Процедуры, функции, пакеты

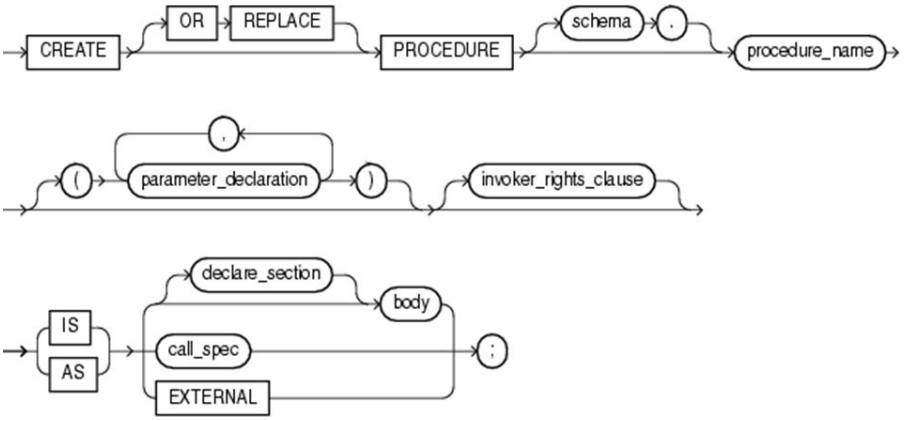
Оглавление

Структура процедуры/функции

Обычно, такие процедуры и функции называются хранимыми.

Хранимая процедура

На диаграмме отображено создание (пересоздание) хранимой процедуры:



г д е :

create or replace procedure – к л ю ч е в ы е с л о в а

schema.procedure_name – и м я с х е м ы и п р о ц е д у р ы ч е р е з т о ч к у

parameter_declaration – п е р е ч е н ь и м е н и т и п о в п а р а м е т р о в ч е р е з з а п я т у ю

invoker_rights_clause – п о д к а к и м и п р а в а м и б у д е т з а п у с к а т ь с я п р о ц е д у р а

external – в н е ш н я я п р о ц е д у р а (в н е ш н я я п о о т н о ш е н и ю к Б Д)

declare_section + body – с е к ц и я о б ъ я в л е н и я т и п о в и п е р е м е н н ы х + с е к ц и я и с п о л н е н и я

call_spec – у к а з а н и е т о г о , ч т о п р о ц е д у р а р е а л и з о в а н а н а JAVA и л и C

П а р а м е т р ы м о г у у к а з ы в а т ь с я :

IN – в х о д н о й п а р а м е т р

IN/OUT – в х о д н о й и в ы х о д н о й п а р а м е т р

OUT – в ы х о д н о й п а р а м е т р

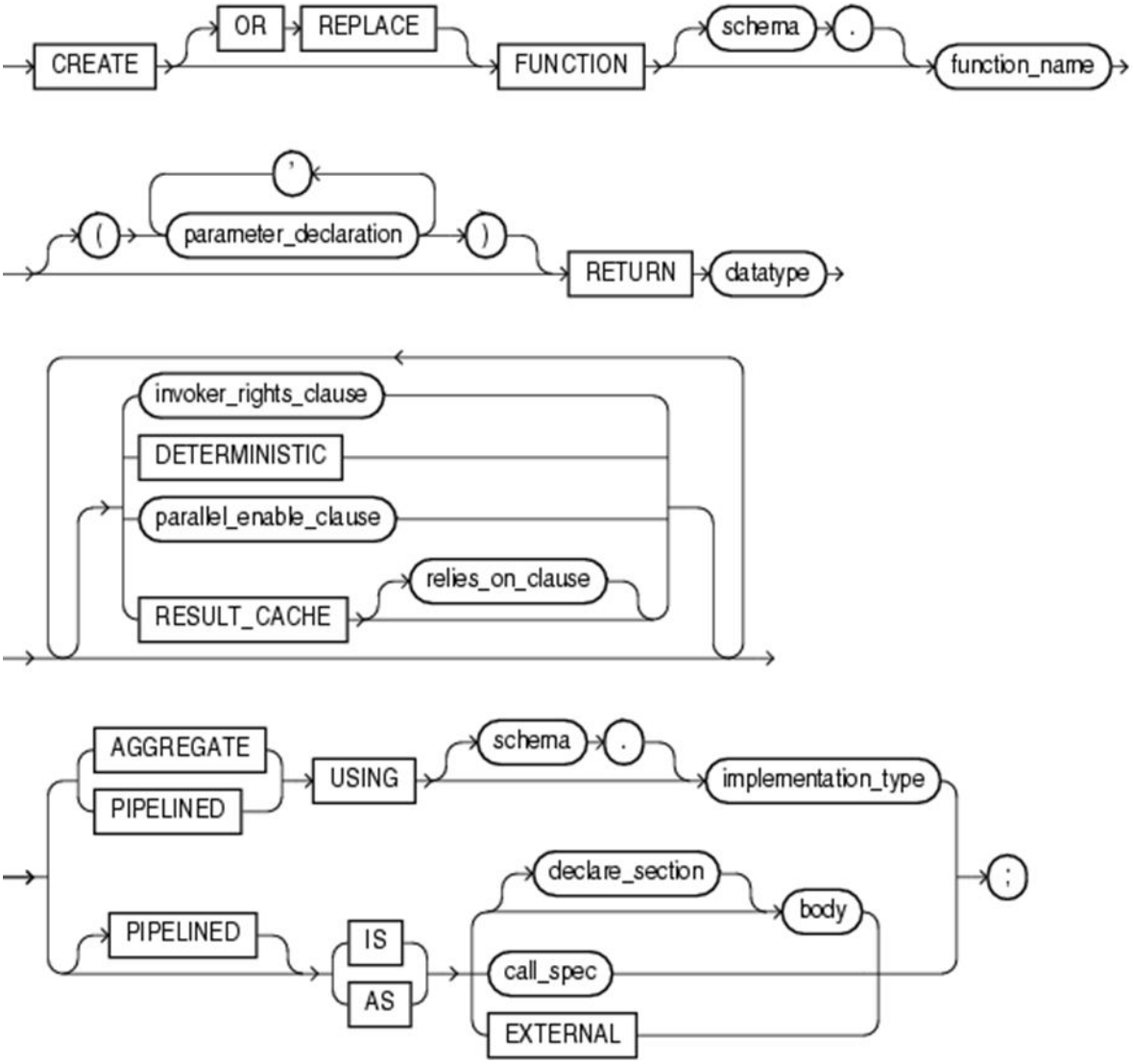
NOCOPY – п а р а м е т р п е р е д а е т с я п о с с ы л к е , а н е п о з н а ч е н и ю (п о в е д е н и е п о у м о л ч а н и ю)

DEFAULT(:=) expression- п р и с в о е н и е з н а ч е н и я п о у м о л ч а н и ю, т о г д а э т о т п а р а м е т р в в ы з о в е я в л я е т с я н е о б я з а т е л ь н ы м

Д о к у м е н т а ц и я Oracle

Хранимая функции

На диаграмме отображено создание(пересоздание) хранимой функции:



Отличительные особенности от процедуры:

DETERMINISTIC – сообщает ораклу, что для одних и тех же входных параметров значения функции не будут меняться

Parallel_enable_clause – возможность исполнения функции в параллель в конструкции Select from table(my_func(...)).

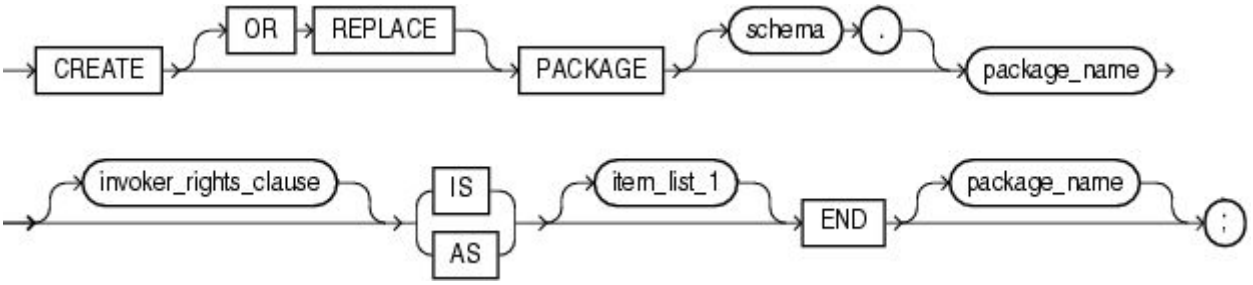
RESULT_CACHE relies_on_clause – кеширование результата исполнения функции в специальной области SGA. Relies on (table_name) сообщает ораклу о необходимости инвалидации закешированного результата.

AGGREGATE – агрегатная функция, на входе которой несколько строк, а на выходе один результат

PIPELINED – предназначена для создания табличной функции, строки внутри которой «выталкиваются» наверх конструкцией pipe row. Выходной тип функции – коллекция, выталкивается элемент коллекции

Структура пакета

На диаграмме отображено создание(пересоздание) спецификации пакета:



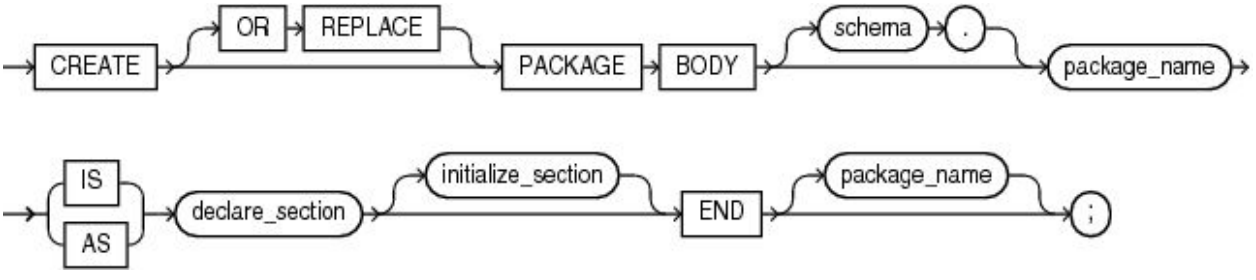
Пакет - это объект схемы, который группирует логически связанные PL/SQL-типы, переменные, константы, подпрограммы, курсоры и исключения. Пакет компилируется и хранится в БД, Даже можно рассматривать пакет как приложение (сомнительная цитата с сайта оракл).

Спецификация представляет собой *интерфейс* доступа к функционалу пакета.

Item_list_1

Тело пакета

На диаграмме отображено создание(пересоздание) тела пакета:



Детализацию можно посмотреть в разделе declare_section.

Тело пакета представляет собой реализацию интерфейса спецификации. При этом процедуры/функции объявленные в теле пакете, но не описанные в спецификации доступны только

изнутри пакета (аналог модификатора доступа *private*).

Перегрузка процедур/функций

Процедуры и функции пакета могут быть перегруженными, т.е. могут иметь одно имя, но разный набор параметров.

```
FUNCTION f_overload(p_1 NUMBER) RETURN NUMBER
FUNCTION f_overload(p_1 VARCHAR2) RETURN VARCHAR2
```

Глобальные переменные в пакетах

Спецификация пакета позволяет описывать переменные, которые будут являться глобальными в рамках всей сессии (за исключением компиляции пакета с ключевым словом PRAGMA SERIALLY_REUSABLE). Это значит, что переменной спецификации пакета я могу присвоить некое значение в процедуре пакета, а потом, например, в триггере получить значение этого поле. Данное поле будет хранить значение в рамках сессии (за исключением выше описанного случая).

Спецификация без тела

Спецификация без тела предназначена для хранения логической группы констант, исключений, курсоров, переменных, типов. Это, если так можно сказать, такая библиотека, но без исполняемого кода, без какой-то бизнес-логики.

INLINE Pragma

INLINE pragma указывает, что вызов подпрограммы должен быть встроен. Действует непосредственно на последующее описание или оператор, и на некоторые типы операторов

Если подпрограмма перегружена, встраивание будет применимо ко всем подпрограммам с таким именем

Pragma 'NO' перекрывает любое количество подряд идущих Pragma 'YES'

SERIALLY_REUSABLE Pragma

Если пакет не SERIALLY_REUSABLE, то его состояние хранится в User Global Area для каждого пользователя. Так как количество UGA растет линейно от количества пользователей, это уменьшает масштабируемость. Приложение блокирует память до окончания сессии. Иногда сессии длятся днями (Oracle Office)

Если пакет с SERIALLY_REUSABLE, то состояние пакета хранится в рабочей области в небольшом пуле в System Global Area. Состояние пакета сохраняется только на время серверного вызова. После серверного вызова рабочая область возвращается в пул.

Необходимо помнить, что при использовании SERIALLY_REUSABLE при каждом серверном вызове надо инициализировать переменные пакета

Conditional compilation Pragma

Условная компиляция использует директивы выбора, которые похожи на условные операторы, чтобы выбрать исходный код для компиляции. Все директивы условной компиляции построены из лексем управления препроцессора и текста PL/SQL.

Условная компиляций позволяет вам настраивать функциональность PL/SQL без удаления исходного кода.

- Использование новых возможностей СУБД и отключение этих возможностей, когда приложение работает на более старых версиях СУБД
- Активация отладчика или операторов трассировки в разработческом окружении и скрытие их, когда приложение работает в продуктовой среде

Разновидности:

➢ Preprocessor Control Tokens

Идентифицирует код, который обрабатывается перед компиляцией PL/SQL блока

$plsql_identifier ($IF, $THEN,$ELSE,$ELSEIF, $ERROR)

➢ Selection Directives

Выбирает исходный код для компиляции

➢ Error Directives

Генерирует пользовательскую ошибку во время компиляции

➢ Inquiry Directives

Предоставляет информацию об окружении компиляции - $$name

➢ Static Expressions

Выражение, которое может быть вычислено на этапе компиляции и которое может появляться в директивах условной компиляции

Package Writing Guidelines

- Познакомьтесь с пакетами, которые предоставляет Oracle DataBase и не пишите свои пакеты, которые дублируют эту функциональность (всего 239 пакетов тут Oracle Database PL/SQL Packages and Types Reference.
- Делайте пакеты такими, чтобы в будущем их можно было переиспользовать
- Разработка спецификации пакета всегда должна идти до тела пакета
- В спецификации пакета описывайте только те элементы, которые должны быть доступны снаружи пакета:
  - Это защитит других разработчиков от построения небезопасных зависимостей
  - При изменении спецификации надо все зависимые модули перекомпилировать, а при изменении только тела – не надо
- Присваивайте инициализационные значения в секции инициализации вместо секции декларации
  - Код инициализации получается более комплексный и лучше документированный

  - В секции инициализации всегда можно поймать исключение

- Объявляйте курсоры в спецификации, а описывайте в теле пакета

PL/SQL Source Text Wrapping Guidelines

Можно зашифровать следующие типы модулей, чтобы никто не мог посмотреть исходный код:

- Package specification
- Package body
- Type specification
- Type body
- Function
- Procedure

Ограничения:

- Проблемы с обратной совместимостью (нельзя завраплений в 11-м оракле файл перенести в 10-й)
- Это не безопасный путь для хранения паролей или имен таблиц (юзай Oracle Database Vault Administrator's Guide)
- Исходный код триггера завраапить нельзя, врапим процедуру и ее вызываем в триггере
- Врапь только тело пакета и не врапь спецификацию, это поможет другим разработчикам посмотреть информацию в спецификации, которая им может понадобиться при разработке
- Врапь файлы в самом конце разработки, ты не сможешь редактировать завраапленный файл. Если надо внести правки, правь оригинальный файл и потом врапь его снова
- Перед дистрибуцией завраапленного файла посмотри его в текстовом редакторе и убедись, что все важные части завраапленны

Dependencies

Определение объекта A ссылается на объект B, то A зависит от B, при этом:

- A - **dependent object** (зависимый объект) от B
- B - **referenced object** (объект, на который ссылается) A
- A зависит от B, B зависит от C, то A **direct dependent**(прямо зависит) от B, A **indirect dependent**(косвенно зависит) of C.
- Если изменение C делает невалидным B, B делает невалидным A. Это называется **cascading invalidation.**
- **Coarse-grained invalidation** (инвалидируются все зависимости) и **fine-grained invalidation** (в зависимости операций)

(USER/ALL/DBA)_DEPENDENCIES – описание зависимостей между объектами

Запуск удаленных процедур (через dblink)

DBLink позволяет как запрашивать данные с удаленного сервера БД, так и запускать удаленные процедуры. Синтакис в обоих случаях идентичный:

select sysdate from **dual@DB_LINK_TEST**; -- обращение к удаленной таблице

select **get_test@DB_LINK_TEST** from dual; -- вызов удаленной функции

Jobs

- Job предназначен для периодического запуска задач.
- Указываем, *что, когда* и с *каким интервалом* запускать.
- Информация о джобах - DBA_JOBS
- О выполняющихся джобах - DBA_JOBS_RUNNING
- Создание/управление джобами – пакет DBMS_JOB

Особенность:

- Транзакционный (нет commit-а – нет джоба)
- Нет имени, номер задать нельзя (это выходной параметр)

Scheduler

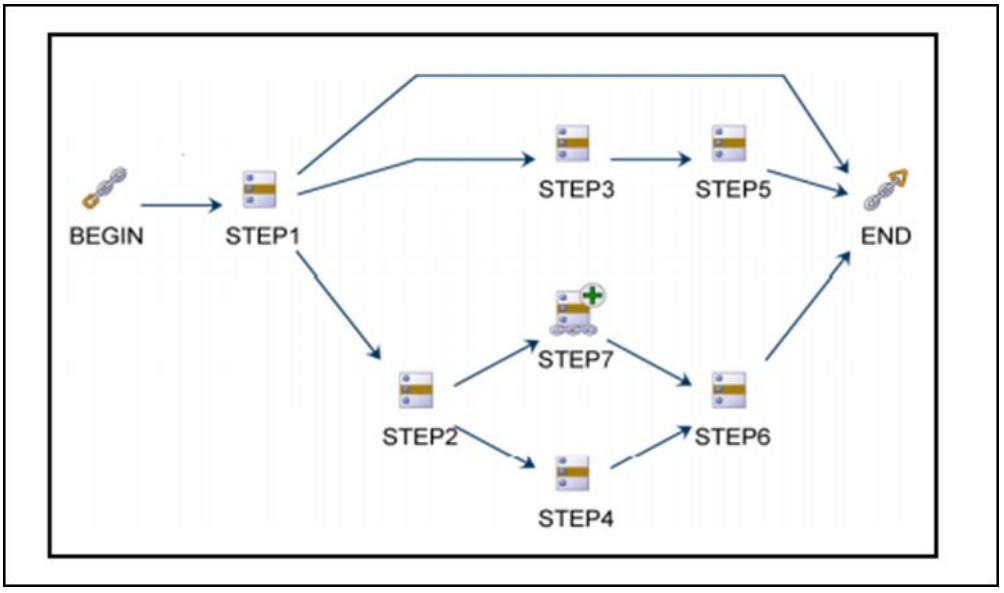
DBMS_JOB считается устаревшей технологией и Oracle рекомендует использовать более мощный Scheduler (с 10g):

- Логирование запуска джобов
- Простой и мощный синтаксис (проще, но гораздо мощнее cron-а)
- Запуск джобов за пределами БД, в ОС
- Управление ресурсами между различными классами джоба
- Возможность просмотра и графического управлений (Oracle SQL developer)
- Возможность использования ресурсного плана (например, не запускать джоб при загрузке процессора более 70%)
- Возможность указать максимальную длительность работы джоба
- Возможность указать последовательно задачи, которые надо выполнить, проверку результатов выполнения задачи ветвление

Информация о джобах щедуллера - DBA_SCHEDULER_JOBS
О выполняющихся джобах - DBA_SCHEDULER_RUNNING_JOBS
Создание/управление scheduler-джобами - DBMS_SCHEDULER

Одна из центральных возможностей Scheduler-а – это chain-ы.

Chain(цепочка) – именованная последовательность задач связанная вместе. Chain-ы являются средством, с помощью которого можно реализовать зависимости между задачами, в которых задачи стартуют в зависимости от результата одной или более предыдущих задач.



Тема scheduler-а очень объемна и для погружения в нее требуется отдельный семинар