PostgresPro

Инструменты и технологии миграции с СУБД Oracle на Postgres Pro Enterprise

Игорь Мельников
i.melnikov@postgrespro.ru

Миграция – это не просто замена одной СУБД на другую

PostgresPro

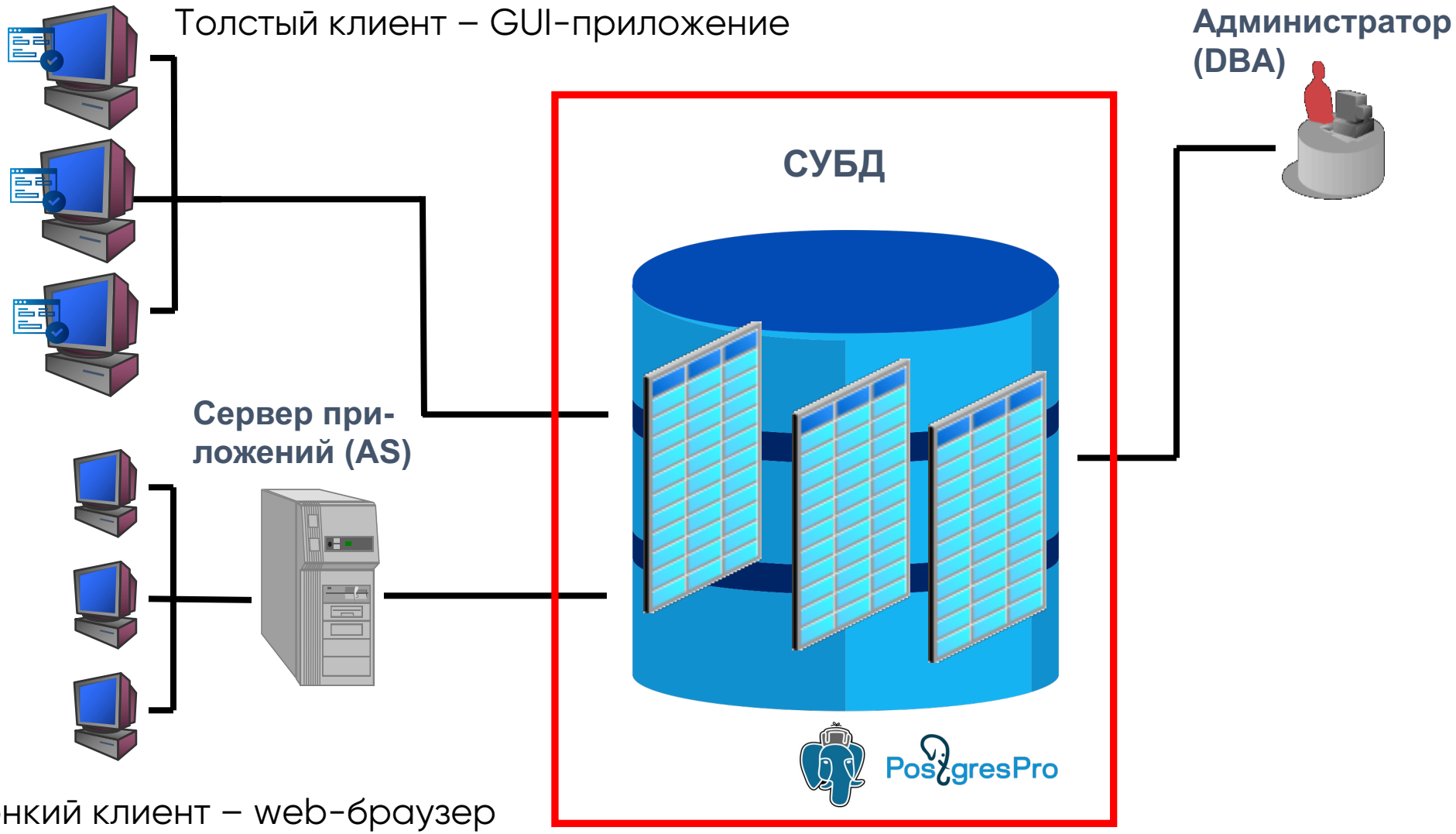
Каждая СУБД имеет свои **особенности**

- Приложение должно быть адаптировано к конкретной СУБД
- Замена СУБД требует изменения бизнес-приложения
- Требуется изменить код приложения и формат данных
- Нужны другие инструменты для обеспечения жизненного цикла БД (backup, monitoring, tuning, и т.д.)
- В PostgreSQL многие технологии реализованы по другому, чем в Oracle или MS SQL Server

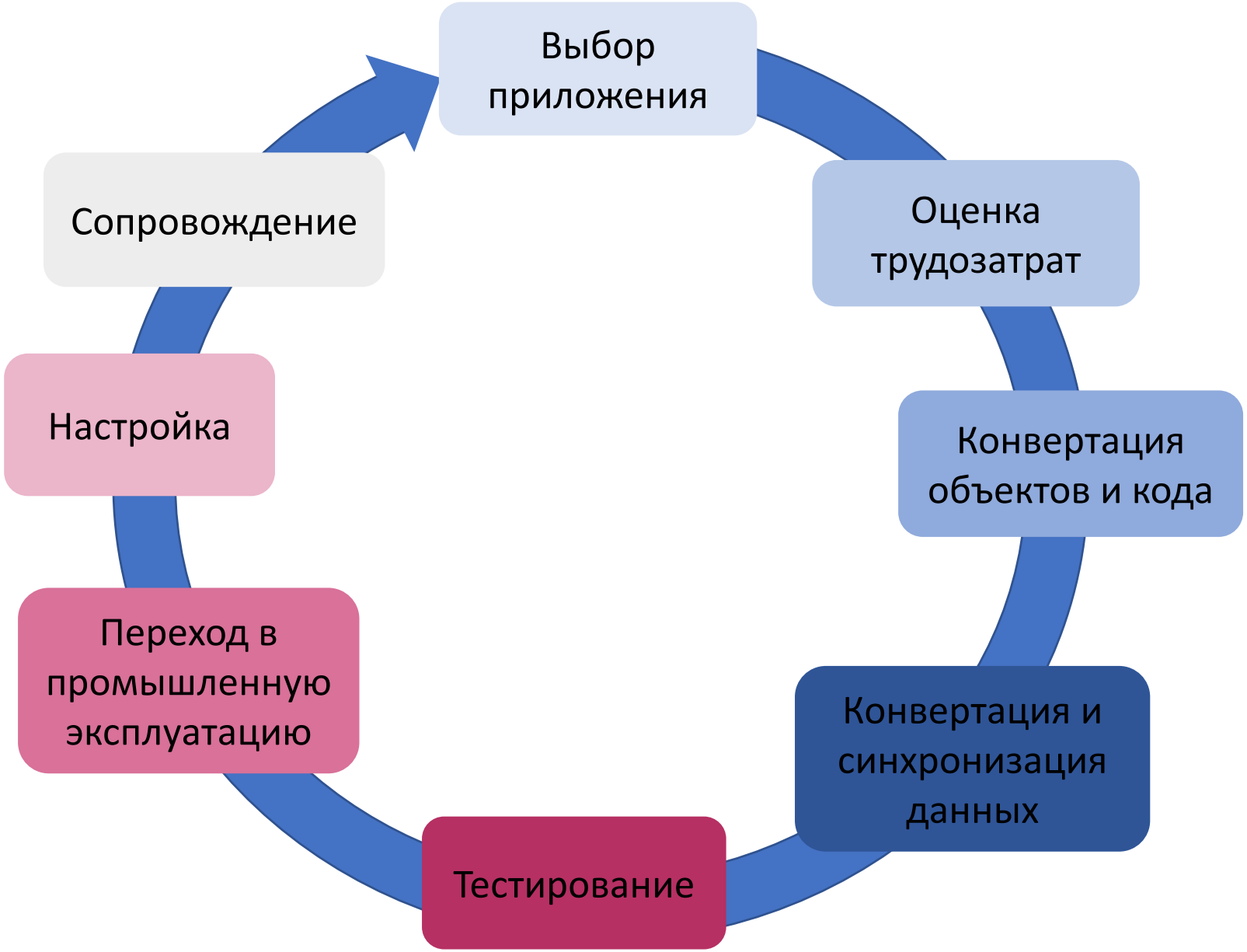
PostgreSQL – **классическая реляционная СУБД**

- Близка к СУБД Oracle по технологиям и методам работы
- Обеспечивает высокую надежность, производительность и масштабируемость
- Имеются много инструментов для автоматизации миграции с Oracle и MS SQL Server
- Имеет множество успешных внедрений
- Много успешных проектов по миграции с коммерческих СУБД

Миграция компонентов приложения



Этапы проекта по миграции – жизненный цикл



- Выбор приложения
- Оценка трудозатрат
- Конвертация объектов и кода
- Миграция данных
- Тестирование
- Переход в production
- Оптимизация и тюнинг
- Сопровождение

План

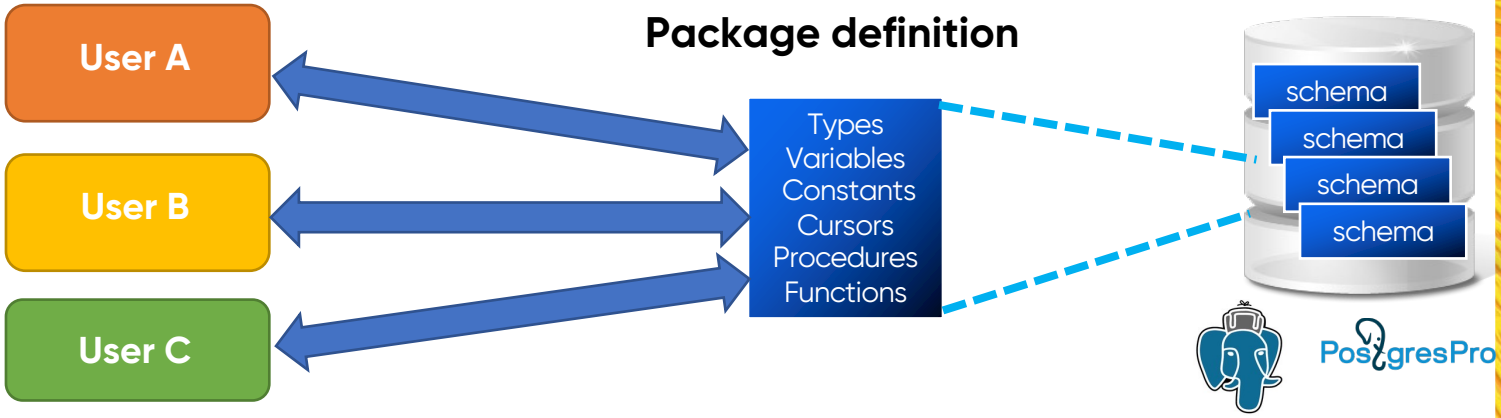
- Пакеты 2.0 (“Oracle like packages”)
- Поддержка коллекций в pg_variables
- Поддержка внешних файлов (BFile)
- PG Pro Superfile – новый тип LOB-данных
- Новые встроенные пакеты DBMS_APPLICATION_INFO, UTL_SMTP, UTL_MAIL, UTL_HTTP и DBMS_LOB
- Автоматическая конвертация кода – ora2pgpro 2.0
- Планы по развитию

PostgresPro

Поддержка пакетов (Пакеты 2.0)

Поддержка пакетов PL/pgSQL в Postgres Pro Ent 15:

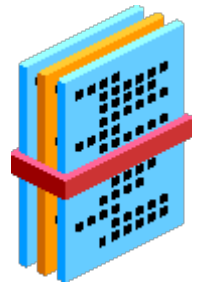
- Пакет является отдельной схемой
- Поддерживается функции инициализации – функция с фиксированным именем “***__init__***” – вызывается в момент первого обращения к пакету
- Поддерживаются глобальные переменные – все переменные объявленные в процедуре инициализации “***__init__***”
- Поддерживаются обращение к элементам других пакетов (схем), в том числе к глобальным переменным!



Пример пакета в СУБД Postgres Pro Ent 15

```
CREATE  OR  REPLACE  PACKAGE sample_pkg

 CREATE TYPE  THostInfo AS
   (
    Name         VARCHAR(32),
    Description VARCHAR(256)
   )

 CREATE FUNCTION __init__() RETURNS VOID AS $$
 DECLARE
   v_g_host_name VARCHAR(50);
 BEGIN
  v_g_host_name := get_default_host_name();
 END;
 $$;
```

• Функция инициализации пакета: все объявленные в ней переменные являются глобальными переменными пакета



Использование пакета в Postgres Pro Ent 15

```
DO $$
#import sample_pkg
BEGIN
  RAISE NOTICE 'Hostname => %', sample_pkg.v_g_host_name;
END;
$$;

NOTICE:  Hostname => localhost
DO
```

```
SELECT sample_pkg.get_default_host_name();
 get_default_host_name
-----------------------
 localhost
(1 строка)
```

- Прагма для импорта пакета
- Доступ к глобальной переменной пакета
- Вызов функции пакета в запросе

Приватность элементов пакета в PG Pro Ent 16

PostgresPro

PG Pro Ent 16

Поддержка **приватности подпрограмм** пакета

- Новая директива PL/pgSQL: *#PRAGMA PRIVATE*
- Указывается в заголовке подпрограммы
- При попытке вызова такой подпрограммы вне пакета – вызывается exception

Поддержка **приватности глобальных переменных** пакета

- Новая директива: PL/pgSQL: *#PRAGMA EXPORT* <имя переменной n1>, …
- Все переменные публичные (поведение по умолчанию): *#PRAGMA EXPORT ON*
- Все переменные приватные: *#PRAGMA EXPORT OFF*

Пример: приватность подпрограмм пакета

PostgresPro

PG Pro Ent 16

```
CREATE  OR  REPLACE  PACKAGE sample_pkg

  CREATE PROCEDURE private_proc() AS $$
   #PRIVATE
  BEGIN
    RAISE NOTICE 'private invoked';
  END;
$$;
```

• Процедуру можно вызывать только в пределах своего пакета!

```
CALL sample_pkg.private_proc();

ERROR:  private package function or procedure sample_pkg.private_proc() called out of its package
CONTEXT:  PL/pgSQL function private_proc()
```

Пример: приватность глобальных переменных пакета

PostgresPro

PG Pro Ent 16

```
CREATE  OR  REPLACE  PACKAGE sample_pkg
 CREATE FUNCTION __init__() RETURNS VOID AS $$
    #EXPORT OFF
    #EXPORT v_g_host_name;
 DECLARE
   v_g_host_name VARCHAR(50);
   v_g_user_name VARCHAR(50);
 BEGIN
  v_g_host_name := get_default_host_name();
 END;  $$;


DO $$
#import sample_pkg
BEGIN
  RAISE NOTICE 'Hostname => %', sample_pkg.v_g_user_name;
END; $$;
```

- Глобальная переменная доступна снаружи своего пакета – является публичной
- Глобальная переменная НЕ доступна снаружи своего пакета – является приватной!

```
ERROR:  missing FROM-clause entry for table “sample_pkg"
LINE 4: sample_pkg.v_g_user_name
```

PostgresPro

Поддержка коллекций в хранимых процедурах PL/pgSQL

Что такое коллекции в Oracle PL/SQL

PostgresPro

- Упорядоченная группа элементов одного типа
  - Одномерный “массив” в терминах других языков программирования
  - Может быть разреженной – некоторые элементы отсутствуют (для ассоциативных массивов)
  - Переменные коллекции имеют методы: FIRST, LAST, NEXT, PREV, DELETE, EXTEND, COUNT, …
- Nested Table (индексом может быть только число)

```
TYPE <type name> IS TABLE OF <element_type>;
```

- Varray (индексом может быть только число)

```
TYPE <type name> IS VARRAY(<limit>) OF <element_type>;
```

- Association Array (индексом может быть или число или строка)

```
TYPE <type name> IS TABLE OF <element_type> INDEX BY <index_type>;
```

Поддержка работы с коллекциями в pg_variables

PostgresPro

PG Pro Ent 16

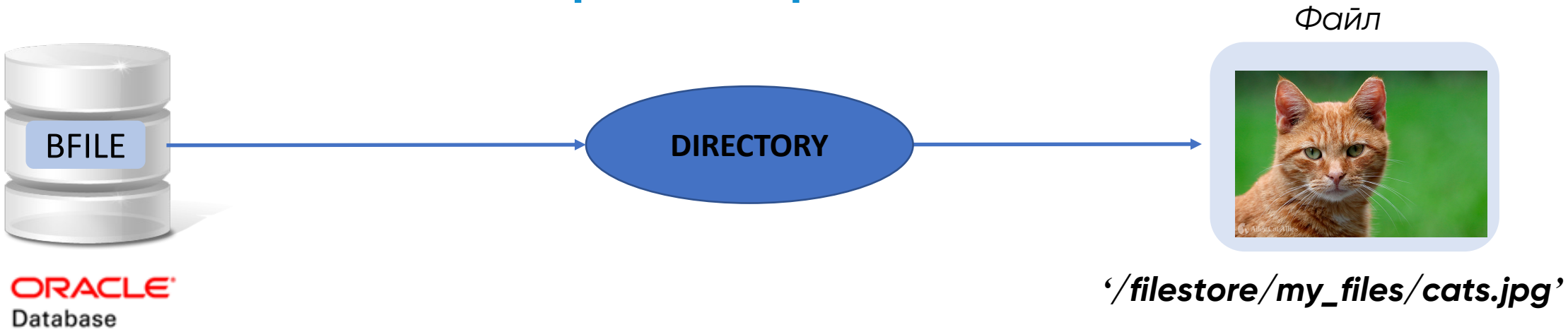
- Добавлены функции для работы с коллекциями в PL/pgSQL

Процедура/функция	Назначение
pgv_first_key	Возвращает ключ первой записи, ключ либо числовой, либо текстовый
pgv_last_key	Возвращает ключ последней записи
pgv_next_key	Возвращает ключ следующей записи от запрошенного ключа
pgv_prior_key	Возвращает ключ предыдущей записи от запрошенного ключа
pgv_set_elem	Присвоение значения элементу коллекции
pgv_remove_elem	Удаление элемента коллекции

```
DO $$
DECLARE
  l_index VARCHAR(10);
  r       "emp";
BEGIN
  l_index := pgv_first_key('scott_array', 'emp_by_lname');

  WHILE (l_index IS NOT NULL) LOOP
    r := pgv_get('scott_array', 'emp_by_lname', l_index::text);
    l_index := pgv_next_key( 'scott_array', 'emp_by_lname', l_index::text);
  END LOOP;
END; $$
```

Поддержка внешних файлов: Postgres Pro BFile

Oracle BFile – внешний файл на файловой системе ОС




```
SQL> CREATE OR REPLACE DIRECTORY my_dir AS '/filestore/my_files/';
SQL> GRANT READ ON DIRECTORY my_dir TO scott;

CREATE TABLE scott.demo_bfile (Id int, image BFILE);

INSERT INTO scott.demo_bfile VALUES(1, BFILENAME('MY_DIR','cats.jpg'));
COMMIT;
```

- Ограничения:
  - Доступен ТОЛЬКО на чтение
  - Размер НЕ более 4GB

Pgpro_bfile – расширение для работы с внешними файлами

PostgresPro

PG Pro Ent 16

- Предоставляет собственное API для работы с внешними файлами:

Процедура/функция	Назначение
bfile_directory_create	Создаёт каталог для заданных псевдонима и пути
bfile_directory_delete	Удаляет каталог с заданным псевдонимом
bfile_directory_rename	Изменяет псевдоним каталога
bfile_directory_set_path	Изменяет путь к каталогу с заданным псевдонимом
bfile_grant_directory	Предоставляет права доступа к каталогу для пользователя/роли
bfile_revoke_directory	Отзывает права доступа к каталог
bfile_open	Открывает файл
bfile_close	Закрывает файл
bfile_length	Возвращает длину открытого файла
bfile_read	Читает байты с смещения с начала файла
bfile_fileexists	Проверка существования файла (открывается без ошибок)
bfile_close_all	Закрывает все файлы, которые были открыты ранее с bfile_open
bfile_compare	Сравнение содержимого файлов
... ...	

Встроенный пакет DBMS_LOB – поддержка работы с BFile

PostgresPro

PG Pro Ent 16

```
DO $$
DECLARE
   cur_bfile bfile;
   buffer     bytea;
   amount   int := 3000;
BEGIN
  cur_bfile := dbms_lob.bfilename('BFILE_DATA', 'bfile.data');
  CALL dbms_lob.fileopen(cur_bfile, 0);

  CALL dbms_lob.read(cur_bfile, amount, 1, buffer);
  raise notice 'file contents: %, bytes read: %', buffer::text, amount;
  raise notice 'file substring from 2th position for 3bytes: %', dbms_lob.substr(cur_bfile, 2, 3);

  call dbms_lob.fileclose(cur_bfile);
END;
$$ ;
NOTICE:  file contents: \x30313233343536373839, bytes read: 10
NOTICE:  file substring from 4th position for 6 bytes: \x3233
DO
```

Postgres Pro Superfile

Postgres Pro Superfile

PostgresPro

PG Pro Ent 16

- Новая высокопроизводительная технология хранения и обработки данных в формате LOB (Large Object Binary):
  - Имеет свое собственное API доступа
  - Поддерживается параллелизм на операциях с SuperFile
  - Лишена недостатков технологии PostgreSQL Large Objects
  - Поддержка хранения LOB-сегментов на разных табличных пространствах
  - Поддержка сжатия с помощью Postgres Pro CFS

Сравнение Oracle Securefile, PostgreSQL Large Object и Postgres Pro Superfile

PostgresPro

PG Pro Ent 16

	Oracle Securefile	Large object	Superfile
Хранение	Отдельные табличные пространства	1 таблица	Отдельная схема
Управление системой хранения	автоматическое	нет	Автоматическое
Дескриптор объекта	10 байт	4 байта	8 байт
Максимальный размер	Не ограничен (?)	4 Тб	4 Тб (искусственно)
Количество объектов	Не ограничено (?)	2^32	2^64
Общий размер объектов	Не ограничен (?)	32 Тб	Не ограничен
Параллельные операции	Да	Нет	Да
Интерфейс	Частично SQL, API доступа	API доступа	API доступа

pg_superfile – расширение для работы с LOB

PostgresPro

PG Pro Ent 16

- Предоставляет собственное API для работы с LOB:

Процедура/функция	Назначение
sf_create	Создание Superfile LOB
sf_write	Запись в конец LOB (операция APPEND)
sf_read	Чтение содержимого LOB
sf_size	Возвращает содержимое LOB
sf_truncate	Удаляет содержимое LOB – он становится пустым (EMPTY LOB)
sf_delete	Удаление LOB-объекта
sf_describe	Возвращает описание (метаданные) LOB-объекта
sf_get_type	Возвращает тип LOB (BLOB или CLOB)
sf_get_json_options	Возвращает метаданные ассоциированные с LOB-объектом в виде JSON
sf_find	Поиск вхождения данных в LOB-объекте
sf_set_option	Установка опций LOB
... ...	

Встроенный пакет DBMS_LOB – поддержка работы с Postgres Pro Superfile

PostgresPro

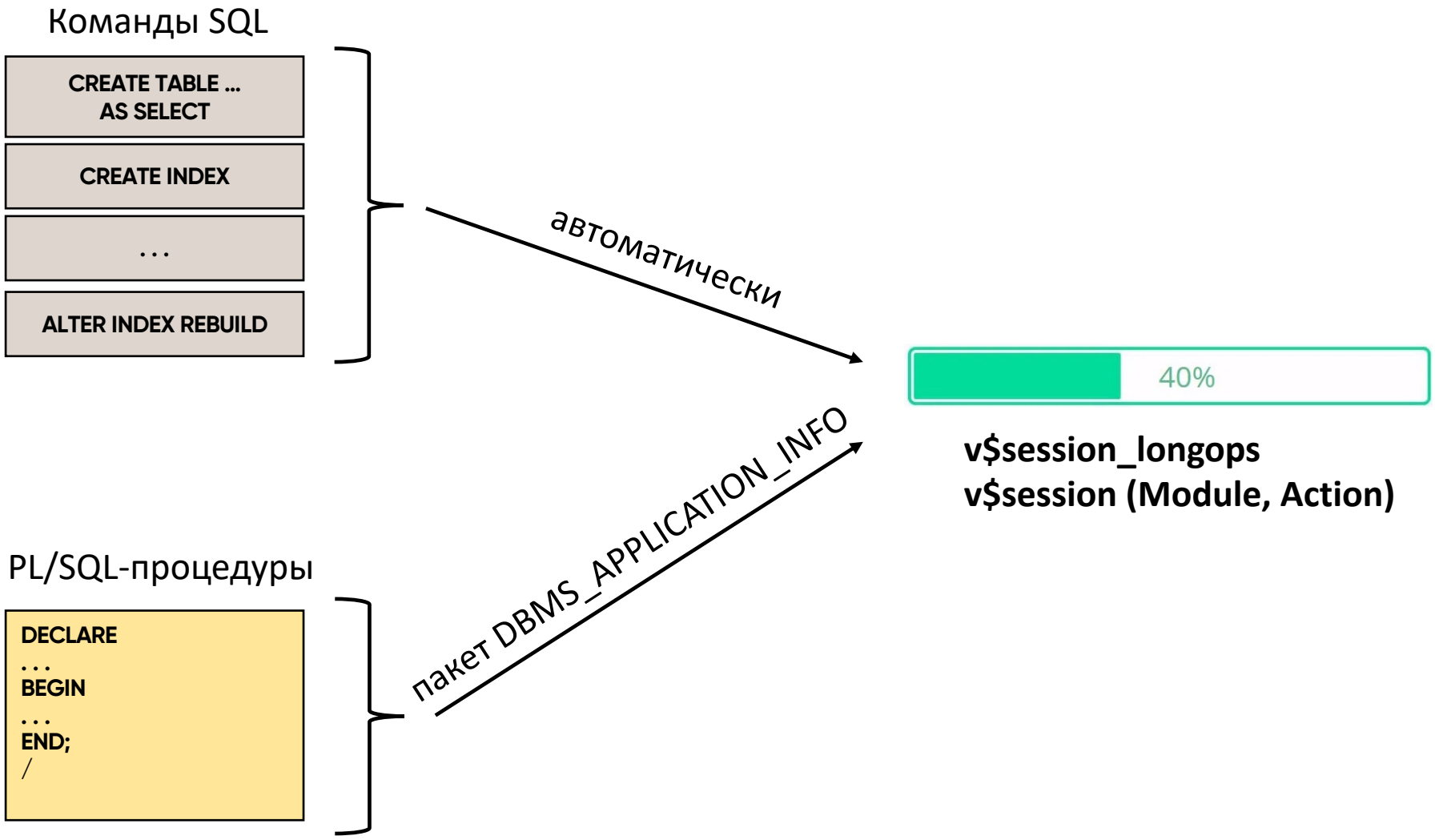
PG Pro Ent 16

```
DO $$
DECLARE
        cur_clob        dbms_lob.clob;
        buffer          text;
        amount          int := 3000;
BEGIN
        cur_clob := dbms_lob.empty_clob();
        cur_clob.t := 'just some sample text';
        raise notice 'clob length: %', dbms_lob.getlength(cur_clob);
        call dbms_lob.read(cur_clob, amount, 1, buffer);
        raise notice 'all clob read: %', buffer;

        call dbms_lob.write(cur_clob, 6, 4, 'foobar');
        call dbms_lob.writeappend(cur_clob, 4, 'test');

        call dbms_lob.erase(cur_clob, 3, 2);
        call dbms_lob.trim_(cur_clob, 22);
        raise notice 'new clob contents: %', cur_clob.t;
END; $$;
```

PostgresPro

Новые встроенные пакеты DBMS_APPLICATION_INFO, UTL_SMTP, UTL_MAIL, UTL_HTTP и DBMS_LOB

Прогресс выполнения операции в Oracle Database



Инструментирование кода в СУБД Oracle с помощью пакета DBMS_APPLICATION_INFO

PostgresPro

```
BEGIN
  dbms_application_info.set_module('Зарплата и кадры');
  dbms_application_info.set_action('Расчет премии');

  FOR i IN 1.. total_bonus_count LOOP
    dbms_application_info.set_session_longops(RINDEX      => rindex,
                                              SLNO        => slno,
                                              OP_NAME     => 'Считаем премию',
                                              CONTEXT     => 42,
                                              SOFAR       => i,
                                              TOTALWORK   => total_bonus_count,
                                              UNITS       =>'рубль');

    calc_bonus();
  END LOOP;
  dbms_application_info.set_action(NULL);
END;
```

Отображение прогресса выполнения в системах мониторинга Oracle

PostgresPro

- Мониторинг активности по Module и Action (из v$session):

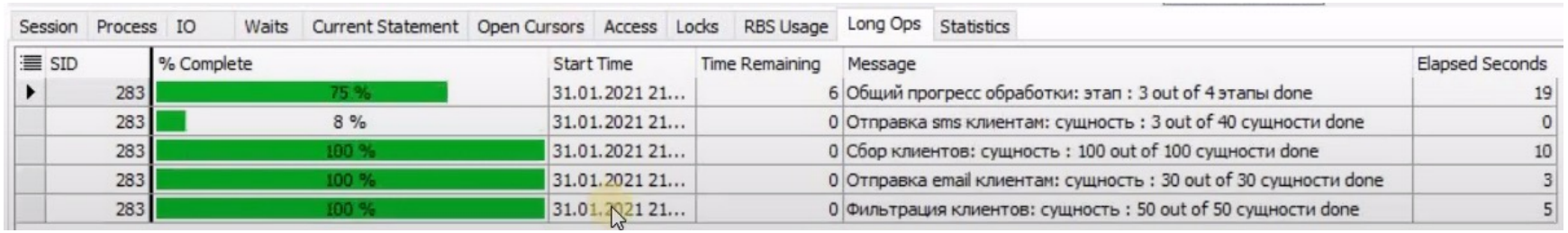
Top Modules

View Top Modules

Activity (%)	Service	Module
38.85	DB11G.WORLD	New Order
21.89	DB11G.WORLD	Browse Products
14.59	DB11G.WORLD	
7.99	DB11G.WORLD	Process Orders
5.49	SYS$BACKGROUND	
3.96	DB11G.WORLD	New Customer
2.43	DB11G.WORLD	Browse and Update Orders
1.11	SYS$USERS	
.97	SYS$USERS	Realtime Connection
.63	SYS$USERS	emagent_SQL_oracle_database

Total Sample Count: 1,439

Мониторинг прогресса выполнения (из представления v$session_longops):



Session | Process | IO | Waits | Current Statement | Open Cursors | Access | Locks | RBS Usage | Long Ops | Statistics

SID	% Complete	Start Time	Time Remaining	Message	Elapsed Seconds
283	75 %	31.01.2021 21...	6	Общий прогресс обработки: этап : 3 out of 4 этапы done	19
283	8 %	31.01.2021 21...	0	Отправка sms клиентам: сущность : 3 out of 40 сущности done	0
283	100 %	31.01.2021 21...	0	Сбор клиентов: сущность : 100 out of 100 сущности done	10
283	100 %	31.01.2021 21...	0	Отправка email клиентам: сущность : 30 out of 30 сущности done	3
283	100 %	31.01.2021 21...	0	Фильтрация клиентов: сущность : 50 out of 50 сущности done	5

PG Pro Application Info

PG Pro Ent 16

- Расширение (extension) для Postgres Pro Enterprise 16
- Бэкпортировано в Postgres Pro Enterprise 15.4+
- Метрики и атрибуты сессии как и в Oracle, хранятся в Shared Memory
- Может работать на read-only реплике БД (не хранит информацию на диске!)
- Для просмотра информации о сессиях: представления v$session и v$session_longops в PG Pro
- Пакет DBMS_APPLICATION_INFO: API полностью повторяет аналогичный пакет из Oracle

PG Pro Application Info - пример

PostgresPro

PG Pro Ent 16

```
BEGIN
  CALL dbms_application_info.set_module('Зарплата и кадры');
  CALL dbms_application_info.set_action('Расчет премии');

  FOR i IN 1..total_bonus_count LOOP
    CALL dbms_application_info.set_session_longops(RINDEX      => rindex,
                                                   SLNO        => slno,
                                                   OP_NAME     => 'Считаем премию',
                                                   CONTEXT     => 42,
                                                   SOFAR       => i,
                                                   TOTALWORK   => total_bonus_count,
                                                   UNITS       =>'рубль');

    CALL calc_bonus();
  END LOOP;

  CALL dbms_application_info.set_action(NULL);
END;
```

PostgresPro

Встроенные пакеты UTL_SMTP, UTL_MAIL и UTL_HTTP

PostgresPro

PG Pro Ent 16

- Реализация пакетов UTL_SMTP, UTL_MAIL и UTL_HTTP в СУБД Postgres Pro
- UTL_SMTP, UTL_MAIL – для отправки сообщений по email из хранимых процедур PL/pgSQL
- UTL_HTTP для интеграции с внешними источниками по протоколу HTTP из хранимых процедур PL/pgSQL
- API почти полностью соответствует пакетам из СУБД Oracle Database!

Использование пакета UTL_SMTP в Postgres Pro – Пример

PostgresPro

PG Pro Ent 16

```
DO $$
DECLARE
   conn utl_smtp.connection;
BEGIN
   conn := utl_smtp.open_connection('smtp.mail.ru', 25, 10);
   perform utl_smtp.ehlo(conn, 'localhost');    perform utl_smtp.starttls(conn);
   perform utl_smtp.ehlo(conn, 'localhost');
   perform utl_smtp.auth(conn, 'test_email@example.com', 'super-secret-password');
   perform utl_smtp.mail(conn, 'sender@example.com');   perform utl_smtp.rcpt(conn, 'recipient@example.com');
   perform utl_smtp.open_data(conn);
   perform utl_smtp.write_data(conn, E'Content-Type: multipart/mixed; boundary=------------------------6f48b7d5ded0c5fc\n');
   perform utl_smtp.write_data(conn, E'Mime-Version: 1.0\n');
   perform utl_smtp.write_data(conn, E'From: Sender <sender@example.com>\n');
   perform utl_smtp.write_data(conn, E'To: Recipient <recipient@example.com>\n');
   perform utl_smtp.write_data(conn, E'Subject: mail from utl_smtp\n');
   perform utl_smtp.write_data(conn, E'--------------------------6f48b7d5ded0c5fc\n');
   perform utl_smtp.write_data(conn, E'Content-Type: text/plain; charset=\"UTF-8\"\n');
   perform utl_smtp.write_data(conn, E'Content-Transfer-Encoding: 8bit\n\n');
   perform utl_smtp.write_data(conn, E'This is body from inside Postgres Pro\n');
  perform utl_smtp.write_data(conn, E'\n--------------------------6f48b7d5ded0c5fc--\n');
   perform utl_smtp.close_data(conn);   perform utl_smtp.quit(conn);
END$$;
```

Postgres Pro UTL_SMTP – особенности при обработке исключений

Oracle PL/SQL

```
EXCEPTION
  WHEN utl_smtp.transient_error THEN
     ...
  WHEN utl_smtp.permanent_error THEN
        ...
  WHEN others
        ...
END;
```

Postgres Pro

```
EXCEPTION
  WHEN raise_exception THEN
    CASE utl_smtp.last_error()
      WHEN utl_smtp.transient_error() THEN
        ...
      WHEN utl_smtp.permanent_error() THEN
        ...
      ELSE
      ...
    END CASE;
END;
```

PG Pro UTL_HTTP – Пример

```
DO
$$
DECLARE
        request         utl_http.req;
        response        resp;
        raw_body        bytea;
        text_body       text;
        header_value text;
        encoding        text;
BEGIN
        request    := utl_http.begin_request('http://lib.ru/RUFANT/CIOLKOWSKIJ/luna.txt');
        response := utl_http.get_response(request);

        CALL utl_http.get_header_by_name(response, 'Content-Type', header_value);
        encoding = substring(header_value from '.*charset=(.*)$');

        CALL utl_http.read_raw(response, raw_body);
        text_body := convert_from(convert(raw_body, encoding, 'UTF-8'), 'UTF-8');
        text_body = substring(text_body from 420 for 600);
        RAISE NOTICE '%', text_body;
END; $$
```

PG Pro UTL_HTTP – Пример (продолжение)

PostgresPro

```
psql:c:/temp/lib.sql:22: NOTICE:  <ul><a name=0></a><h2>Константин Циолковский. На Луне</h2></ul>

  ------------------------------------------------------------------
  В сб. "У светлого яра Вселенной".
  М., "Правда", 1989 (серия "Мир приключений").
  OCR & spellcheck by HarryFan, 25 May 2001
  ------------------------------------------------------------------



<ul><a name=1></a><h2>1</h2></ul>

  Я проснулся и, лежа еще в постели, раздумывал  о  только  что  виденном мною сне: я видел себя купающимся, а так как была зима,  то  мне  особенно казалось приятно помечтать о летнем купанье.
  Пора вставать!

DO
```

Postgres Pro DBMS_LOB

- Аналог системного пакета DBMS_LOB из СУБД Oracle
- API практически полностью идентично Oracle
- API почти полностью соответствует пакетам из СУБД Oracle Database!
- Поддержка временных LOB-объектов (Temporary LOB)
- Упрощение миграции с Oracle

```
DO $$
DECLARE
        cur_blob dbms_lob.blob;
BEGIN
        call dbms_lob.createtemporary(cur_blob, true);
        raise notice '%', cur_blob;
        call dbms_lob.writeappend(cur_blob, 4, 'text');
        raise notice '%', cur_blob;
        call dbms_lob.freetemporary(cur_blob);
        raise notice '%', cur_blob;
END;
$$;
```

PostgresPro

Конвертация PL/SQL кода – утилита ora2pgpro 2.0

Утилита ORA2PGPRO (в составе PG Pro Enterprise):

- Форк от ORA2PG, с дополнительными новыми возможностями
- Транслятор кода пакетов Oracle в синтаксис Postgres Pro 15 Ent
- Генерация скриптов выдачи привилегий на пакет (grant)
- Конвертация синтаксиса автономных транзакций Oracle PL/SQL в синтаксис Postgres Pro
- Исправление ошибок ORA2PG
- И т.д.

Пример пакета в СУБД Oracle Database

```
CREATE OR REPLACE PACKAGE pkgc IS
TYPE r_customer_type IS RECORD(
   customer_name varchar2(50),
   credit_limit number(10,2)
);

TYPE t_customer_type IS VARRAY(2)
   OF r_customer_type;

PROCEDURE VARRAY_TEST AS
   t_customers t_customer_type := t_customer_type();
   rec r_customer_type;
   tmp_string varchar2(2000);
BEGIN
   t_customers.EXTEND;
   t_customers(t_customers.LAST).customer_name := 'ABC Corp';
   t_customers(t_customers.LAST).credit_limit  := 10000;

   FOR indx in 1 .. t_customers.COUNT LOOP
      rec := t_customers(indx);
      tmp_string := 'RECORD: ' || rec.customer_name || ', ' || rec.credit_limit;
      insert into log_table (id, line) values (1+ indx, tmp_string);
   END LOOP;
END;
BEGIN   delete from log_table; END pkgc;
```

ora2pg: пример преобразование с Oracle PL/SQL на PL/pgSQL (Преобразованный ora2pg код для PL/pgSQL)

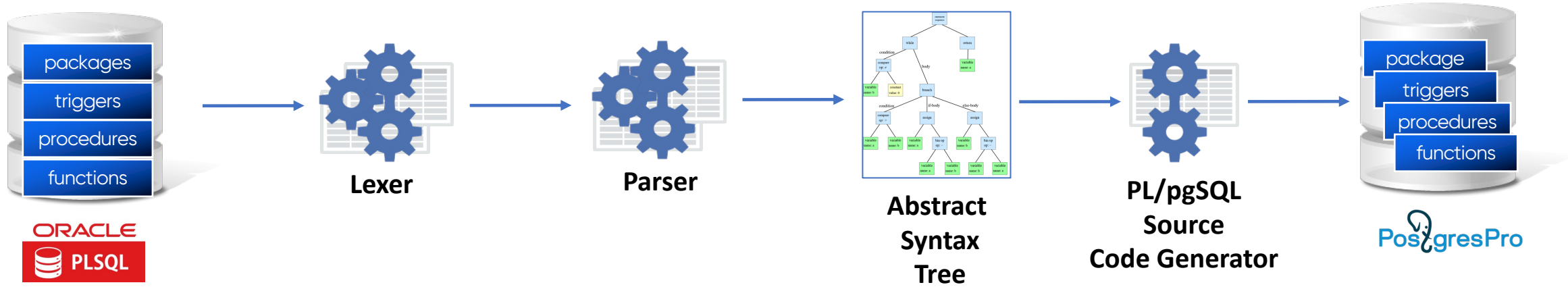
```
CREATE TYPE pkgc.r_customer_type AS (customer_name varchar(50),
                    credit_limit double precision);

CREATE TYPE pkgc.t_customer_type AS (t_customer_type pkgc.r_customer_type[2]);

CREATE OR REPLACE PROCEDURE pkgc.varray_test () AS $body$
DECLARE
   t_customers t_customer_type := t_customer_type();
   rec r_customer_type;
   tmp_string varchar(2000);
BEGIN
   t_customers.EXTEND;
   t_customers[t_customers.LAST].customer_name := 'ABC Corp';
   t_customers[t_customers.LAST].credit_limit  := 10000;

   FOR indx in 1 .. t_customers.COUNT LOOP
      rec := t_customers(indx);
      tmp_string := 'RECORD: ' || rec.customer_name || ', ' || rec.credit_limit;
      insert into pkgc_log(id, line) values (1+ indx, tmp_string);
   END LOOP;
END;

begin
  delete from pkgc_log;
;
$body$
```

ora2pg:

Код нужно вручную переписывать!

ora2pgpro 2.0: Транспилятор с Oracle PL/SQL на PL/pgSQL

- В отличие от ora2pg, имеет полноценный парсер с Oracle PL/SQL с построением AST (Abstract Syntax Tree) – дерева разбора исх. кода
- С дальнейшим проходом AST PL/SQL и генерацией эквивалентного кода на PL/pgSQL
- Возможны сколько угодно сложные преобразования исходного кода пакетов из Oracle в Postgres Pro



ora2pgpro: пример преобразование с Oracle PL/SQL на PL/pgSQL (Преобразованный ora2pgpro код для PL/pgSQL) – ч.1

```
CREATE SCHEMA PKGC ;

CREATE OR REPLACE FUNCTION PKGC.__INIT__() RETURNS VOID AS $$
#package

begin
  delete from pkgc_log;
END;
$$ LANGUAGE plpgsql;

CREATE TYPE PKGC.r_customer_type AS (
    customer_name varchar(50),
    credit_limit numeric(10,2)
);

CREATE DOMAIN PKGC.t_customer_type PKGC.r_customer_type[];
```

ora2pgpro: пример преобразование с Oracle PL/SQL на PL/pgSQL (Преобразованный ora2pgpro код для PL/pgSQL) – ч.2

```
CREATE OR REPLACE PROCEDURE PKGC.VARRAY_TEST () AS $$
#package
   DECLARE

   /*TODO: collection constructors are not supported.*/
   t_customers PKGC.t_customer_type /*:=*/ /*t_customer_type() /*varray_constructor_call*/*/;
   rec PKGC.r_customer_type;
   tmp_string varchar(2000);
BEGIN
   t_customers = array_cat(t_customers, array_fill(NULL::PKGC.R_CUSTOMER_TYPE, ARRAY[1]));
   t_customers[array_upper(T_CUSTOMERS, 1)].customer_name = 'ABC Corp';
   t_customers[array_upper(T_CUSTOMERS, 1)].credit_limit  = 10000;

   FOR indx in 1 .. array_length(T_CUSTOMERS, 1) LOOP
      rec = t_customers[indx];
      tmp_string = 'RECORD: ' || rec.customer_name || ', ' || rec.credit_limit;
      insert into pkgc_log (id, line) values (1+ indx, tmp_string);
   END LOOP;
END; $$ LANGUAGE PLPGSQL;
```

- Корректный исходный код PL/pgSQL – полностью работоспособен!

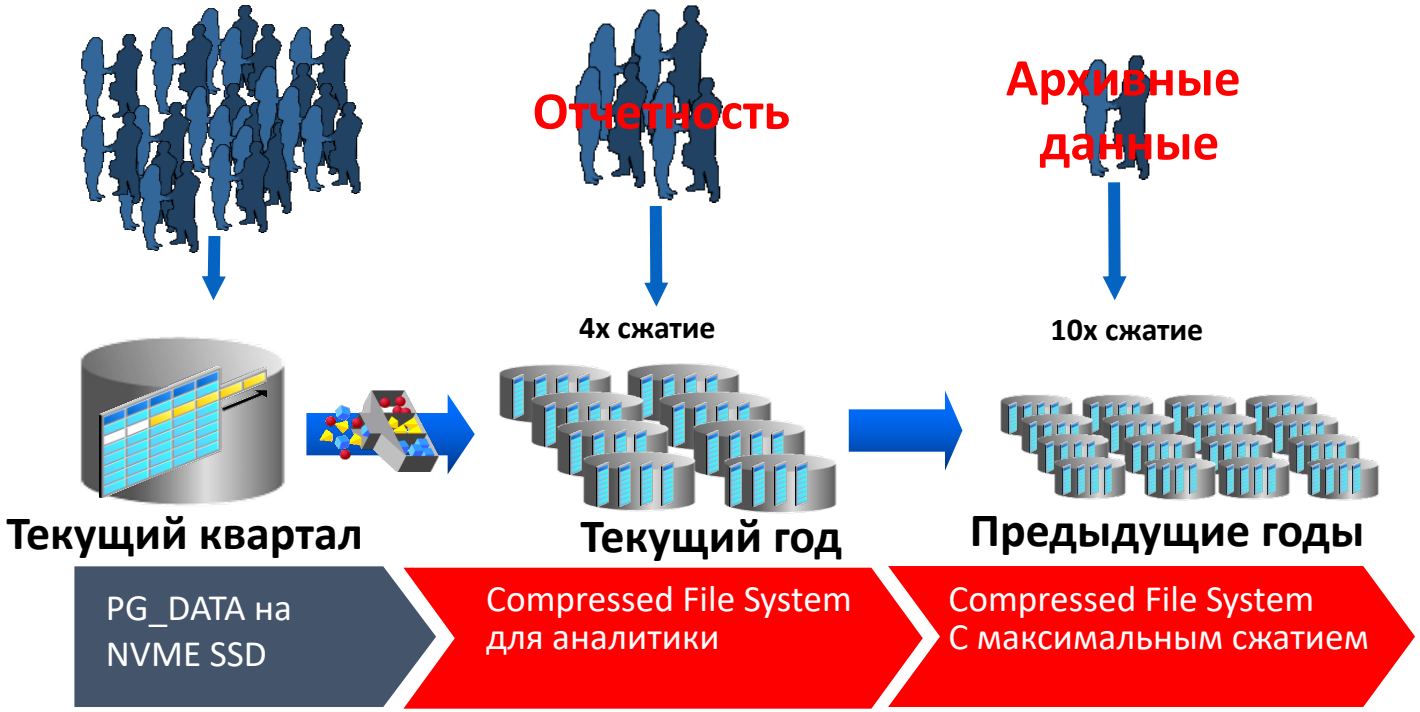
PostgresPro

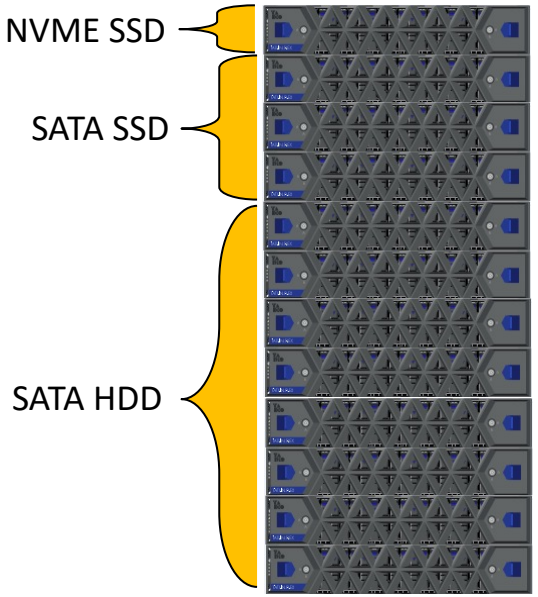
Планы по развитию

Планы по развитию

- могут измениться

- Postgres Pro Advanced Queuing – хранение и обработка очередей в СУБД
- Postgres Pro ILM – управление жизненным циклом информации
- Добавление новых системных пакетов





В соответствии с политиками, Postgres Pro ILM автоматически перемещает данные и может их сжимать “на лету”

PostgresPro

Заключение

Postgres Pro Ent. 16 - упрощение миграции с Oracle

Продукт	Аналог Oracle	Функциональность	Преимущества
Postgres Pro SuperFile	Oracle Securefile	Высокопроизводительная технология для данных типа LOB	LOB Compression, снятие ограничений PG и новая функциональность
Postgres Pro BFile	Oracle BFile	LOB в виде ссылок на внешние файлы на ФС	Упрощение миграции и новая функциональность
Postgres Pro DBMS_LOB	Пакет DBMS_LOB	Аналог пакета DBMS_LOB (поверх SuperFile и pgpro_BFile)	Упрощение миграции
ora2pgpro 2.0	-	Конвертация кода работы с коллекциями PL/SQL	Упрощение миграции
UTL_SMTP, UTL_MAIL, UTL_HTTP, DBMS_APPLICATION_INFO	UTL_SMTP, UTL_MAIL, TL_HTTP, BMS_APPLICATION_INFO	Системные пакеты	Упрощение миграции и новая функциональность
pg_variables	Переменные коллекций	Поддержка коллекций	Упрощение миграции и новая функциональность

Заключение

- Ora2pgpro 2.0
- PG Pro Application Info
- PG Pro Utils: UTL_SMTP, UTL_MAIL, UTL_HTTP
- PG Pro SuperFile
- PG Pro BFILE
- PG Pro DBMS_LOB
- Поддержка коллекций
- Приватность элементов пакета

- Значительное уменьшение трудозатрат на миграцию с Oracle: UTL_SMTP, UTL_MAIL, UTL_HTTP, DBMS_LOB, DBMS_APPLICATION_INFO
- Новый востребованный функционал: PG Pro BFile, PG Pro SuperFile

- API системных пакетов аналогично пакетам из СУБД Oracle Database!

Компания Postgres Pro

Протестировать
СУБД Postgres Pro:

117036, Москва, ул. Дмитрия Ульянова, 7А

8 (495) 150-06-91

sales@postgrespro.ru

postgrespro.ru

PostgresPro

Спасибо за внимание

PostgresPro

Q&A