Calc_Amount(i), 0)
    , ...);
```

Java после JDP

```
try (var bulkOperationContext = cm()
    .bulkContextManager()
    .createBulkOperationContext(BulkMode.BASE)) {
 for (var i : range(1, pCache.inCache())) {
  @CursorBind("P_CACHE_AGREEMENT_ID_I_")
  final NumberVar bindPCacheAgreementId =
    number().set(pCache.agreementId().get(i));
  // так же ещё 10 полей
  QUERY = """
    INSERT INTO DPS_AGR_CALC_INT
    (Agreement_Id, Calc_Amount, ...) VALUES
    (:P_CACHE_AGREEMENT_ID_I_,
     nvl(:P_CACHE_CALC_AMOUNT_I_, 0), ...) """;
  bulkOperationContext.executeInContext(i,
    () -> cm().executeDml(new ImplicitInsert0()));
```

1. FORALL стал контекстом массовой операции и циклом по индексам коллекции
2. INSERT переносится запросом с именованными параметрами, NVL как в Oracle
3. Каждая строка исполняется внутри контекста массовой операции: executeInContext

Приложение. Пример 4: ошибки массовой операции

Разбор SQL%BULK_EXCEPTIONS после массовой вставки из примера 3. Фрагмент сокращён: в обеих версиях опущены два вызова Set_Exception

PL/SQL

```
exception
  when others then
    declare
      v_Err_Code number := sqlcode;
      v_Err_Text varchar2(2000) := sqlerrm;
    begin
      if sql%bulk_exceptions.Count > 0 then
        COR_DCM_UTIL.Add_Exception(
          sql%bulk_exceptions(1).Error_Index,
          sql%bulk_exceptions(1).Error_Code);
      end if;
      COR_TRACE.Error_Handler(
        v_Err_Code, v_Err_Text);
    end;
```

Java после JDP

```
} catch (ApplicationException e) {
 try (var err = createErrorContext(e)) {
  if (err.isExceptionEquals(OTHERS)) {
   final NumberVar vErrCode =
       number().set(sqlcode());
   final Varchar2Var vErrText =
       varchar2(2000).set(sqlerrm());
   if (cm().bulkExceptions().count()
         .moreThan(0).isTrue()) {
    corDcmUtil.addException(cm().bulkExceptions()
        .get(var(1)).errorIndex().toNumber(),
      cm().bulkExceptions().get(var(1))
        .errorCode().toNumber(), defaultValue());
   }
   corTrace.errorHandler(of(vErrCode),
       of(vErrText), defaultValue());
```

1. WHEN OTHERS и переменные ошибки переносятся с теми же типами и значениями
2. SQL%BULK_EXCEPTIONS доступен как cm().bulkExceptions(): номер итерации и код ошибки
3. Общий обработчик ошибок вызывается так же, как в PL/SQL

Приложение. Пример 5: редкий случай отмечен заранее

Типовой BULK COLLECT JDP переводит сам (пример на слайде 6). Здесь редкое сочетание: выборка по флагу из одного из двух курсоров. JDP не пропускает его молча: помечает в коде и показывает в отчёте диагностики до переноса

PL/SQL

```
if cor_sys.Get_Boolean(
     p_Property_Name => 'DPS_C_INMEMORY_QUERY',
     p_Default => false)
then
  open cur_Data_mem(p_Beg_Date, p_End_Date);
  fetch cur_Data_mem bulk collect
    into v_Turns, v_MT_A;
  close cur_Data_mem;
else
  open cur_Data(p_Beg_Date, p_End_Date);
  fetch cur_Data bulk collect
    into v_Turns, v_MT_A;
  close cur_Data;
end if;
```

Java после JDP

```
if (corSys.getBoolean(var("DPS_C_INMEMORY_QUERY"),
        of(FALSE)).isTrue()) {
  curDataMem.set(cm().open(new CurDataMem(),
      pBegDate, pEndDate));
  /* curDataMem.fetchInto(vTurns, vMtA);
     -- REFACTORING REQUIRED */
  curDataMem.close();
} else {
  curData.set(cm().open(new CurData(),
      pBegDate, pEndDate));
  /* curData.fetchInto(vTurns, vMtA);
     -- REFACTORING REQUIRED */
  curData.close();
}
```

1. Условие и параметры вызова переведены автоматически
2. Открытие курсора с параметрами тоже переведено
3. Выборку JDP пометил REFACTORING REQUIRED. В наших переносах такие случаи единичные