1. **Язык PL/SQL, его структура, основные операторы.**
2. **Курсоры, операторы работы с курсором, оператор SELECT . . . INTO.**
3. **Процедуры, функции, пакеты.**
4. **Триггеры, их основные свойства и значение**

Язык PL/SQL. Основные характеристики.

SQL (*Structured Query Language*) – непроцедурный язык

PL/SQL (*Procedural Language extensions to the Structured Query Language*) – процедурные языковые расширения SQL

PL/SQL

Создан по образцу языка Ada (в честь математека Ады Лавелейс, считающейся первым в мире программистом).

Тесно интегрирован с SQL(SQL-инструкции можно вызывать из процедурной программы без помощи промежуточного API.

Язык PL/SQL. Основные характеристики.

- **Стандартизованный и переносимый язык разработки приложений для баз данных Oracle.**
- **Встроенный язык (функционирует в конкретной хост-среде). Программы на PL/SQL запускаются из БД.**
- **Высокопроизводительный, высокоинтегрированный язык доступа к БД. Более всего подходит для написания высокоэффективного кода для доступа к БД Oracle.**

Для разработки приложений клиент/сервер с помощью PL/SQL можно улучшить производительность приложений и системы:

- вместо операторов SQL используются скомпилированные программы;
- значительно сокращается сетевой трафик между клиентом и сервером.

Язык PL/SQL.
Базовый синтаксис. Блок PL/SQL

Блок PL/SQL состоит из четырех секций:

- ✔ **секция заголовка (header section)**
- ✔ **секция объявления (declaration section)**
- ✔ **выполняемой секции (execution section)**
- ✔ **секция исключений (exception section)**

Выполняемая секция является обязательной.

Язык PL/SQL. Блок PL/SQL

```
DECLARE
-- определение переменных, констант,
  новых
-- типов данных, курсоров и т.д.
BEGIN
-- набор операторов исполняемого раздела
BEGIN
-- набор операторов вложенного блока
END;
EXCEPTION
-- набор операторов драйверов для
-- обработки исключительных ситуаций в
-- программе
END;
/
```

Блок PL/SQL

- ***Секция заголовка*** содержит спецификацию процедуры, функции, пакета или триггера. Включает в себя название блока, описание входных и выходных параметров.
- ***Секция объявлений*** предназначена для объявления переменных, констант, курсоров, которые будут использоваться в выполняемой секции процедуры, функции или триггера. Расположена сразу после секции заголовка, если она есть, и перед выполняемой секцией.
- ***Выполняемая секция*** (тело) содержит один или более операторов PL/SQL. Начинается со слова BEGIN и заканчивается словом EXCEPTION, если есть секция исключений, или словом END.
- ***Секция исключений*** содержит обработчики исключительных ситуаций. *Исключительной ситуацией* называют такую ситуацию, когда дальнейшее выполнение выполняемой секции не имеет смысла.

В анонимных блоках и в триггерах для создания секции заголовка указывается ключевое слово DECLARE. Во всех остальных случаях ключевое слово DECLARE не используется.

Блок PL/SQL

Блоки могут быть вложены друг в друга. Самый "верхний" блок PL/SQL называется ***базовым*** и должен заканчивается символом "/". Этот символ сообщает серверу, что можно приступать к компиляции введенного кода.

Блок, не имеющий заголовка, называется ***анонимным***. Вложенными могут быть только анонимные блоки. Они используются в функциях, процедурах и триггерах. Анонимный базовый блок не сохраняется на сервере, а выполняется сразу. Если же базовый блок имеет заголовок, то он хранится на сервере в виде скомпилированной процедуры, функции, пакета или триггера(в зависимости от типа заголовка).

Управляющие структуры PL/SQL

Операторы условного перехода IF

- **IF-THEN**. Форма условного оператора для проверки простых условий. Если условие верно (TRUE), то выполняются указанные операторы. Если условие не выполняется (FALSE), то управление передается на следующий оператор.
- **IF-THEN-ELSE**. Эта форма аналогична предыдущей, но при невыполнении условия (FALSE) управление передается на операторы, указанные после ELSE.
- **IF-THEN-ELSIF**. Этот формат является альтернативой использованию вложенных операторов IF-THEN-ELSE.

Управляющие структуры PL/SQL

Оператор CASE

- ✔ **Простой**
- ✔ **Поисковый**

Простой. Связывает одну или несколько последовательностей операторов с соответствующим значением.

```
CASE выражение
WHEN результат1 THEN
Операторы1
WHEN результат1 THEN
Операторы1
…
ELSE
ОператорыELSE
END CASE;
```

Управляющие структуры PL/SQL

Оператор CASE

Поисковый. Выбирает для выполнения одну из последовательностей операторов в зависимости от результатов вычисления списка логических условий.

```
CASE
WHEN выражение1 THEN
Операторы1
WHEN выражение2 THEN
Операторы2
…
ELSE
ОператорыELSE
END CASE;
```

Управляющие структуры PL/SQL

Оператор GOTO

Оператор безусловного перехода

```
GOTO имя_метки;
 …
<<имя_метки>>
Операторы_после_метки;
```

- За меткой должен следовать хотя бы один исполняемый оператор
- Метка должна находиться в пределах области действия оператора **GOTO** (функция, процедура, анонимный блок, оператор IF, оператор LOOP, обработчик исключения, оператор CASE)
- Метка должна находиться в той же части блока, что и оператор **GOTO**

Оператор NULL

Используется для указания компилятору “ничего не делать”: **NULL;**

Управляющие структуры PL/SQL

Операторы цикла

Циклы позволяют организовать многократное выполнение одного и того же участка программы до полного завершения обработки.

- **Простой цикл.**
- **Цикл `FOR`.**
  1. **с числовым счетчиком**
  2. **с курсором**
- **Цикл `WHILE`.**

Управляющие структуры PL/SQL

Синтаксис:

- **Простой цикл.**

```
LOOP
 EXIT WHEN условие_завершения;
 Операторы_цикла;
END LOOP;
```

- **Цикл FOR.**

```
FOR переменная_цикла IN нижняя_граница_диапазона
   .. верхняя_граница_диапазона | имя_курсора
LOOP
Операторы_цикла;
END LOOP;
```

- **Цикл WHILE.**

```
WHILE условие_завершения
LOOP
Операторы_цикла;
END LOOP;
```

Набор символов PL/SQL. Переменные.

- ✔ **Прописные и строчные буквы**
- ✔ **Цифры от 0 до 9**
- ✔ **Знаки ( ) + √ * / > < = ! ~ ; : . ' @ % , " # $ ^ & _ { } ? [ ]**

Переменные рассматриваются как имена ячеек, используемых для обработки и хранения элементов данных.

- ❑ Переменные должны начинаться с буквы (A-Z)
- ❑ За первой буквой переменной может следовать одна или несколько букв, цифр (0-9) или специальных символов $, # или _
- ❑ Длина имени переменной не может превышать 30 знаков
- ❑ Имя переменной не может содержать пробелы

Типы данных

Числовые

Символьные

Даты и времени

Логический

Составные типы: записи и коллекции

Двоичные типы

ROWID и UROWID

REF CURSOR

Типы данных для поддержки Internet

ANY

Объекты (типы данных, определяемые пользователем)

Предопределенные типы данных объявлены в пакете STANDART.

Числовые типы

Основные числовые типы: NUMBER, PLS_INTEGER, BINARY_INTEGER

NUMBER – единственный числовой тип, непосредственно поддерживаемый ядром БД.

Number (precision, scale), где

precision – число значащих цифр (от 1 до 38) .

scale – число цифр после запятой (от -84 до 127).

PLS_INTEGER

Позволяет хранить целые числа в диапазоне от – 2 147 483 647 до 2 147 483 647. Был разработан для увеличения скорости вычислений.

BINARY_INTEGER

Позволяет хранить целые числа со знаком в двоичном формате в диапазоне от – 2 147 483 647 до 2 147 483 647. Не использует встроенную машинную арифметику. Обеспечивает ускорение вычислений при большом объеме операций с целочисленными значениями.

Числовые подтипы

Подтипы введены для достижения совместимости с типами ANSI, SQL, SQL/DS, DB2 и представляют собой альтернативные имена для основных типов.

NUMBER:

```
DEC, DECIMAL, DOUBLE PRECISION,
  FLOAT, INT, INTEGER, REAL,
  NUMERIC, SMALLINT
```

BINARY_INTEGER:

```
NATURAL, NATURALN, POSITIVE,
  POSITIVEN, SIGNTYPE
```

Символьные типы. Набор символов.

Набор символов – совокупность символов и соответствующий ей набор битовых последовательностей для представления этих символов в машинном виде.

`ASCII, CP-1251, UNICODE`

Классифицируется по признакам:

- ✔ многобайтовый / однобайтовый
- ✔ фиксированной / переменной длины

`ASCII` – однобайтовый набор символов фиксированной длины

`UNICODE UTF-8` – многобайтовый набор символов переменной длины

`UNICODE UTF-8` – многобайтовый набор символов фиксированной длины

Набор символов.

С каждой БД ORACLE связаны два набора символов:

Набор символов базы данных. Используется для представления значений столбцов типа CHAR и VARCHAR2, имен таблиц, столбцов, переменных, строковых литералов.

Набор символов национального алфавита. Используется для представления значений столбцов типа NCHAR и NVARCHAR2, строковых литералов с префиксом N.

Запрос информации об используемом наборе символов:

```
SELECT * FROM nls_database_parameters
  WHERE parameter IN('NLS_CHARACTERSET',
  'NLS_NCHAR_CHARACTERSET');
```

Строковые типы данных

Набор символов	Фиксированная длина	Переменная длина
Базы данных	**CHAR** *Символьные строки фиксированной длины (от 1 до 32 767 байт)*	**VARCHAR2** *Символьные строки переменной длины (от 1 до 32 767 байт)*
Национальный	**NCHAR** *Строки фиксированной длины , состоящие из символов национального алфавита*	**NVARCHAR2** *Строки переменной длины , состоящие из символов национального алфавита*

Строковые подтипы

VARCHAR2:

CHAR VARYING, CHARACTER VARYING, STRING, VARCHAR

CHAR:

CHARACTER

NCHAR:

NATIONAL CHAR, NATIONAL CHARACTER

NVARCHAR2:

NATIONAL CHAR VARYING, NATIONAL CHARACTER VARYING, NCHAR VARYING

Дата и время

DATE

Год, месяц, день, часы, минуты, секунды

TIMESTAMP

Дата и время с точностью до миллиардной доли секунды.

INTERVAL

Момент, интервал, период.

Дата и время

Исходный тип данных – `DATE`. Используется для хранения значения даты или даты и времени.

Ограничения для типа данных `DATE`:

- Точность времени – до секунды
- Не содержит информации о часовом поясе

Тип данных `TIMESTAMP`(временная метка). Используется для хранения времени с точностью до миллиардной доли секунды.

`TIMESTAMP`. Хранит дату и время без информации о часовом поясе

`TIMESTAMP WITH TIME ZONE`. Хранит дату и время с информацией о часовом поясе

`TIMESTAMP WITH TIME ZONE`. Хранит дату и время, соответствующие локальному часовому поясу

Дата и время

Типы данных INTERVAL

Момент – временная точка с некоторой точностью(до часа, до минуты)

Интервал – количество времени(час, три часа, пять минут)

Период – интервал, который начинается и заканчивается в заданные моменты времени.

INTERVAL YEAR TO MONTH

-- интервал времени в годах и месяцах

INTERVAL DAY TO SECONDS

-- интервал времени в днях, часах, минутах и секундах (включая доли секунды)

Тип данных BOOLEAN

Допустимые значения – `TRUE`, `FALSE`, `NULL`.

СУБД `ORACLE` не поддерживает тип данных BOOLEAN.

Следует учитывать в операторах сравнения, что логическая переменная может принимать значение `NULL`.

Составные типы данных

RECORD (запись) похожа на строку из таблицы базы данных, обрабатывается как единое целое. Не имеет собственного значения. Значение имеет каждый компонент записи.

- ✔ Запись на основе курсора
- ✔ Запись на основе таблицы
- ✔ Запись, определяемая программистом

TABLE (коллекция) – составной тип данных, предназначенный для хранения одномерных массивов в программах PL/SQL.

- ✔ Ассоциативные массивы
- ✔ Вложенные таблицы
- ✔ Массив типа VARRAY

Записи

Записи трактуются в языке PL/SQL Oracle8 как совокупность разнотипных компонентов, которые можно хранить в столбцах реляционных таблиц, передавать в качестве параметров и т.п.

```
TYPE AgendaItem IS RECORD (
  subject   VARCHAR2 (100),
  duration  TimeInterval);

item_info AgendaItem;
```

Типы двоичных данных

Двоичные данные являются неструктурированными, не обрабатываются и не интерпретируются Oracle: `RAW`, **`LONG RAW`**, **`BFILE`**, **`BLOB`**

`RAW` предназначен для хранения и обработки двоичных данных малых объемов.

`BLOB(Binary Large Object)` предназначен для работы с большими объемами информации(звук, изображения). Переменная этого типа содержит локатор `LOB`, указывающий на хранящийся в базе данных большой двоичный объект.

`BFILE` – двоичный файл. Переменная этого типа содержит локатор файла, указывающий на файл операционной системы, хранящийся вне базы данных. Oracle интерпретирует содержимое файла как двоичные данные.

Типы данных ROWID и UROWID

Представление адреса строки в таблице.

`ROWID` – уникальный адрес строки таблицы(двоичное значение, определяющее физический адрес каждой строки в таблице). Создается при добавлении строки в таблицу. Позволяет повысить скорость обработки, т.к. доступ к строке по идентификатору выполняется быстрее, чем по ключу. Используется при повторном доступе к строке.

`UROWID` – логическая позиция строки в индекс-таблице. Используется для сохранения идентификатора строки.

Тип данных REF CURSOR

Позволяет объявлять курсорные переменные для использования в статических и динамических SQL-инструкциях.

Курсорная переменная – это переменная, указывающая на курсор. Предоставляет механизм передачи результатов запроса (выбранных строк из курсора).

- ✔ Курсорную переменную можно связывать с разными запросами
- ✔ Курсорную переменную можно передать в качестве аргумента процедуре или функции(совместное использование вызываемой и вызывающей программой результирующего набора строк)
- ✔ Курсорные переменные сохраняют все возможности статических курсоров
- ✔ Значения курсорных переменных можно переприсваивать

Поддержка ИНТЕРНЕТ

XMLType и URIType

XMLType позволяет хранить в базе данных **XML-** данные.

URIType – основной тип.

URI - Uniform Resource Information

Подтипы:

HttpUriType идентифицирует web-страницу

DbUriType – подтип **UriType**, поддерживающий представленный в виде выражения **Xpath URL**

XDBUriType - подтип **UriType**, поддерживающий **URL** и идентифицирующий объекты **Oracle XML DB.**

Типы данных ANY

Семейство типов any предназначено для выполнения операций над данными неизвестного типа.

- **`AnyData`** – содержит одиночное значение любого типа: скалярная величина, объект, созданный пользователем, массив типа VARARRAY и т.д.
- **`AnyDataSet`** – содержит набор однотипных значений любого типа
- **`AnyType`** – содержит описание типа (тип без данных)

Объявление данных

Объявление переменных

При объявлении переменной ей присваивается имя, задается тип и выделяется память для ее хранения.

```
имя тип_данных [NOT NULL] [DEFAULT значение_по
   _умолчанию]|[:= значение_по _умолчанию];
```

Примеры:

```
total NUMBER;
account CHAR(15);
userName VARCHAR2(50);
dateN DATE NOT NULL DEFAULT SYSDATE;
```

Объявление константы

```
имя CONSTANT тип_данных [DEFAULT значение_по
   _умолчанию]|[:= значение_по _умолчанию];
```

Пример:

```
author CONSTANT VARCHAR2(80) DEFAULT 'Ivanov I.';
NMAX CONSTANT PLS_INTEGER := 25;
```

Объявление с ограничениями

Объявление с указанием ограничений допустимых значений.

```
-- объявление без ограничений:
-- для хранения переменной выделяется 38 разрядов
no_limits NUMBER;

-- объявление с ограничениями:
-- требуется меньше памяти
small NUMBER(1);
large NUMBER(25,6);
title VARCHAR(200);
```

Объявления с привязкой

Устанавливается тип данных на основе типа уже определенной структуры данных. Виды привязки:

- **Скалярная привязка.** С помощью атрибута %TYPE переменная определяется на основе типа столбца таблицы базы данных или другой скалярной переменной
- **Привязка к записи.** Через атрибут %ROWTYPE определяется переменная на основе таблицы базы данных или предопределенного явного курсора.

Синтаксис:

```
имя_переменной_ тип_атрибута %TYPE [DEFAULT];
имя_переменной_ имя_таблицы | имя_курсора%ROWTYPE
   [DEFAULT];
```

где

`тип_атрибута` – имя ранее объявленной переменной или спецификация столбца таблицы в формате `таблица.столбец`

Обработка исключений

Системное исключение. Инициируется исполняемым ядром PL/SQL(NO_DATA_FOUND, TOO_MANY_ROWS, INVALID_NUMBER).

Исключение, определяемое программистом. Определяется в коде PL/SQL, специфично для данного приложения. Имя исключения связывается с конкретной ошибкой Oracle с помощью директивы компилятора EXCEPTION_INIT. Присвоить номер исключению и создать для него описание можно с помощью процедуры **`RAISE_APPLICATION_ERROR.`**

Инициировать исключение. Остановить выполнение текущего блока PL/SQL путем уведомления исполняемого ядра об ошибке.

Обработать исключение. Перехватить ошибку, передав управление обработчику исключений.

Неименованное (анонимное) исключение. Исключение, с которым связан номер ошибки и описание. Не имеет имени, поэтому не может быть использовано в операторе RAISE или предложении WHEN обработчика исключений.

Именованное исключение. Исключение, которому присвоено имя.

Обработка исключений

Раздел обработки исключений:

```
EXCEPTION
    WHEN имя_искл_1
    THEN
    операторы_обработчика_искл_1;
    . . .
    WHEN имя_искл_N THEN
    операторы_обработчика_искл_N;
    END;
```

Объявление именованных исключений

```
имя_искл_1 EXCEPTION;
```

Обработка исключений

Связывание имени исключения с кодом ошибки

Коды ошибок – от -20999 до -20000.

SQLCODE – функция, возвращающая код последней сгенерированной ошибки.

Директива EXEPTION_INIT позволяет связать имя объявленной ошибки с некоторым кодом.

```
DECLARE
    имя_исключения EXCEPTION;
    PRAGMA EXEPTION_INIT(имя_исключения,
   целое_число);
```

Ключевое слово **PRAGMA** указывает, что часть оператора после нее является директивой компилятора. Не включается в исполняемый код.

Обработка исключений

Инициирование исключений

- **Оператор RAISE**

```
RAISE имя_исключения;
-- инициирование системных и объявленных в текущем
   блоке исключений
RAISE имя_пакета.имя_исключения;
-- инициирование исключения, объявленного в пакете
RAISE;
-- повторное инициирование исключения в обработчике
   исключения
```

- **Процедура RAISE_APPLICATION_ERROR**

Инициирование специфических для приложения исключений. Позволяет связать с исключением сообщение об ошибке.

```
RAISE_APPLICATION_ERROR (ERRNUM, ERRMES)
```

ERRNUM – номер ошибки от -20000 до -20999

Процедуры

Создание процедуры

```
CREATE [OR REPLACE] PROCEDURE имя_процедуры
   [(аргумент1 [{IN | OUT |IN OUT}] тип [:= значение_по
   умолчанию | DEFAULT], . . .,
   аргументN [{IN | OUT |IN OUT}] тип [:= значение_по
   умолчанию | DEFAULT] {IS | AS}
   тело_процедуры
```

имя_процедуры - имя создаваемой процедуры,

аргумент - имя параметра процедуры,

тип - тип соответствующего параметра,

тело процедуры - блок PL/SQL, содержащий раздел объявлений, выполняемый раздел и раздел исключительных ситуаций.

```
CREATE OR REPLACE PROCEDURE имя_процедуры AS
/* Раздел объявлений. */
   BEGIN              /* Выполняемый раздел. */
/* Раздел исключительных ситуаций. */
   EXCEPTION
END [имя_процедуры];
/
```

Процедуры

Процедура может содержать только операторы DML, управляющие конструкции и вызовы процедур и функций. В процедуре нельзя использовать операторы DDL в число которых входят CREATE, ALTER, DROP.

Вызов процедур:

Для вызова процедур используется оператор execute или exec.

```
execute имя_процедуры;
```

Удаление процедур

```
DROP PROCEDURE имя_процедуры.
```

Хранимая процедура - приложение, объединяющее запросы и процедурную логику и хранящееся в базе данных. Позволяет содержать вместе с БД достаточно сложные программы, выполняющие большой объем работы без передачи данных по сети и взаимодействия с клиентом.

Функции

Создание функции

```
CREATE [OR REPLACE] FUNCTION имя_функции
   [(аргумент1 [{IN | OUT |IN OUT}] тип [:=
   значение_по умолчанию | DEFAULT], . . .,
   аргументN [{IN | OUT |IN OUT}] тип [:=
   значение_по умолчанию | DEFAULT]
    RETURN возвращаемый_тип {IS | AS}
   тело_ функции
```

Оператор RETURN применяется для возврата управления программой и результата выполнения функции в вызывающую среду. Завершение функции без оператора RETURN является ошибкой.

Вызов процедуры является оператором PL/SQL, вызов функции - это часть некоторого выражения.

Удаление функций

```
DROP FUNCTION имя_функции;
```

Пакеты

Пакет - это конструкция PL/SQL, позволяющая хранить связанные объекты в одном месте.

Преимущества пакета:

- облегчает процесс разработки;

- дополнительная функциональность (глобальные переменные);

- повышает производительность приложений.

Пакет состоит из двух частей: спецификации (описания) и тела, каждая из которых хранится по отдельности в словаре данных. Спецификация является обязательной частью.

```
CREATE [OR REPLACE] PACKAGE имя_пакета
  {IS |AS}
-- список всех общедоступных элементов
  пакета;
END [имя_пакета];
```

Пакеты

Тело пакета (package body) - это объект словаря данных, содержащий код реализации пакета. Описание процедур и/или функций должно соответствовать спецификации.

```
CREATE [OR REPLACE] PACKAGE
  BODY имя_пакета {IS |AS}
-- код для всех элементов,
  объявленных в спецификации;
END [имя_пакета];
```

Вызов элементов пакета:

```
имя_пакета.имя_элемента
```

Курсоры

Курсор - это имя запроса или указатель на контекстную область, с помощью которого программа PL/SQL управляет этой областью и ее состоянием во время обработки.

Неявные курсоры. Создается автоматически при выполнении инструкций **SELECT … INTO, INSERT, UPDATE, DELETE.** Структура запроса на выборку для неявного курсора:

```
SELECT список_столбцов INTO список_переменных
  FROM список_таблиц [WHERE условие … ];
```

Используются для поиска данных на основе значений первичного ключа. Операции открытия, выборки строк и закрытия производятся автоматически.

Инициируемые исключения:

- ✔ **NO_DATA_FOUND.** По запросу не найдено ни одной строки.
- ✔ **TOO_MANY_ROWS.** Инструкция SELECT вернула несколько строк.

Явные курсоры. Объявляется явно через ключевое слово CURSOR в разделе объявлений. Может использоваться многократно.

```
CURSOR имя_курсора IS оператор_SELECT;
```

Курсоры. Обработка явного курсора

1) Объявление курсора

CURSOR *имя_курсора* IS *оператор_select;*

2) Открытие курсора для запроса

OPEN *имя_курсора;*

3) Выбор результатов в переменные PL/SQL

Производится считывание строк из курсора.

FETCH *имя_курсора* INTO *список_переменных* | *запись_PL/SQL;*

4) Закрытие курсора

Курсор следует закрыть и освободить отведенные для него ресурсы.

CLOSE *имя_курсора;*

оператор_select - запрос, который будет обрабатываться.

список_переменных - список объявленных переменных PL/SQL, разделенных запятыми

запись_PL/SQL - предварительно объявленная запись PL/SQL.

Атрибуты курсоров.

Состояние курсора определяется через его атрибуты:

имя_курсора%атрибут

имя_курсора – имя объявленного явного курсора или **SQL** для неявного курсора

Атрибуты курсора

%ISOPEN	открыт курсор (true, false). Для неявного курсора всегда возвращает false
%ROWCOUNT	число строк, выбранных после открытия курсора
%NOTFOUND	последняя команда выборки возвратила строку, то %NOTFOUND – false, %FOUND – true, и наоборот
%FOUND	

Коллекции

Коллекция – это составной тип данных, предназначенный для хранения одномерных массивов PL/SQL. Коллекции используются для хранения множества однотипных элементов в кодах PL/SQL и таблицах базы данных.

Операции, при которых целесообразно использовать коллекции:

- Эмуляция двунаправленных курсоров и курсоров с произвольным доступом
- Хранение списков подчиненной информации в столбцах таблицы – в виде вложенной таблицы или массива VARRAY. Производительность поиска заметно увеличивается.
- Отслеживание элементов данных, отобранных в программе для специальной обработки.
- Кэширование статичной информации базы данных

Коллекции

Типы коллекций:

1. **Ассоциативные массивы** – одномерные неограниченные разреженные коллекции, которые можно обработать только в PL/SQL. Индексирование содержимого производится посредством значений типа VARCHAR2 или PLS_INTEGER.
2. **Вложенные таблицы** – одномерные несвязанные коллекции. Первоначально заполняются полностью, но после удаления элементов могут стать разреженными. Можно определять и в кодах PL/SQL и в таблицах БД.
3. **Массив типа VARRAY** – одномерные коллекции ограниченного размера, не могут быть разреженными. Можно определять и в кодах PL/SQL и в таблицах БД.

Способы задания типа коллекции:

1. **`CREATE TYPE`** для определения вложенной таблицы и типа **`VARRAY`** в базе данных.
2. **`TYPE … IS`** для определения типа коллекции в программе PL/SQL

Коллекции

Терминология коллекций

Коллекция называется ***ограниченной***, если заранее определены границы возможных значений индексов или номеров ее элементов. Если верхняя или нижняя граница не указана, коллекция называется ***неограниченной***.

Коллекция называется ***плотной***, если все ее элементы определены и каждому из них присвоено некоторое значение. Коллекция считается *разреженной*, если отдельные ее элементы отсутствуют.

Внешняя таблица – таблица, содержащая столбец типа вложенной таблицы или массив VARRAY.

Вложенная таблица – коллекция, содержащаяся в столбце таблицы.

Коллекции

Объявление ассоциативного массива:

1) **TYPE** … **TABLE**. Задается конкретная структура ассоциативного массива
2) Объявляется коллекция на основе заданного типа.

```
имя_коллекции табличный_тип
```

```
TYPE имя_типа_таблицы IS TABLE OF тип_данных [NOT NULL]
     INDEX BY [BINARY_INTEGER | подтип_типа
     BINARY_INTEGER | VARCHAR2(максимальный размер)];
```

Варианты индексов:

```
INDEX BY PLS_INTEGER
INDEX BY NATURAL
INDEX BY POSITIVE
INDEX BY VARCHAR2(max_size)
INDEX BY таблица.столбец%TYPE
INDEX BY пакет.переменная%TYPE
INDEX BY подтип
```

Для поля табличного типа:

- Скалярный тип данных

1) Тип данных с привязкой

Примеры объявления коллекций

```
-- создание типа
TYPE list_of_dates _t IS TABLE OF DATE;
TYPE list_of_names_t IS TABLE OF
  VARCHAR2(100) INDEX BY BINARY_INTEGER;
TYPE emp IS TABLE OF employees%ROWTYPE
  INDEX BY BINARY_INTEGER;

-- объявление коллекции на основе ранее
-- заданного типа
birthdays list_of_dates_t;
family list_of_names_t;
personal emp;
```

Встроенные методы коллекций

Метод	Описание
Функция **COUNT**	Возвращает текущее количество элементов коллекции
Процедура **DELETE**	Удаляет из вложенной таблицы один или несколько элементов. Для VARRAY удаляет все содержимое
Функция **EXISTS**	Возвращает значение TRUE или FALSE, указывающее, существует ли в коллекции заданный элемент.
Процедура **EXTEND**	Увеличивает количество элементов в коллекции. Неприменима к ассоциативным массивам.
Функции **FIRST, LAST**	Возвращают индексы первого/последнего элемента коллекции
Функция **LIMIT**	Возвращает максимальное количество элементов в массиве VARRAY.
Функции **PRIOR, NEXT**	Возвращают индексы элементов, предшествующих заданному/следующего за ним
Функция **TRIM**	Удаляет элементы с конца коллекции.

Триггеры

Триггеры – это особые хранимые процедуры, запускаемые в ответ на происходящие в базе данных события. Триггер выполняется, когда происходит событие, запускающее этот триггер (операторы INSERT, UPDATE или DELETE).
Используются для:

- ✔ Реализации сложных ограничений целостности данных, которые невозможно осуществить через описательные ограничения
- ✔ Сложные проверки для защиты информации
- ✔ Слежения за информацией, хранимой в таблице, путем записи вносимых изменений и пользователей, вносящих эти изменения
- ✔ Автоматического оповещения других программ о том, что делать в случае изменения информации, содержащейся в таблице

Триггеры

События и компоненты процесса обработки данных, с которыми могут быть связаны триггеры:

- **Инструкции DML.** Триггер запускается в ответ на вставку, удаление или обновление строки в таблице базы данных. Цель – проверка значений, устанавливаемых по умолчанию, аудит изменений, запрет определенных DML-инструкций.
- **Инструкции DDL.** Триггер запускается в ответ на выполнение DDL-инструкций. Цель – аудит и запрет определенных операций.
- **События базы данных.** Триггер срабатывает при запуске и останове базы данных, подключении и отключении сервера, возникновении ошибок Oracle.
- **Триггеры INSTEAD OF.** Запускаются непосредственно перед операциями вставки, удаления или обновления. Управляют операциями над представлениями. Позволяют преобразовывать необновляемые представления в обновляемые.
- **Приостановленные инструкции**. Триггер запускается при условии возникновения некоторой проблемы(исчерпана квота, недостаточно табличного пространства).

Концепции триггеров

Триггер BEFORE. Вызывается до внесения каких-либо изменений.

Триггер AFTER. Выполняется после того, как произведены изменения.

Триггер уровня инструкции. Выполняется для отдельной SQL-инструкции.

Триггер уровня записи. Вызывается для отдельной записи, обрабатываемой SQL-инструкцией.

Псевдозапись NEW. Структура данных с именем NEW, обладающая такими же свойствами, что и запись PL/SQL. Доступна только внутри триггеров обновления и вставки, содержит значение модифицированной записи после внесения изменений.

Псевдозапись OLD. Структура данных с именем OLD, обладающая такими же свойствами, что и запись PL/SQL. Доступна только внутри триггеров обновления и удаления, содержит значение модифицированной записи до внесения изменений.

Предложение WHEN. Часть триггера DML, определяющая условия выполнения кода триггера.

В триггере нельзя задавать операторы управления транзакциями: COMMIT, ROLLBACK или SAVEPOINT.

Создание триггера DML

```
CREATE [OR REPLACE] TRIGGER
  имя_триггера
{BEFOR | AFTER | INSERT | DELETE |
  UPDATE | UPDATE OF список_столбцов
  } ON имя_таблицы
[FOR EACH ROW]
[WHEN (…)]
[DECLARE]
BEGIN
… исполняемые операторы
[EXCEPTION]
END имя_триггера;
```

События триггеров DDL

Событие	Описание
CREATE	Создание объекта базы данных
ALTER	Изменение объекта базы данных
DROP	Удаление объекта базы данных
ANALYZE	Анализ состояния объекта базы данных
ASSOCIATE STATISTICS	Связывание статистики с объектом базы данных
AUDIT	Включение средства аудита базы данных
NOAUDIT	Выключение средства аудита базы данных
COMMENT	Создание комментария для объекта базы данных
DDL	Наступление любого из выше перечисленных событий
DISASSOCIATE STATISTICS	Удаление статистики объекта базы данных
GRANT	Назначение прав
REVOKE	Отмена прав
RENAME	Переименование объекта базы данных
TRUNCATE	Очистка таблицы

Создание триггера DDL

```
CREATE [OR REPLACE] TRIGGER
  имя_триггера
{BEFORE | AFTER} {событие_DLL}
  ON {DATABASE | SCHEMA}
DECLARE
…
BEGIN
…
END имя_триггера;
```

Триггеры событий баз данных

Событие базы данных	Описание
STARTUP	Открытие базы данных
SHUTDOWN	Нормальное закрытие базы данных
SERVERERROR	Возникновение ошибки Oracle
LOGON	Запуск сеанса Oracle
LOGOFF	Нормальное завершение сеанса Oracle

```
CREATE [OR REPLACE] TRIGGER имя_триггера
{BEFORE | AFTER} {событие_базы_данных} ON {DATABASE |
SCHEMA}
DECLARE
…
BEGIN
…
END имя_триггера;
```

Триггеры INSTEAD OF

Предназначены для выполнения операций вставки, обновления и удаления элементов *представлений*.

```
CREATE [OR REPLACE TRIGGER]
  имя_триггера
INSTEAD OF операция ON
  имя_представления
[DECLARE]
BEGIN
…
END;
```