

ORACLE®



ORACLE®

Оптимизатор PL/SQL

Игорь Мельников
старший консультант IMC Moscow

План

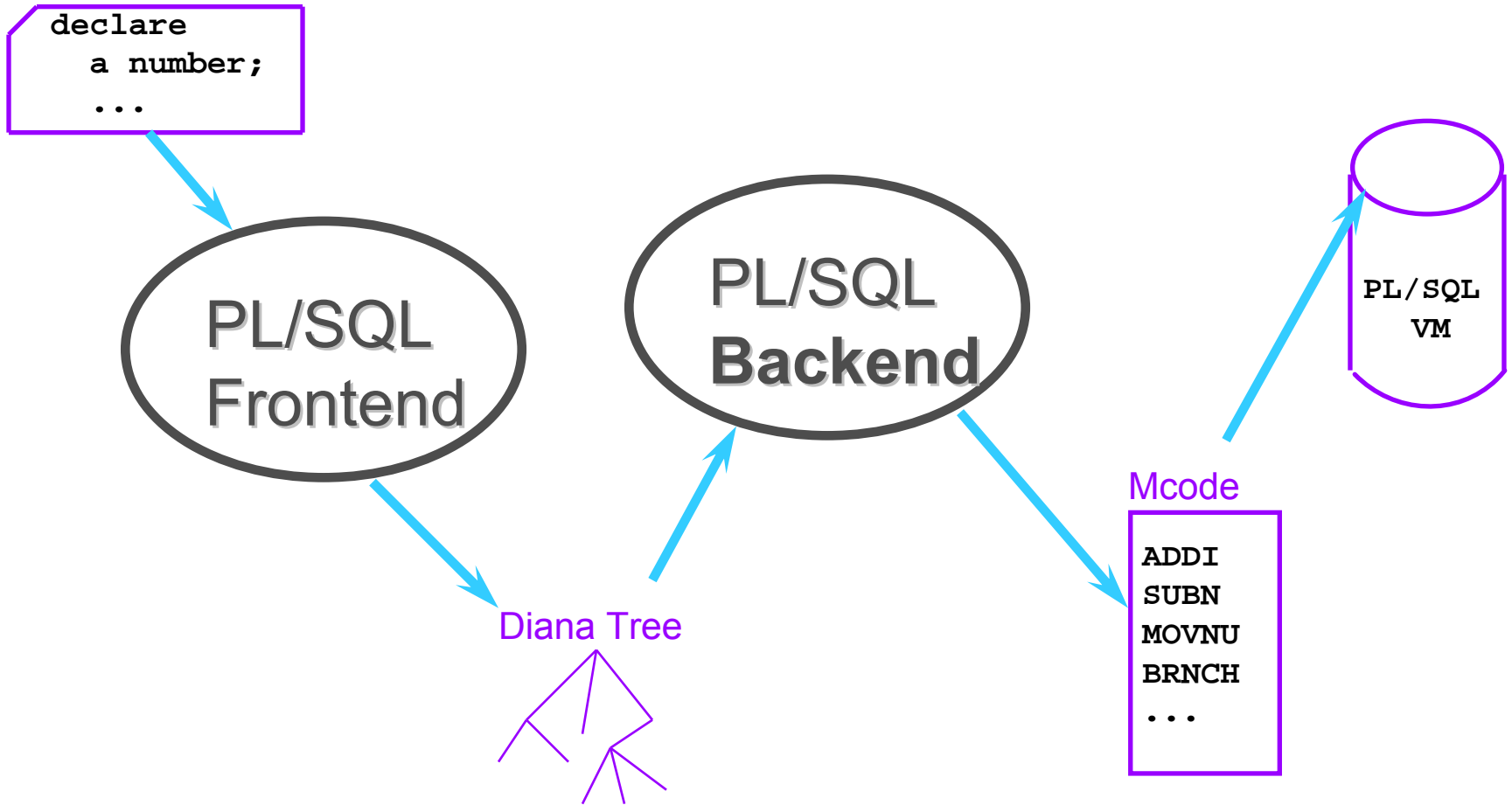
- Введение в компиляцию PL/SQL
- Оптимизатор PL/SQL
- Использование оптимизатора
- Новые возможности в Oracle Database 11g
- Демонстрация



Компиляция PL/SQL



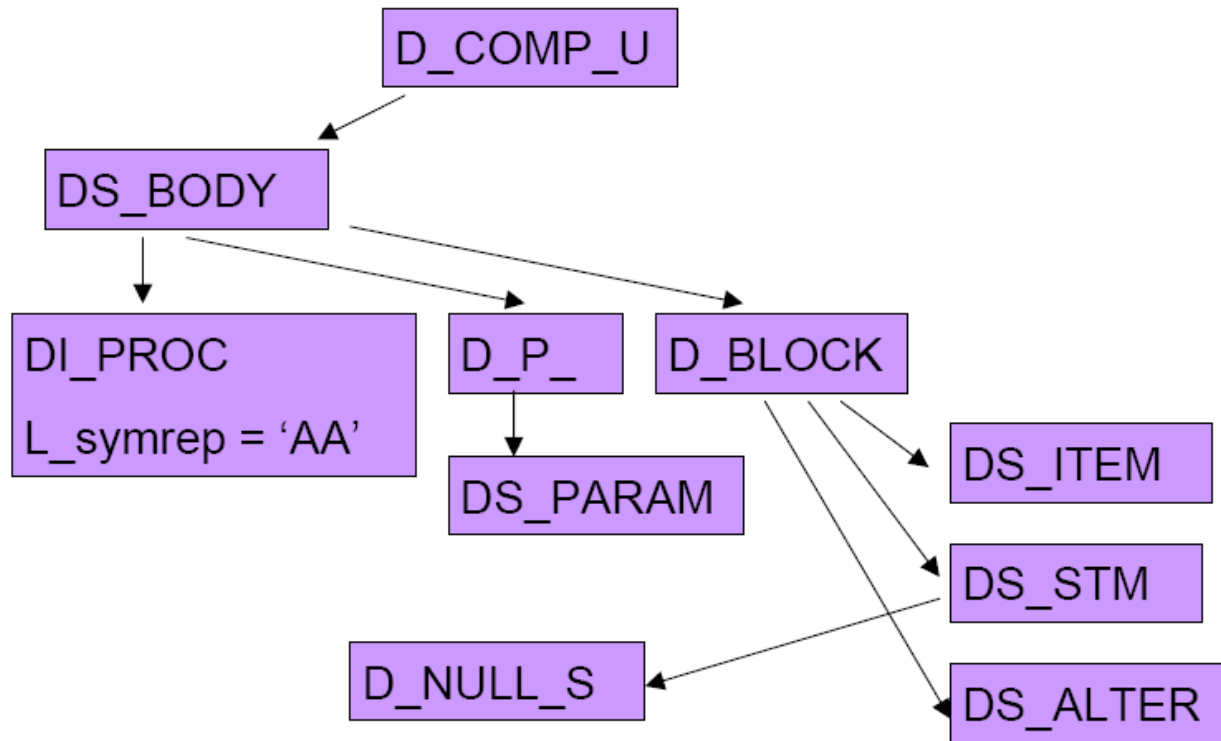
Компиляция и выполнение PL/SQL



Что такое DIANA

Descriptive Intermediate Attributed Notation for ADA

```
create or replace procedure AA is
begin
  null;
end;
```



Что такое PL/SQL VM

- Виртуальная машина PL/SQL
- Выполняет MCode (команды VM)
- Регистро-ориентированная
 - Напр: $x := y * z$;

```
ld  y, R1
ld  z, R2
mul R1, R2, R1
st  R1, x
```
- *Стек-based* (как *Java VM*)

```
push y  # push y on stack
push z  # push z on stack
imul   # multiply
pop  x  # pops result into x
```
- Но также поддерживает операции “память-память”

```
muli  addr_of_y, addr_of_x, addr_of_z
```

Инструкции MCode

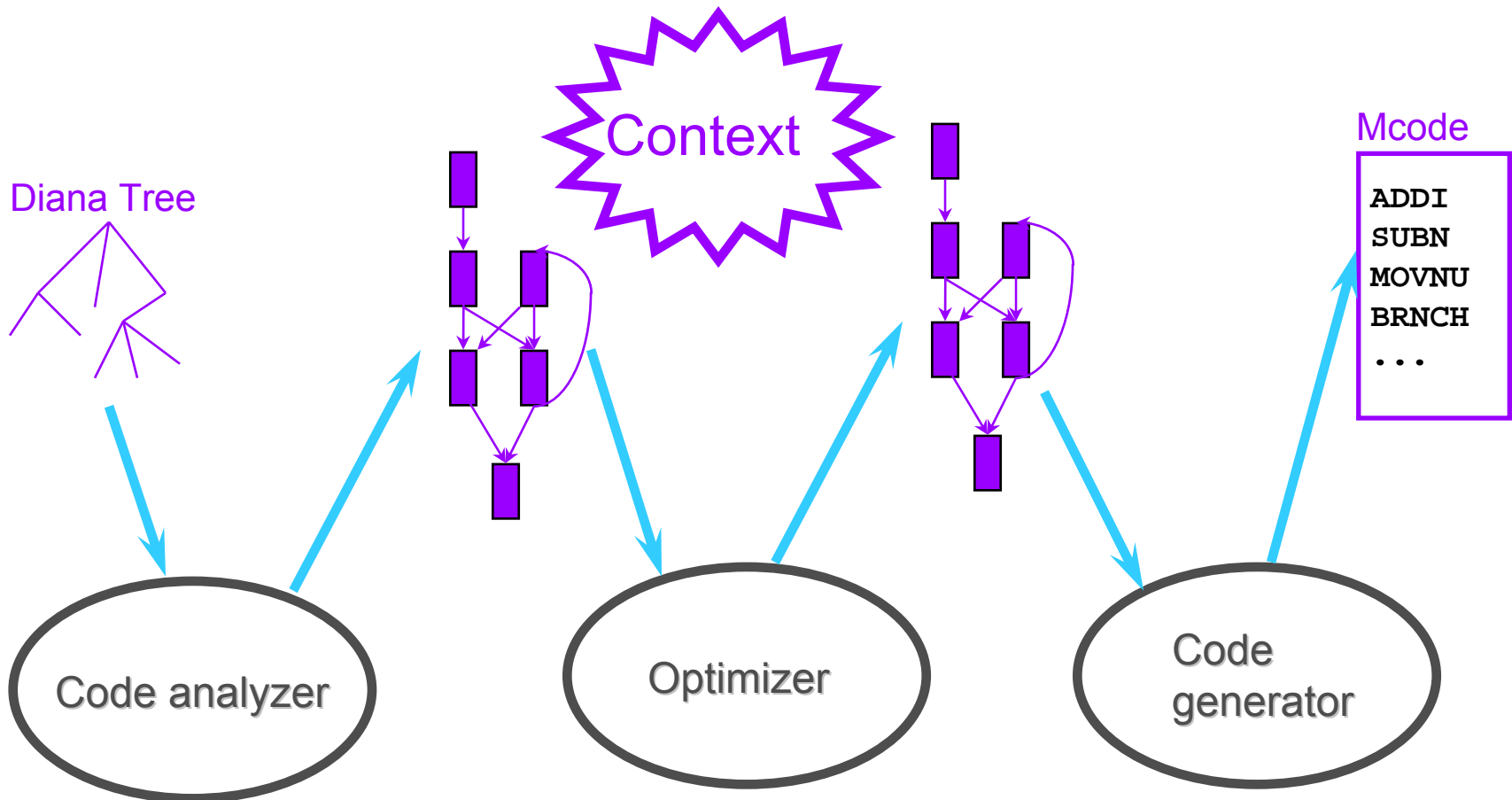
- Зависят от платформы
 - Велина смещения при переходе, размер фрейма, и т.д..
- Одно-байтовые команды
- В основном команды имеют фиксированное число операндов
- Но некоторые команды имеют переменное число операндов

Оптимизатор PL/SQL



Оптимизатор PL/SQL

Новый backend в 10g



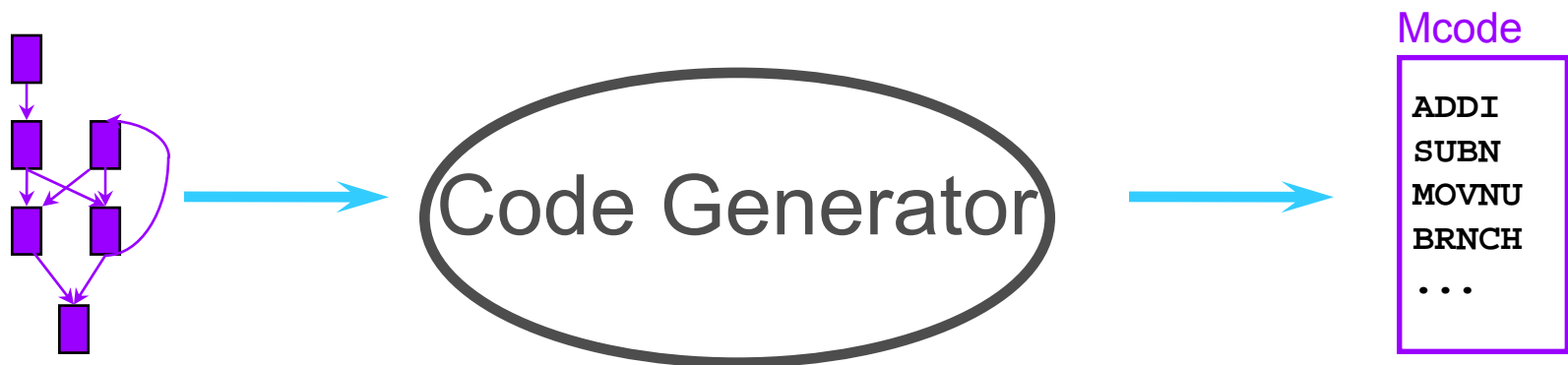
Действия оптимизатора

- Merge chained **Blocks**
- Replace branches with terminals
- Kill dead **Blocks**
- Evaluate (simplify) constant operations
- Propagate constants
- Propagate variables
- Eliminate dead **Instructions**
- Lift relations into branches
- Remove assignments to temporaries
- Lift addressing
- Eliminate temporaries

Новый генератор MCode

Новый генератор кода PL/SQL VM в 10g

- Добавлены новые инструкции VM (для более полного использования команд CPU) и поддержки оптимизации
- Удалены инструкции которые теперь не нужны
- Анализ дополнительной информации от оптимизатора (например: привязка к номерам строк исх. текста)
- Внутренний редизайн PL/SQL VM (напр: выравнивание фреймов на границу слова)



Использование оптимизатора PL/SQL



Оптимизация PL/SQL

Установка уровня оптимизации

- Всего существуют три уровня оптимизации
- Устанавливается с помощью параметра *PLSQL_OPTIMIZE_LEVEL*:
- Устанавливается как на уровне экземпляра, так и на уровне сессии;
- Может быть задан для конкретного объекта:

```
SQL> alter package account_pkg compile plsql_optimize_level=2;
```

```
Package altered.
```

```
SQL> |
```

Оптимизация PL/SQL

Уровни оптимизации

- 0 - оптимизации отключена (как в пред. версиях);
 - 1 – обычная (common) оптимизация;
 - 2 – агрессивная оптимизация.
-
- По умолчанию установлена агрессивная оптимизация: *PLSQL_OPTIMIZE_LEVEL* = 2

Уровень оптимизации N 1

Локальная оптимизация

- Оптимизация в пределах одного программного блока
- Рекомендуется использовать этот уровень, только если очень критично время компиляции
- Методы оптимизации:
 - Удаление неиспользуемых переменных;
 - Удаление “мертвого” кода
 - Выделение констант;
 - Оптимизация конкатенации нескольких строк в одном выражении;
 - Регистровый инкремент/декремент
 -

Оптимизация первого уровня


Пример (выделение инвариантов цикла)

- До оптимизации:

```
begin
  for fAcc in (select * from accounts)
  loop
    xCnt := xAmount / 2;
    ... ..
  end loop;
end;
```

- После оптимизации:

```
begin
  xCnt := xAmount / 2;
  for fAcc in (select * from accounts)
  loop
    ... ..
  end loop;
end;
```



Оптимизация первого уровня

Пример (выделение повторяющихся выражений)

- До оптимизации:

```
begin
  xA := xB + xC + 12;
  ... ..
  xD := xB + xC - 24;
end;
```

- После оптимизации:

```
xT pls_integer;
begin
  xT := xB + xC;
  xA := xT + 12;
  ... ..
  xD := xT - 24;
end;
```

Оптимизация первого уровня

Пример (удаление “мертвого” кода)

- До оптимизации:

```
declare
  K pls_integer;
  S pls_integer;
begin
  S := 12;
  return;
  K := S/2;
end;
```

- После оптимизации:

```
declare
  S pls_integer;
begin
  S := 12;
  return;
end;
```

Уровень оптимизации N 2

Агрессивная оптимизация

- Включает в себя методы из уровня N1
- Более продвинутые технологии оптимизации:
 - Уменьшение переключений контекста;
 - Учет особенностей ядра Oracle и машины PL/SQL;
 - ...
- К сожалению, методы нигде не документированы !

Оптимизация PL/SQL (Уровень N2)

Пример (уменьшение переключений контекста)

- Был описан *Stewen Feuerstein* в 5-ом издании своей книги “Oracle PL/SQL Programming”
- Исходный код ДО оптимизации:

```
begin
  for fEmp in (select * from emp)
  loop
    ... ..
  end loop;
end;
```

Оптимизация PL/SQL (Уровень N2)

Пример (уменьшение переключений контекста)

- Код ПОСЛЕ оптимизации:

```
declare
  type t_Table is table of emp%rowtype index by pls_integer;

  v_xTable t_Table;
  v_xCount pls_integer;
begin
  select * bulk collect into v_xTable from emp;
  v_xCount := v_xTable.Count;

  for v_xIndex in 1..v_xCount loop
    ... ..
  end loop;
end;
```

- Подобная “мутация” кода происходит и для явных курсоров!

Как увидеть код после оптимизации ?

- Специальный *event*
event="10944 trace name context forever, level <whatever>"
- Нужен рестарт экземпляра
 - **none, 0, or 1** = use old compiler.
 - **6** = run new backend with full dump.
 - **10** = dump Diana.
 - **11** = use new backend.
 - **13** = new backend with no optimizer.
 - **15** = new backend with no optimizer.
 - **18** = new backend with more informative dump.
 - **25** = global optimizer with dump.
 - **26** = global optimizer without dump.
 - **27** = dump MCode only, new backend.

Для анализа нужно иметь парсер 😊

```
*** [ 0 ] : Code size = 273 IL lines.
=== Graph and code before optimization. ===
*** Graph dump for unit Unit_Xanon_4A5FAFB0__AB...(0DCE6E74). ***

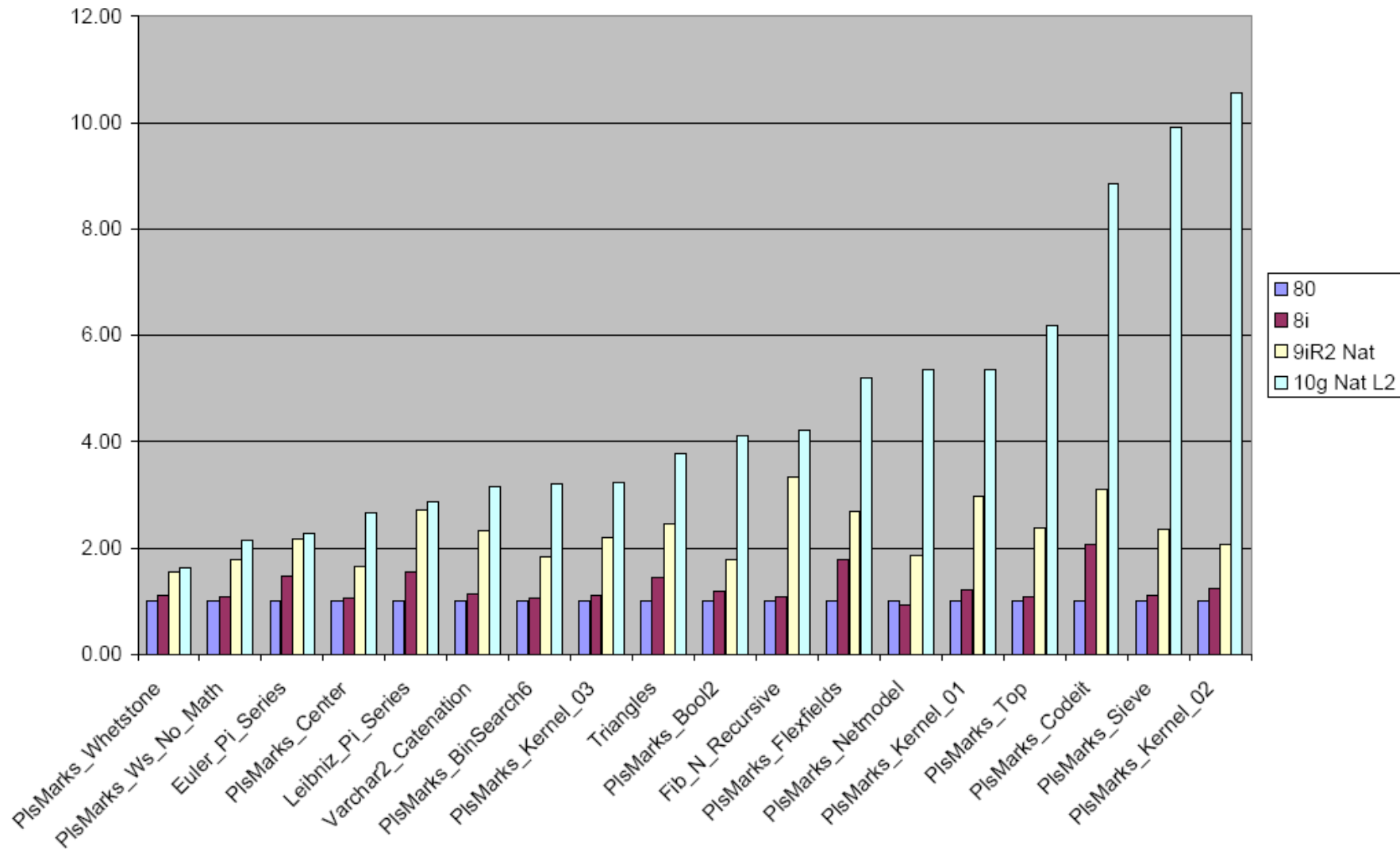
-- Procedure __anonymous_block(0E6EF128) flow graph dump. --
-- Dump of Block Set 0E6EE11C. --

-- Fields of a flow graph Block --
Self          = 075FCFC4  Name          = *Anon Block*  ID              = <1, *Anon Block*>
Label         = 0E6EF068  Label text     = $L1           Audit serial    = 1
Labels        = 0E6EE018  Frame         = 0E6EF128      Frame text     = __anonymous_block
Exception range = 00000000  CG Info       = 00000000      Line          = -2
Optim flags   = 0          Is_Entry      = FALSE         Is_Exit        = FALSE
Is_Enter      = TRUE       Is_Handler    = FALSE

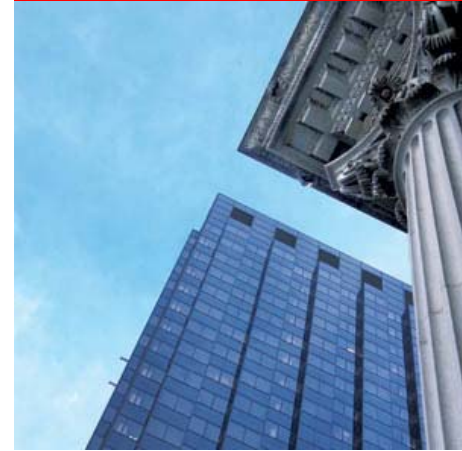
-- End Fields of a flow graph Block --
In:  <3, Entry __anonymous_block> --> <1, *Anon Block*>
Out: <1, *Anon Block*> --> <5, *Anon Body*>

-- Fields of a flow graph Block --
Self          = 075FD058  Name          = *Anon Tail*  ID              = <2, *Anon Tail*>
```

Тесты – увеличение скорости в ~2 раза



НОВЫЕ ВОЗМОЖНОСТИ В Oracle Database 11g



ORACLE®
DATABASE 11^g

ORACLE®

Inline-подстановка в PL/SQL

PL/SQL 11g: Подстановка тела функции вместо вызова

```
begin
  PRAGMA INLINE(my_func, 'YES'); -- включаем подстановку

  for f in (select * from employees)
  loop
    x:= my_func(f.Name, f.amount) + 17; -- не вызов, а тело
                                         -- функции!
  end loop;

  PRAGMA INLINE(my_func, 'NO'); -- выключаем подстановку
  ...
end;
```

- Увеличение скорости выполнения: вместо передачи параметров, возврата управления и результатов
- Включение/выключение подстановки в коде

Inline-подстановка в PL/SQL

Новый уровень оптимизации (level 3)

```
alter session set plsql_optimize_level=3;
```

- Оптимизатор PL/SQL сам делает подстановку (*inline*) исходя из:
 - Размера процедуры/функции
 - Предполагаемой частоты вызова (вызов в цикле)
 - Числа и типов параметров
- Проведены тесты на системе E-Business Suite
 - Система состоит из больших PL/SQL-пакетов с небольшими процедурами (всего ~20 млн. строк кода PL/SQL)
 - Получено увеличение быстродействия на ~20%
- В общем случае зависит от структуры PL/SQL-кода



ORACLE®

Демонстрация

**Использование оптимизатора
PL/SQL**

Демонстрация

Оптимизация выключена

- Выключаем оптимизатор

```
alter session set plsql_optimize_level=0;
```

- Таблица accounts содержит ~ 200 тыс. записей
- Время выполнения: ~5 сек.

```
begin
  for f in (select * from accounts)
  loop
    s := f.Name;
  end loop;
end;
```

Демонстрация

Оптимизация включена (уровень 2)

- Включаем оптимизатор

```
alter session set plsql_optimize_level=2;
```

- Время выполнения: ~1 сек.

```
begin
  for f in (select * from accounts)
  loop
    s := f.Name;
  end loop;
end;
```

- Время выполнения уменьшилось в 5 раз!

Оптимизатор Oracle PL/SQL

Заключение

- Средство увеличения быстродействия хранимых процедур PL/SQL
- Тонкие средства настройки
- Проверьте значение параметра *PLSQL_OPTIMIZE_LEVEL* в вашей БД !
- Если он не выставлен – то все хорошо 😊

Литература

Ссылки

- PL/SQL Just Got Faster
 - http://www.oracle.com/technology/tech/pl_sql/files/Plsql_Just_Got_Faster.zip
- PL/SQL Performance Measurement Harness
 - http://www.oracle.com/technology/tech/pl_sql/files/Plsql_Performance_Measurement_Harness.zip
- Freedom, Order, and PL/SQL Optimization
 - http://www.oracle.com/technology/tech/pl_sql/files/CodeOrder.zip
- PL/SQL Performance — Debunking the Myths
 - http://www.oracle.com/technology/tech/pl_sql/pdf/Plsql_Performance_Debunking_The_Myths.pdf

Q&A



ORACLE®

Игорь Мельников
Старший консультант Oracle СНГ

Email : Igor.Melnikov@oracle.com
Phone : +7 (495) 641 14 00
Direct: +7 (495) 641 14 42
Mobile: +7 (915) 205 26 27