1. Дистрибутивы СУБД Oracle. Установка СУБД Oracle 12с на Windows. Global Database Name и SID.

Oracle — открытый [дистрибутив](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8%D0%B2), доступный и свободный для скачивания через сайт корпорации [Oracle](https://ru.wikipedia.org/wiki/Oracle) (после регистрации в Oracle Software Delivery Cloud). На сайте oracle.com выбрать Database 12c > Download > 2 файла для вашей ОС

Редакции поставки:

- Express Edition бесплатная

- Standard Edition One для дома и малого бизнеса

- Standard Edition малый и средний бизнес

**- Enterprise Edition промышленный уровень**

**Запуск Oracle Database 12c Installer**

**Database Configuration Assistant**

Создать и конфигурировать базу данных/установить ПО базы данных/обновить существующую базу

Desktop class/Server class

Single instance database installation

Typical install/advanced install

Username, password – пользователь Windows или использовать существующего

Oracle base, Software location, Database file location, Global database name, Administrative password, Create as container database + pluggable database name

**SID (System ID) – уникальное имя, которым однозначно идентифицируется экземпляр/БД**

Скрытые утилиты – как правило это исполняемые файлы или пакеты PL/SQL, которые не задокументированы или документация по которым сложна для поиска. Например, некоторые PL/SQL пакеты которые никогда не загружаются инсталлятором Oracle, однако их описание присутствует в файлах операционной системы.

Основные директории, представляющие интерес:

$ORACLE\_HOME/bin - Содержит бинарные и исполняемые файлы, используемые сервером Oracle

$ORACLE\_HOME/rdbms/admin - Эта директория содержит много SQL скриптов используемых для создания PL/SQL пакетов и требуемого окружения

$ORACLE\_HOME/sqlplus/admin - Эта директория содержит скрипты, используемые автоматическим преобразованием и другими утилитами

ORACLE\_HOME – это путь к директории в файловой системе, где установлено ПО Oracle.

Как проверить установку?

**Сервисы и Список установленных программ**

**Редактор реестра regedit**

**Установщик Universal Installer**

**Пользователи и группы**

**Соединение с базой данных**

**Переменная окружения ORACLE\_HOME**

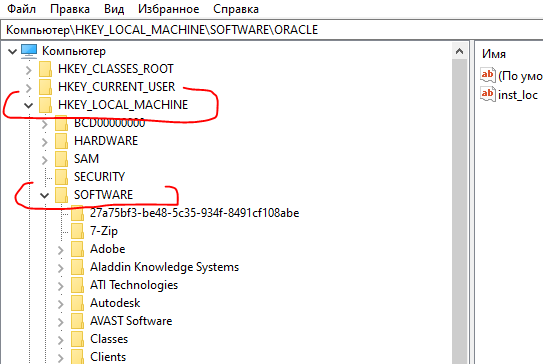
**Каждый экземпляр базы данных идентифицируется с помощью SID (System IDentifier – системный идентификатор).** Он хранится в переменной среды **ORACLE\_SID и используется утилитами и сетевыми компонентами для доступа к базе данных. Кроме понятия SID существует также и понятие SERVICE NAME, которые зачастую не различают.**

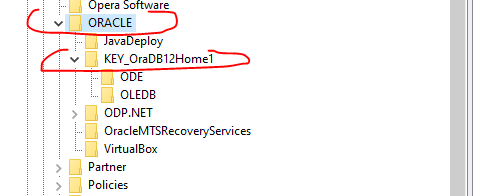
**SERVICE\_NAME определяет одно или ряд имен для подключения к одному экземпляру базы данных.** То есть можно указать несколько имен сервиса, ссылающихся на один экземпляр, с различными настройками.

Глобальное имя базы **данных однозначно отличает базу данных от любой другой базы данных в той же сети**. **Вы указываете глобальное имя базы данных при создании базы данных во время установки** или с помощью помощника по настройке базы данных Oracle.

***Oracle 12c имеет структуру клиент-сервер. Приложение базы данных включает клиентскую часть – это запуск приложений и взаимодействие с ним, а также серверную часть – это ПО Oracle, предоставляющее пользователю функции доступа к базе данных, это ПО, обслуживающее приложение.***

**ПРИ ИНСТАЛЛЯЦИИ СОЗДАЮТСЯ 2 табличных пространства: SYSTEM, SYSAUX**





Проверка установки через редактор реестра regedit

1. Основные системные пользователи. Основные специальные привилегии. Роль DBA.

При установке создаются 2 системных пользователя – **SYS и SYSTEM**

При установке Oracle Database автоматически создаются следующие учетные записи администраторов. **Они оба создаются с паролем, который вы указали при установке, и им обоим автоматически назначается роль DBA.**

**SYSTEM может выполнять все административные функции**, **кроме: резервное копирование и восстановление, обновление базы данных**

**SYS может выполнять все административные функции**. **Пользователю SYS предоставляются привилегия SYSDBA и SYSOPER, которые позволяют выполнять административные задачи высокого уровня**, такие как резервное копирование и восстановление

**DBA – предопределенная роль, которая автоматически создается для каждой базы данных и содержит все системные привилегии, кроме SYSDBA и SYSOPER**.

**SYSDBA и SYSOPER представляют собой привилегии для базовых задач администрирования**, запуска, остановки экземпляра, создания, удаления, монтирования базы (ассоциации с экземпляром), определения управляющих файлов. **Есть еще : SYSBACKUP, SYSDG, SYSKM, SYSASM**

**Полномочия «SYSOPER», а именно полномочия оператора базы данных, полномочия включают:**

Запустить сервер базы данных

Остановить сервер базы данных

Резервное копирование базы данных

Восстановление базы данных

Ограничение сеанса архивирования журналов

**Полномочия «SYSDBA», а именно полномочия администратора базы данных, включают:**

Запустить сервер базы данных

Остановить сервер базы данных

Резервное копирование базы данных

Восстановление базы данных

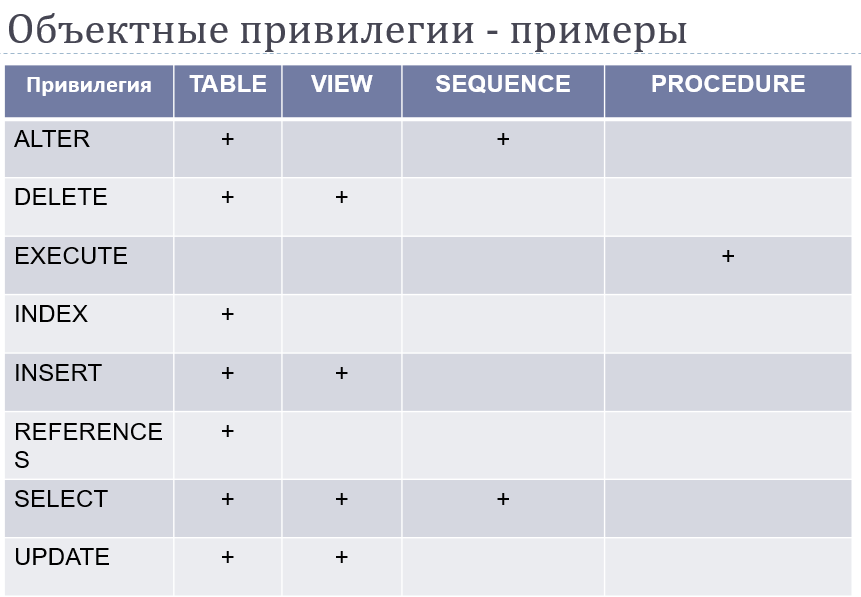
Ограничение сеанса архивирования журналов

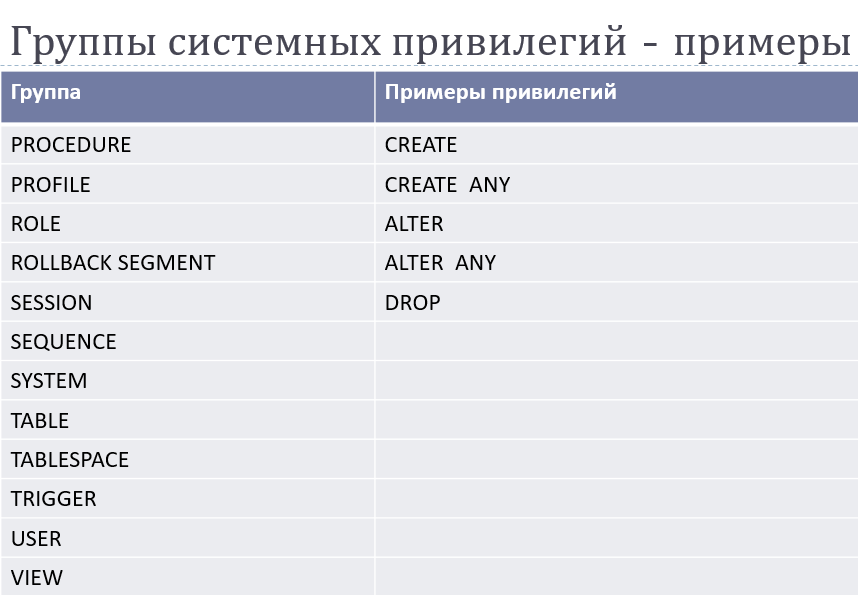
Функция управления

Создание базы данных

**Привилегия - это право выполнять конкретный тип предложений SQL, или право доступа к объекту другого пользователя**.

**ORACLE имеет два вида привилегий: системные и объектные**. Назначаются оператором **GRANT**. Отзываются оператором **REVOKE.**





1. Понятия базы данных и экземпляра базы данных.

Экземпляр сервера СУБД (instance) представляет собой три компоненты:

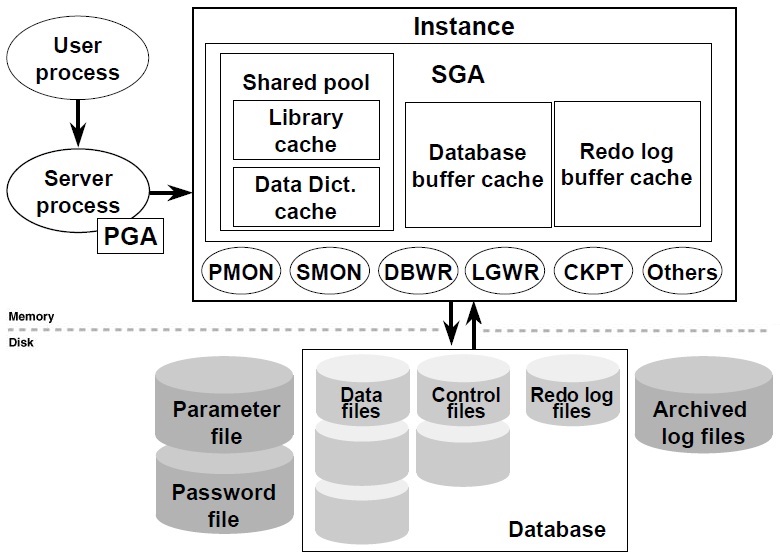
- **это запущенный сервер** (программа) СУБД Oracle

- **SGA (system global area), общая глобальная область памяти (оперативной памяти), которую разделяют все основные процессы запущенного экземпляра. Состоит из пулов. SGA используют все процессы – это и PMON, SMON, DBWR, CKPT, LGWR**

- **фоновые процессы для управления файлами базы данных**

**Каждый запущенный экземпляр активно использует ресурсы процессора и памяти.**

База данных представляет собой НАБОР ФИЗИЧЕСКИХ ФАЙЛОВ **ОПЕРАЦИОННОЙ СИСТЕМЫ (УПРАВЛЯЮЩИЕ ФАЙЛЫ, ФАЙЛЫ ДАННЫХ, ФАЙЛ ПАРОЛЕЙ, ВРЕМЕННЫЕ ФАЙЛЫ, ФАЙЛЫ ЖУРНАЛОВ ПОВТОРА), а ЭКЗЕМПЛЯР – ЭТО ПРОЦЕССЫ И ПАМЯТЬ.**



Компоненты архитектуры Oracle:

* **Файлы**, **образующие базу данных и поддерживающие экземпляр - файлы параметров, сообщений, данных, временных данных и журналов повторного выполнения**
* **Структуры памяти** - **системная глобальная область (System Global Area — SGA) и входящие в SGA пулы**
* **Физические процессы или потоки** - **серверные процессы, фоновые процессы и подчиненные процессы**

**Экземпляр**

**- Файлы параметров**

**База данных**

**- Файлы данных**

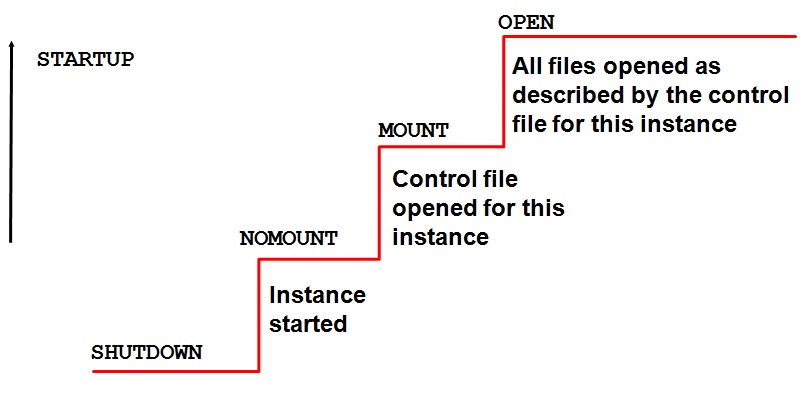
**- Файлы журнала повторного выполнения**

**- Управляющие файлы**

**- Временные файлы**

**- Файлы паролей**

1. Запуск и останов экземпляра базы данных Oracle.



STARTUP nomount / STARTUP mount / STARTUP open

**STARTUP force – принудительный запуск, даже если база в несогласованном состоянии**

**STARTUP restrict – ограниченный режим для обслуживания**

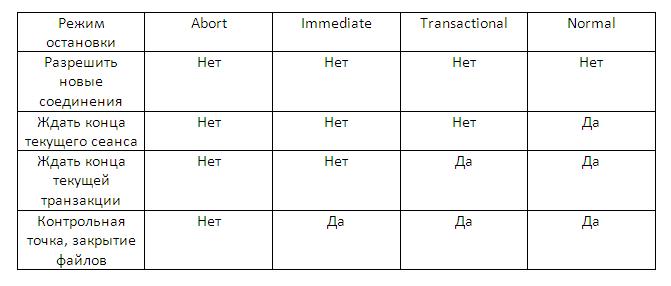
**SHUTDOWN – новые подключения никогда не разрешаются!**

**- abort – аварийное завершение в критических ситуациях**

- **normal – обычно, ждет завершения всего**, безопасно, ждет окончаний сеансов, ждет конца транзакций, ждет контрольных точек

- immediate – немедленно, сразу закрывает все, создает контрольные точки, откатывает незафиксированные транзакции

**- transactional – дожидается завершения текущих транзакций и контрольных точек, не дает открывать новые транзакции**



NOMOUNT

- запуск экземпляра

**- выбор состояния, в которое перейдет экземпляр**

MOUNT

- открыты контрольные файлы для экземпляра

**- база данных ассоциируется с экземпляром**

**- читается файл параметров, инициализируются фоновые процессы**

**- база данных пока не запущена**

**- пользователи пока не подключаются**

**- определяются управляющие файлы**

OPEN

**- все файлы открыты**

**- база данных запущена и открыта**

**- все связывается с дисковыми структурами**

**- доступно подключение пользователей**

1. Словарь базы данных: назначение, применение, основные представления.

Словарь данных – набор таблиц. Пользователи просматривают содержимое словаря данных с помощью представлений. Это сердце СУБД. **Это набор таблиц и СВЯЗАННЫХ С НИМИ ПРЕДСТАВЛЕНИЙ, которые предоставляют возможность отследить внутреннюю структуру базы данных и деятельность СУБД Oracle.**

**Таблицы словаря данных описывают всю базу данных: ее логическую и физическую структуру, использование свободного пространства, ее объекты с их ограничениями, а также информацию о пользователях.**

- Создается при генерации базы данных

- Обновляется и обслуживается сервером Oracle в фоновом режиме

**- Позволяет запрашивать данные в виде представлений**

**- Привилегия grant select any dictionary**

**- Расположены в табличном пространстве SYSTEM**

**- Владелец SYS, некоторыми владеет SYSTEM**

Внутренние таблицы можно разделить на два больших типа: **таблицы словаря со статическими данными и динамические таблицы производительности**. **Вам не придется обращаться к этим таблицам напрямую, взамен вы будете обращаться к хранимой в них информации через представления, основанные на них**

**Обращение к словарю происходит путем обращения к его представлениям. Они служат только для чтения.**

**USER** объекты, которые **принадлежат** пользователю

**ALL** объекты, **к которым пользователь имеет доступ**

**DBA** **все объекты базы данных для администраторов**

**V$ производительность сервера**

select \* from dictionary;

select \* from v$session; select \* from v$parameter; select \* from v$instance;

select \* from dba\_tablespaces;

select \* from dba\_users; select \* from dba\_profiles; select \* from dba\_roles;

select \* from dba\_(role | sys | tab | col)\_privs;

**Таблицы пользователя**

- **Набор таблиц, созданных и обслуживаемых пользователем**

**- Содержат информацию пользователя**

**Словарь данных**

**- Набор таблиц, созданных и обслуживаемых сервером Oracle**

**- Содержат информацию о базе данных**

1. Мультиарендная архитектура Oracle Multitenant.

**Oracle Multitenant - технология, позволяющая запустить несколько независимых баз данных в рамках одного экземпляра. Это опция новой архитектуры, в которой одна контейнерная база данных (CDB) может содержать много подключаемых баз данных (PDB).**

**Это запуск нескольких независимых баз данных в рамках одного экземпляра.** Эти базы данных изолированы друг от друга и не конфликтуют между собой.

**Каждая база данных имеет свой набор табличных пространств и набор схем, но при этом у них общая SGA и один набор серверных процессов.**

Словарь разбивается на две части: общую часть и локальную.

CDB – container database (контейнер баз данных)

PDB – pluggable database (подключаемая база данных)



* **Можно создавать несколько CDB – для разных версий программного обеспечения СУБД.**
* Одну и ту же PDB можно **переносить** между CDB.
* **В CDB создается главный контейнер Root. Root содержит метаданные CDB.**
* **В одной CDB можно создать до 252 PDB**.

Зачем? Многочисленные PDB совместно используют память (SGA) и процессы, благодаря этому на одной платформе можно использовать НАМНОГО БОЛЬШЕ PDB, чем если разворачивать это все как разные самостоятельные базы данных. **Это дает управление многими PDB как одной**.

**Root компонент – содержит служебные метаданные, это тоже контейнер**

**PDB$SEED – шаблон для создания PDB (не входит в число 252), CDB создается с этим шаблоном внутри**

Локальный пользователь: создается только в PDB, администрирует только свою PDB

Общий пользователь: создается в CDB$ROOT, виден в каждой PDB, **подключается к тем PDB, где у него есть привилегия grant create session to user container=all. Его имя начинается с C##. Может администрировать CDB, если есть привилегия DBA**

1. Файлы экземпляра Oracle. Файл параметров, управляющие файлы, файлы паролей, файлы трассировки.

**1) Файл параметров (.ora)** – хранит параметры инициализации экземпляра, например, количество сессий, размер SGA, количество процессов, курсоров.

**SPFILE – бинарный файл параметров**

**PFILE – текстовый файл параметров, используется, если не найден бинарный**

create **pfile** = ‘\_\_\_\_\_\_\_‘ from spfile; создать текстовый файл параметров из бинарного

create **spfile** = ‘\_\_\_\_\_\_\_’ from spfile; создать бинарный файл параметров из текстового – **ВСЕ ЭТО ПОД ПРИВИЛЕГИЕЙ SYSDBA**

show parameter имя\_параметра ИЛИ show parameter;

select \* from v$parameter;

**show parameter spfile; - расположение файла параметров**

* ***Может изменяться командой ALTER SYSTEM SET параметр=значение SCOPE=SPFILE***
* **Изменение параметров сохраняется в SPFILE и будет применяться при следующем старте Oracle**

Порядок поиска имен:

- spfileORACLE\_SID.ora

- spfile.ora

- initORACLE\_SID.ora

**2) Управляющий файл (.ctl)** – поскольку база данных является физическим набором связанных файлов данных, для их контроля и синхронизации требуются особые управляющие файлы (control files). Они отвечают за:

- управление состоянием базы данных

- **он один**, **но имеются копии** (для надежности лучше 2 на разных дисковых носителях)

- управляющий файл и его копии обновляются вместе

- содержит имена и расположение файлов данных, файлов журналов повтора, номера системных изменений SCN

**- ИСПОЛЬЗУЕТСЯ ДЛЯ ПОИСКА ДРУГИХ ФАЙЛОВ ОПЕРАЦИОННОЙ СИСТЕМЫ**

**- ПРИ ЗАПУСКЕ ЭКЗЕМПЛЯР ORACLE его читает и идентифицирует все файлы базы**

Местоположение управляющих файлов экземпляр получает из файла параметров

Как изменить? Shutdown immediate – скопировать управляющий файл – изменить параметр CONTROL\_FILES в файле параметров – startup open

**show parameter control;**

**select \* from v$controlfile; - расположение управляющего файла**

**alter database backup controlfile to trace;**

3) **Файл паролей - для аутентификации администраторов базы данных. Можно создавать, пересоздавать, изменять.**

ORACLE\_HOME\database

**select \* from v$pwfile\_users;**

4) **Файл трассировки (сообщений, log.xml)** – фиксирует все события, происходящее на сервере, в том числе и критические. По их содержимому можно выяснить причину сбоя.

**- Протоколы работы**

**- Трассировки – перехват событий и данных диагностики**

**- Дампы – файлы с полным содержимым базы данных в момент его создания**

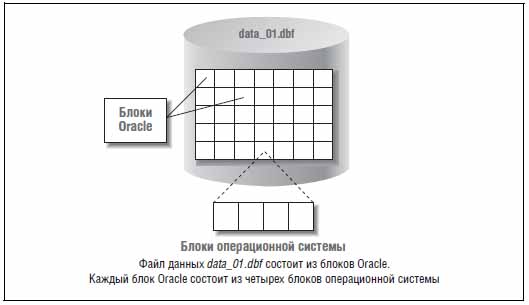
**- Записи с дополнительной информацией**

**select \* from v$diag\_info;**

1. Файлы базы данных Oracle. Файлы данных, журналы, архивы.

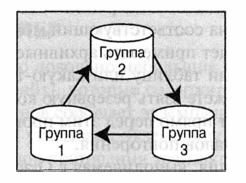
Файлы данных (.dbf) – файлы, в которых хранятся все данные базы данных. В каждой базе данных такой файл хотя бы один.

В **файлах данных** находятся собственно данные, хранящиеся в **базе данных Oracle**: [**таблицы**](https://oracle-patches.com/oracle/prof/3002-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8B-oracle)**и**[**индексы**](https://oracle-patches.com/oracle/prof/3013-%D0%B8%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D1%8B-oracle)**,**[***словарь данных***](https://oracle-patches.com/oracle/begin/2963-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-oracle-%D0%B8-%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8)**, в котором сохраняется информа­ция об этих структурах, и сегменты отката, необходимые для реализа­ции конкурентного доступа.**



**Файл данных состоит из блоков базы данных**, в свою очередь, **состоя­щих из дисковых блоков операционной системы. Размер блока в Ora­cle варьируется от 2 до 32 Кбайт**

**Журналы повтора или повторного выполнения (redo, .log) – дисковые ресурсы, в которых фиксируются изменения, вносимые пользователем в базу данных.** Журнал – файл операционной системы. Их минимум 2. **Применяются при восстановлении базы данных. ИСПОЛЬЗУЮТСЯ ЦИКЛИЧЕСКИ! Сначала запись в первый файл, после заполнения – во второй и так далее:**



select \* from v$**logfile**;

**Мультиплексирование журналов повтора** – поддержка нескольких копий каждого журнала

select \* from v$**log**;

**Для хранения на диске информации для повторного выполнения операций. Содержат:**

**- идентификатор начала транзакции**

**- наименование транзакции**

**- имя обновляемого объекта базы**

**- образ перед транзакцией**

**- образ после транзакции**

**- индикаторы фиксации**

**Журналы повтора – файлы повтора, объединенные в группы.**

SCN (system change number) – системный номер изменения в базе данных (select \* from v$log столбец FIRST\_CHANGE#)

Параметры журналов повтора (указаны в CONTROLFILE):

\***maxlogfiles** – максимальное количество групп журналов повтора

\***maxlogmembers** – максимальное количество файлов в группе

\***maxinstances**

\***maxdatafiles** – максимальное количество файлов данных

**Добавить новой группы:**

**alter database add logfile group 5 ‘путь\redo5.log’ size 50m blocksize 512;**

Добавление файла в группу:

**alter database add logfile member ‘путь\redo5-1.log’ to group 5**

**Переключение журналов:**

**alter system switch logfile;**

STATUS – current, active, inactive

Архивы журналов повтора:

**Режимы работы экземпляра (select name, log\_mode from v$database;)**

* + **ARCHIVELOG**
  + **NOARCHIVELOG**

Жизненно важны при восстановлении базы данных

Включить процесс архивирования в SQL Plus

**shutdown immediate**

**startup mount**

**alter database archivelog**

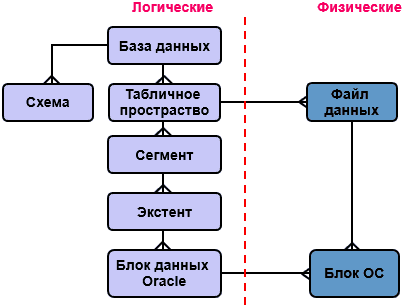
**alter database open**

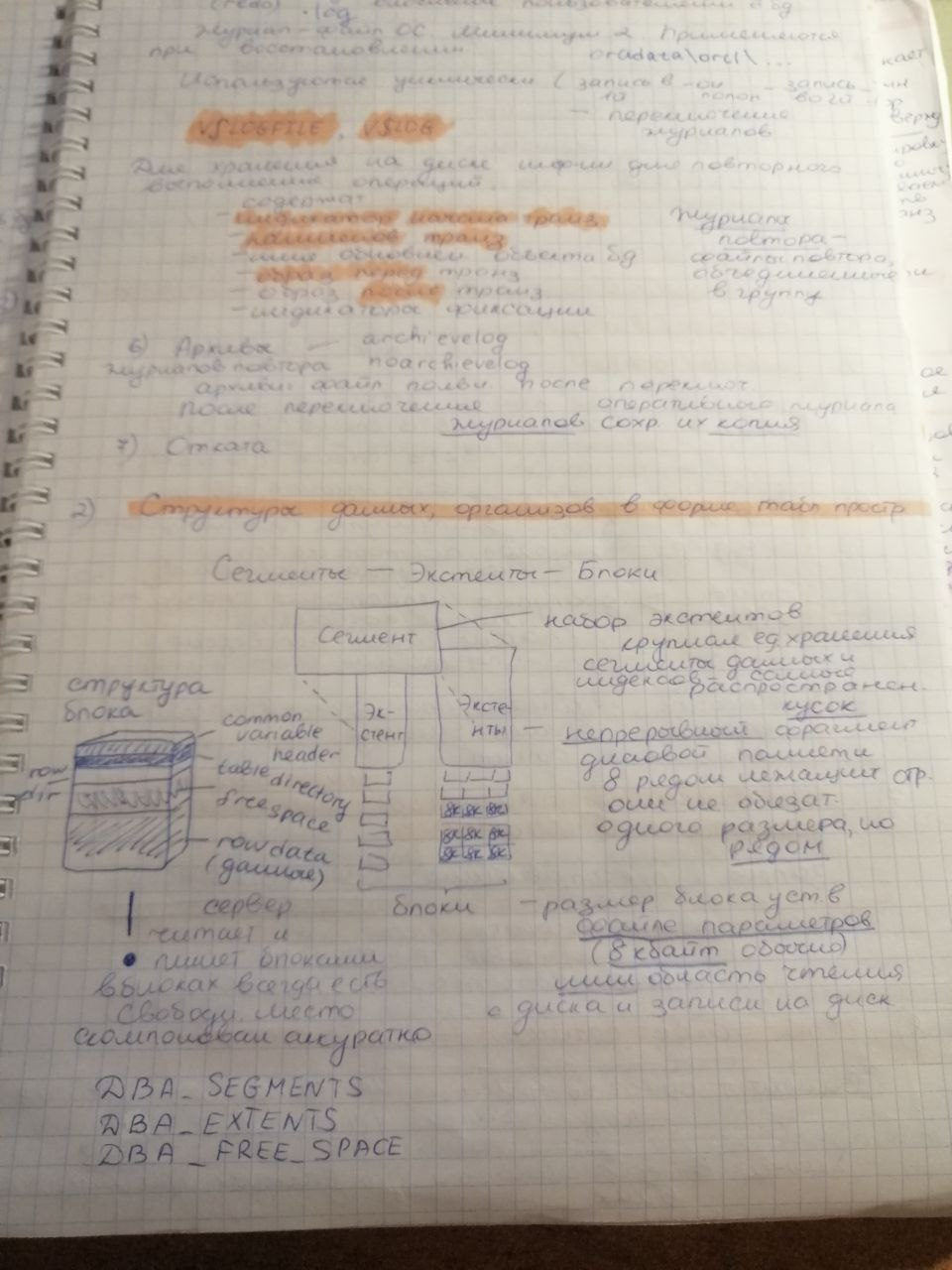
Архивный журнал появляется после переключения оперативного журнала

**Проверка состояния архивирования: select \* from v$database;**

Отключить: shutdown immediate -> **startup mount -> alter database noarchivelog**

1. Абстрактная модель Oracle. Логическая структура внешней памяти.





**СЕГМЕНТЫ – ЭКСТЕНТЫ – БЛОКИ**

В Oracle используется двухуровневая организация базы данных: **объекты верхнего уровня, или логические структуры базы данных**, и **объекты нижнего уровня, или физические структуры базы данных**

**Логические структуры – это табличное пространство, сегмент, экстент, блок данных Oracle**

**Физические структуры – файл операционной системы, блок операционной системы**

1. **Табличное пространство – единица хранения базы данных, эквивалент файловых групп MS SQL, логическая структура хранения и контейнер сегментов. С ТАБЛИЧНЫМ ПРОСТРАНСТВОМ СВЯЗАНО МНОГО ФАЙЛОВ, С ОДНИМ ФАЙЛОМ – ТОЛЬКО ОДНО ТАБЛИЧНОЕ ПРОСТРАНСТВО**

Datafile ‘ ‘ / Tempfile ‘ ‘, файл с расширением .dbf

Size

Autoextend on next

Maxsize

Extent management local – *размер экстентов определяется системой, локальное управление – значит табличное пространство имеет изображение блоков, экстентов, это позволяет отслеживать распределение экстентов без обращений к словарям*

Online / offline

Logging / nologging

Smallfile / bigfile

**PERMANENT - хранение постоянных объектов базы данных**

**TEMPORARY - хранение временных данных**

**UNDO - хранение сегментов отката, используется всегда один**

1. **Сегмент – это набор экстентов, крупная единица хранения.** Сегменты данных и индексов – самые распространенные. Область на диске, выделяемая под объекты. В **ОДНОМ ТАБЛИЧНОМ ПРОСТРАНСТВЕ МОЖЕТ БЫТЬ МНОГО СЕГМЕНТОВ!**

**Сегменты типизируются в зависимости от типа данных, хранящихся в них – сегменты таблиц, сегменты индексов, сегменты кластеров (всего 10 типов)**

**ТАБЛИЦЫ, ИНДЕКСЫ, КЛАСТЕРЫ, LOBINDEX, LOBSEGMENT, ROLLBACK, TYPE2 UNDO, NESTED TABLE**

*Управление размерностью сегментов может осуществляться автоматически (segment space management auto) или вручную (segment space management manual)*

***Сразу после создания таблицы сегмент не создается***

***Deferred segment creation – отложенное создание (show parameter segment)***

**Сегмент хранит только данные, поэтому он создается только при добавлении данных (в примере строки в таблицу)**

**При удалении строк (delete) из таблицы, сегмент не удаляется**

**Когда мы удаляем таблицу drop, имя сегмента изменяется, информация об удалении записывается в словарь. DBA\_SEGMENTS столбец segment\_name изменится имя на BIN$. Появится запись в USER\_RECYCLEBIN**

flashback table to before drop;

USER\_RECYCLEBIN отображает информацию о корзине текущего пользователя, механизм восстановления. Таблицы не удаляются полностью, они попадают в корзину. Вместо того, чтобы удалить таблицу, Oracle переименовывает ее и связанные с ней объекты дает системное имя BIN$... и помещает в корзину.

Для удаления recyclebin-сегмента применяется команда purge

drop table xxxx purge

или

drop table xxxx

purge table xxxx – теперь восстановление невозможно

1. **Экстент – НЕПРЕРЫВНЫЙ фрагмент дисковой памяти, 8 РЯДОМЛЕЖАЩИХ страниц**, они не обязательно одного размера, **ГЛАВНОЕ – РЯДОМ! Размер – свойство сервера. Опция UNIFORM SIZE \_\_\_ для одинаковой длины экстента при создании ТП**
2. **Блок – МИНИМАЛЬНАЯ единица чтения и записи, сервер пишет и читает блоками, в блоках всегда есть свободное место, скомпонован аккуратно. Наименьшая единица ввода-вывода**

*Если при создании табличного пространства задана опция UNIFORM, то все экстенты будут иметь одинаковую длину.*

Запрос к представлениям **dba\_extents/user\_extents, столбец BYTES**

**Блок – минимальная область чтения и записи с диска на диск. Основа иерархии хранения. Минимальный размер 8 Кб (кратен размеру блока ОС).** Сервер читает и пишет блоками. Когда делаем select, данные читаются из файлов базы в единицах блоков.

Размер блока устанавливается в файле параметров экземпляра init.ora: параметр db\_block\_size (посмотреть - show parameter db\_block\_size). Общепринято 8 Кб. Размер блока кратен размеру блока ОС.

Блок можно разделить на две части: **часть строк данных (row data) и часть свободного пространства (free space). Раздел строк данных содержит данные. Раздел свободного пространства - место, оставшееся в блоке для вставки новых данных или расширения существующих строк**. Дополнительно есть области ***- пространство заголовков (информация о table directory и common variable)***

* Размер кратен 2К, и должен быть кратен величине блока операционной системы (2К, 4К, 8К, допустимы 16К, 32К).

**В ОДНОМ ТАБЛИЧНОМ ПРОСТРАНСТВЕ РАЗМЕРЫ ВСЕХ БЛОКОВ ОДИНАКОВЫ!**

1. Абстрактная модель Oracle. Физическая структура внешней памяти.

Физически база данных организована как совокупность файлов, создаваемых обычными средствами операционной системы. **Таким образом, основой физической структуры внешней памяти является файл операционной системы, состоящих из блоков операционной системы.**

2 группы компонентов физического уровня:

- Системные объекты: файл параметров, управляющий файл, файл трассировки, журналы повтора, архивы

- Объекты пользователя – файлы данных

**Физические структуры базы данных Oracle (Oracle Database) — это действительные файлы данных Oracle на уровне операционной системы**. База данных Oracle состоит из следующих **трех основных типов файлов.**

* [**Файлы данных**](https://oracle-patches.com/oracle/begin/1272-%D1%84%D0%B0%D0%B9%D0%BB%D1%8B-%D0%B1%D0%B0%D0%B7%D1%8B-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-oracle)**. Эти файлы хранят данные таблиц и индексов. Один или более файлов составляют логическую сущность – табличное пространство.**
* [**Управляющие файлы**](https://oracle-patches.com/oracle/prof/641-%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D1%8E%D1%89%D0%B8%D0%B5-%D1%84%D0%B0%D0%B9%D0%BB%D1%8B-oracle)**. Эти файлы записывают изменения всех структур баз данных.**
* [**Файлы журналов повторного выполнения**](https://oracle-patches.com/oracle/prof/642-%D1%84%D0%B0%D0%B9%D0%BB%D1%8B-%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%BE%D0%B2-%D0%BF%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%B5%D0%BD%D0%B8%D1%8F-%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D0%B9-oracle)**. Эти онлайновые файлы содержат изменения, проведенные в табличных данных.**

Когда экземпляр базы данных нуждается в чтении данных таблицы или индекса, ***он читает их из файлов данных на диске, если только нужные данные уже не кэшированы в памяти Oracle.***

1. Абстрактная модель Oracle. Структура SGA.

**SGA – System Global Area – область данных, которая нужна всем, это часть оперативной памяти, которую используют все процессы и которую разделяют все процессы одного экземпляра базы данных.** SGA содержит всю необходимую информацию для операций 1 экземпляра. **Совместно используется всеми СЕРВЕРНЫМИ И ФОНОВЫМИ процессами.**

**SGA состоит из: 8 элементов**

- **Разделяемый пул SHARED POOL** – **кэш проверенных SQL-выражений, кэш словарей, данных, таблиц, представлений и триггеров, кэш словарей и библиотечный кэш. УПРАВЛЯЮЩИЕ СТРУКТУРЫ**. Разделяемая SQL область и PL\SQL-область, планы выполнения, операторы.

- **Большой пул LARGE POOL – хранит большие объекты памяти.** Здесь хранятся данные при резервном копировании. Не подчиняется алгоритму LRU

**- JAVA POOL** – позволяет работать Java-машине

**- Фиксированная область SGA – НЕ ДОПУСКАЕТ ИЗМЕНЕНИЕ РАЗМЕРА! Хранит значения параметров, указатели на другие области памяти, ЗАГРУЗОЧНЫЙ БИНАРНЫЙ КОД.** Размер зависит от ОС и платформы

**- Буферный пул (кэш) – содержит образы блоков, считанных из файлов или созданных динамически, чтобы обеспечить согласованное чтение.**

- Буферные пулы - **Keep** (постоянно хранит блоки данных), **default** (по умолчанию, если ничего не назначено), **recycle** (удаляет данные из кэша после использования). **Служат для минимизации промахов при обращении к кэшу**

**- Кэширование – это хранение поблизости копий любы объектов, которые возможно будут использоваться повторно, чтобы быстро получать к ним доступ**

*- Пользовательский процесс требует данные*

*- Данные ищутся в буферном пуле*

*- Если обнаружены, можно считать их из памяти*

*- Если нету – скопировать блок данных из файла на диске в буфер*

**- Input/Output pool –** пул ввода-вывода

**- NULL POOL – свободное место,** ничем не занято

**- Буфер журналов повтора – redo log buffer, ускоряет работу сервера, работает циклически**

Память пулам выделяется **гранулами (блоками).** Это наименьшая единица выделения памяти. **4, 8 или 16 Мбайт**. Можно установить разную гранулу разным пулам

С каждым блоком связан счетчик использований. Списки блоков упорядочены по числу обращений к ним.

Грязные блоки: данные в блоке и на диске различны. Отличаются от своего образа на диске и должны быть записаны в табличное пространство

Чистые блоки: когда данные одинаковы и в блоке, и на диске

Запись грязных блоков на диск:

- по таймауту 3 секунды

- контрольная точка checkpoint

- превышение длины грязного блока заданного лимита (заполнение на 1/3)

- процесс не может найти свободный блок

*(LRU) Least recently used – НАИМЕНЕЕ ЧАСТО ИСПОЛЬЗУЕМЫЙ - алгоритм - первыми вытесняются блоки с наименьшим значением счётчика использований (те, что используются реже всего). Счетчик TCH – touch count*. Алгоритм определяет, какой буфер затереть, если потребуется прочитать в память новый блок. Алгоритм вступает в силу в момент принятия решения о выборе буфера для нового содержимого

|  |  |
| --- | --- |
| LRU (Least Recently Used) | MRU (Most recently used) |
| Отбрасывает в первую очередь наименее используемые элементы  Отслеживает, что и когда использовалось  Список LRU – набор буферов  https://blog.skillfactory.ru/wp-content/uploads/2022/04/lru1-7579061.png | Чем старше элемент, тем больше вероятность следующего обращения к нему  Новый элемент заменит тот, который совсем недавно использовался. А те, которые старше по возрасту, остаются |

**select \* from v$sga;**

**select COMPONENT, CURRENT\_SIZE from v$sga\_dynamic\_components** where current\_size > 0;

Общий размер области SGA:

select sum(VALUE) as "Общий размер области SGA" from V$SGA;

**Размеры всех пулов изменяются через файл параметров (кроме фиксированной области SGA)**

SGA\_MAX\_SIZE и SGA\_TARGET\_SIZE

В каком операторе можно указать пул буферного кэша, который будет применяться для кэширования таблицы?

**storage**

**create table xxx (…) storage (buffer\_pool recycle) tablespace users;**

**keep**

**default**

Области оперативной памяти: SGA и PGA

**PGA (Programming Global Area) – создаваемая Oracle неразделяемая память при запуске серверного или фонового процесса.**

**Области памяти, которые содержат данные и управляющую информацию для серверных и фоновых процессов. Доступ монополен для серверного процесса.**

**Пишет в PGA только Oracle – стековое пространство и UGA (курсоры, данные сеанса, рабочие области SQL)**

1. Абстрактная модель Oracle. Серверные процессы Oracle.

**Процесс – механизм операционной системы Windows, осуществляющий запуск и выполнение приложений**. Процесс создается, когда запускается приложение. В общем случае выполняется в собственной области памяти.

**Поток – индивидуальная ветвь внутри процесса, выполняет конкретные программные инструкции.**

Процессы Oracle разделяются на **серверные, фоновые и подчиненные**.

**Серверные**

**- выполняются на основании запроса клиента**

**- с серверным процессом взаимодействует клиентский процесс**

- получают SQL операторы и выполняет их

- читают файлы данных

- ищут в кэше, отвечают на запросы

**Клиент – любая программа, пытающаяся подключиться к СУБД**

**select \* from xxxx**

**- синтаксический разбор**

**- поиск в разделяемом пуле SGA**

**- создание плана запроса**

**- при необходимости ищут данные в буферном кэше или запрашивают с диска**

**Серверные процессы Oracle** **запускаются из операционной системы и выполняют все операции с базой данных, такие как вставка и удаление данных.** Этот процесс Oracle, вместе со структурами памяти, выделенными Oracle операционной системой, **формируют работающий экземпляр Oracle**. **Это набор обязательных процессов Oracle, которые должны быть запущены, чтобы база данных вообще могла работать.**

Процессы Oracle делятся на **два основных типа** — для эффективности и для отделения клиентских процессов от задач [сервера базы данных](https://oracle-patches.com/oracle/begin/275-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80-oracle).

**Серверный процесс** — это **процесс, который обслуживает индивидуальный пользовательский процесс. Каждый пользователь, подключенный к базе данных, имеет свой отдельный серверный процесс**, существующий на протяжении существования сеанса.

**Серверный процесс создается для обслуживания пользовательского процесса и используется этим пользовательским процессом для взаимодействия с сервером Oracle**. Так, например, **когда пользователь посылает запрос на выборку данных, серверный процесс, созданный для обслуживания пользовательского приложения, проверяет синтаксис кода и выполняет код SQL**. Затем **он читает данные из файлов в блоки памяти. (Если другой пользователь захочет прочесть те же данные, то его пользовательский процесс прочтет их не с диска, а из памяти Oracle, где данные обычно находятся некоторое время.) И, наконец, серверный процесс вернет запрошенные данные пользовательскому процессу.**

***Режим выделенного сервера – (dedicated server, по умолчанию) – каждому клиенту выделяется отдельный серверный процесс****, теневой серверный процесс*

***Режим разделяемого сервера – работа клиента через диспетчера, или Dispatcher.*** *Он получает запрос клиента, помещает в очередь к разделяемым серверам. Незанятый сервер обрабатывает запрос. Разделяемый сервер помещает результат обработки во входную очередь. Диспетчер извлекает результат и посылает клиенту.*

**Oracle Net Listener – процесс на стороне сервера, который прослушивает входящие запросы на соединение с экземпляром**. Listener – программа-сервер, которая прослушивает TCP-порт, принимая запросы на соединение.

**В результате успешной работы Listener устанавливает соединение между клиентом и обработчиком запросов экземпляра. По умолчанию TCP-порт – 1521.**

1. Абстрактная модель Oracle. Фоновые процессы Oracle.

**Процесс – механизм операционной системы Windows, осуществляющий запуск и выполнение приложений**. **Процесс создается, когда запускается приложение. В общем случае выполняется в собственной области памяти**.

**Поток – индивидуальная ветвь внутри процесса, выполняет конкретные программные инструкции.**

Процессы Oracle разделяются на **серверные, фоновые и подчиненные**.

**Фоновые процессы**

**- запускаются вместе с базой данных**

- **выполняют ОБСЛУЖИВАНИЕ**

- **поддержка производительности сервера, поддержка работы большого числа пользователей на сервере**

**Фоновые процессы — это “рабочие лошадки” экземпляра Oracle**. Они позволяют большому числу пользователей параллельно и эффективно использовать информацию, хранимую в файлах данных. **Oracle создает эти процессы автоматически при запуске экземпляра,** и будучи постоянно связанными с операционной системой, **они избавляют программное обеспечение Oracle от многократного запуска множества отдельных процессов для выполнения разнообразных задач, которые нужно выполнять на сервере операционной системы.** **Каждый из фоновых процессов Oracle отвечает за отдельную задачу, тем самым повышая эффективность экземпляра базы данных**. Эти процессы автоматически создаются Oracle при запуске экземпляра базы данных, и прекращают свою работу при его останове.

1. LREG

**– периодически регистрирует сервисы в процессе listener (сервис – точка подключения к экземпляру)**

- listener registration

1. DBWn

**- записывает по LRU грязные блоки в файлы данных**

- их обычно несколько, n – число

- 1 на 8 процессоров, максимум их 12

**- проверяет наличие грязных блоков и инициирует checkpoint и по команде CKPT их сбрасывает на диск, тем самым обеспечивая согласованность данных**

- обычно каждые 3 секунды по таймауту

- Запишет, если нет пустых буферов, если прошел 3 секунды таймаут, если была контрольная точка, если слишком много грязных буферов. Сессии работают не с информацией на диске, а с кэшем. Он выбирает грязные блоки, которые давно не использовались (LRU)

3) CKPT

- checkpoint process

- инициирует сброс грязных блоков DBWriter-ом и сброс буфера в файл лога на диске LGWriter-ом

- обновляет заголовки всех файлов данных, фиксируя детали контрольных точек

- checkpoint – запись информации о контрольной точке в управляющий файл

**Контрольная точка создает известную хорошую точку, с которой ядро СУБД может начать применять изменения, содержащиеся в журнале, во время восстановления после неожиданного завершения работы или сбоя.**

4) LGWR

- **только один**

- **управляет буфером журналов повтора**

**- ЗАПИСЫВАЕТ ИНФОРМАЦИЮ ИЗ БУФЕРА ЛОГОВ В ФАЙЛ ЛОГА НА ДИСКЕ**

- **записывает блоки буфера журналов повтора в группы журналов повтора** каждые 3 секунды, при контрольной точке и при заполнении буфера на 1/3

- **писатель журнала пишет содержимое буфера журнала повторного выполнения в файлы онлайнового журнала повторного выполнения**

5) PMON

- process monitor

- очистка после ненормального закрытия подключений

- откат незавершенных транзакций, снимает блокировки, освобождает ресурсы

- СЛЕДИТ ЗА СЕССИЯМИ

- **следит за работой других ПРОЦЕССОВ, ВОССТАНАВЛИВАЕТ ИХ РАБОТУ ПРИ СБОЕ, выполняет очистку после остановленных и сбойных процессов**

6) SMON

- system process monitor

**- восстанавливает экземпляр, работает с сегментами табличного пространства, очищает их, восстанавливает транзакции, РАБОТАЕТ С ЭКЗЕМПЛЯРОМ**

**- выполняет восстановление ЭКЗЕМПЛЯРА после сбоев**

**7) RECO**

**- разрешает проблемы, связанные с распределенными транзакциями**

Одна из баз данных, к которой подключился первоначально клиент, становится координатором. Сервер опрашивает остальные сервера о готовности фиксировать транзакцию. RECO может восстановить или откатить такую транзакцию в случае сбоев, связавшись по сети с координатором

8) ARCn

- archiver process

- копирует файлы журналов повтора после переключения группы журналов

9) FBDA

- хранение ретроспективных данных

1. Процесс-слушатель Oracle и его основные параметры.

**Прослушиватель (Listener) – компонент, который позволяет устанавливать соединение между клиентом и базой данных.** **Один прослушивать может обслуживать неограниченное количество баз данных.**

**Oracle Net Listener – процесс на стороне сервера, который прослушивает входящие запросы на соединение с экземпляром**. Listener – программа-сервер, которая прослушивает TCP-порт, принимая запросы на соединение.

**В результате успешной работы Listener устанавливает соединение между клиентом и обработчиком запросов экземпляра. По умолчанию TCP-порт – 1521.**

Принцип установки соединения с сервером Oracle по сети:

Клиент устанавливает начальное соединение со слушателем, слушатель принимает и проверяет запрос на подключение клиента и передает его обработчику служб базы данных.

1. **Клиент выполняет запрос к Listener на соединение с сервисом экземпляра**
2. **Listener запрашивает соединение с сервером**
3. **Сервер возвращает параметры соединения обработчиком сервиса**
4. **Listener сообщает параметры соединения клиенту**
5. **Клиент соединяется с обработчиком запросов сервиса для дальнейшей работы в рамках соединения**

**Клиент 🡺 IP или hostname, TCP-порт, имя сервиса 🡺 Instance Oracle**

- Имена экземпляров (SID) к чему будем подключаться

- Имена служб

- Дескрипторы соединений

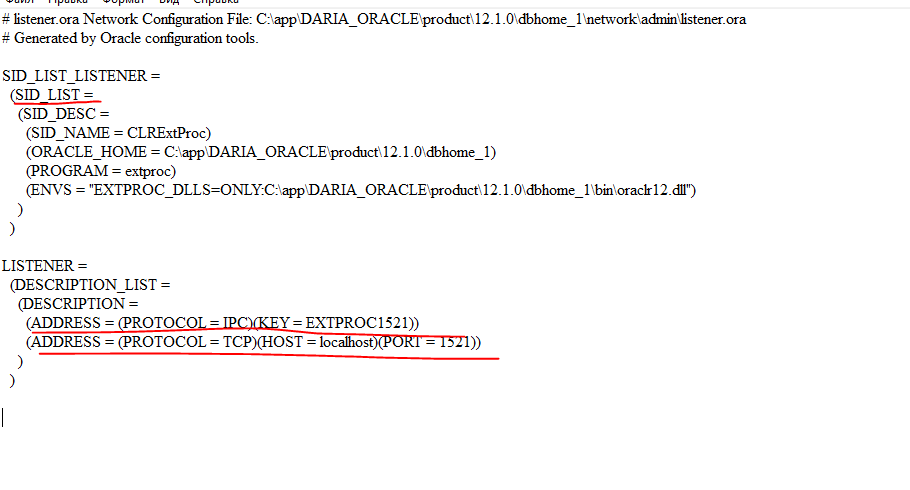
- Идентификаторы соединений

- Строки подключений

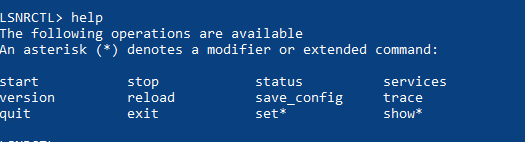
C:\app\DARIA\_ORACLE\**product\12.1.0\dbhome\_1\NETWORK\ADMIN**

**Listener.ora – конфигурация слушателя. Имя listener, протокол, адрес. ТАКОЙ ФАЙЛ КОНФИГУРИРУЕТСЯ ОДИН! Слушателей может быть много, и этот файл будет их всех обслуживать.**

**Sid\_list\_listener – адреса регистрируемых сервисов**



lsnrctl через cmd – утилита для управления listener



services можно посмотреть точки подключения

status

start, stop, reload, version

exit и quit выход, идентичны

1. Сетевые настройки Oracle. Установление соединения по сети.

***Режим выделенного сервера – (dedicated server, по умолчанию) – каждому клиенту выделяется отдельный серверный процесс****, теневой серверный процесс*

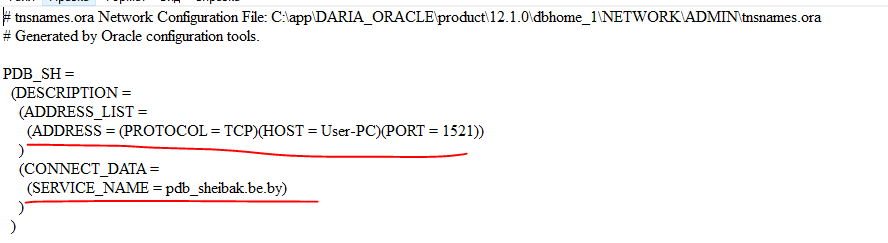
***Режим разделяемого сервера – работа клиента через диспетчера, или Dispatcher.*** *Он получает запрос клиента, помещает в очередь к разделяемым серверам. Незанятый сервер обрабатывает запрос. Разделяемый сервер помещает результат обработки во входную очередь. Диспетчер извлекает результат и посылает клиенту.*

**Sqlnet.ora - обычный текстовый файл конфигурации, который содержит информацию о том, как и сервер Oracle, и клиент Oracle должны использовать возможности Oracle Net для сетевого доступа к базе данных.** Параметры службы аутентификации, именования соединений

SQLNET.AUTHENTICATION\_SERVICES= (NTS)

NAMES.DIRECTORY\_PATH= (TNSNAMES, EZCONNECT)

**Tnsnames.ora – файл конфигурации, определяющий адреса баз данных для установки соединения.** Протоколы, порты, host (словом, адреса), connect\_data – service\_name или имя экземпляра. Определяются имена, которые используются для обращения к базе данных, эти имена используются Прослушивателем Listener.



TNSNAMES.ORA — это конфигурационный файл [SQL\*Net](https://www.orafaq.com/wiki/SQL*Net) , определяющий адреса баз данных для установления к ним соединений

**Listener.ora – конфигурация Прослушивателя Listener**

Виды подключений:

**Простое Basic**

- TCP/IP

- явно указываем все параметры соединения

- имя соединения, пользователь, пароль, имя хоста, порт, SID

Connect имя/пароль@хост:порт/имя\_службы

**Локальное TNS TRANSPARENT NETWORK SUBSTRATE**

- Имена служб и их дескрипторы подключения сохраняются в локальном файле tnsnames.ora

- Указываем имя пользователя и пароль, Oracle и sqlnet.ora

- Net обращается к tnsnames.ora – network alias

В cmd **tnsping алиас** – проверка доступности

Connect имя/пароль@алиас

**LDAP-соединение**

- метод наименование с помощью службы каталогов

- LDAP-сервер для преобразования служб

**Local/bequeath-соединение**

- соединение со стандартным сервисом

- только на сервере

- не указываем параметры соединения

- не действует listener

**Дескриптор соединения – это объединенная спецификация двух компонентов подключения к базе данных: ИМЯ СЛУЖБЫ БД service\_name И МЕСТОПОЛОЖЕНИЕ БД protocol, host, port**

1. Табличные пространства СУБД Oracle и их основные параметры.

**Табличное пространство – единица хранения базы данных, эквивалент файловых групп MS SQL, логическая структура хранения и контейнер сегментов. С ТАБЛИЧНЫМ ПРОСТРАНСТВОМ СВЯЗАНО МНОГО ФАЙЛОВ, С ОДНИМ ФАЙЛОМ – ТОЛЬКО ОДНО ТАБЛИЧНОЕ ПРОСТРАНСТВО**

Create (undo | temporary) tablespace

Datafile ‘ ‘ / (datafile ‘ ‘ | tempfile ‘ ‘) - файл с расширением .dbf

Size

Autoextend on next / autoextend off – авторасширение при достижении maxsize

Maxsize - указывает максимальное дисковое пространство, которое может быть выделено файлу данных

Extent management local – *размер экстентов определяется системой, локальное управление – значит табличное пространство имеет изображение блоков, экстентов, это позволяет отслеживать распределение экстентов без обращений к словарям*

Uniform *size* – экстенты будут иметь фиксированный размер size

Online / offline

Logging / nologging

Smallfile / bigfile

Reuse - Если у вас уже есть предварительно созданные файлы данных ОС, он будет использовать их, что позволит вам предварительно создавать большие файлы ОС

Единицы памяти обычно К (килобайт) или М (мегабайт)

Smallfile табличное пространство, содержащее 1022 файла данных или временных файлов

Bigfile один файл (размер этого файла может составлять до 4 миллионов блоков) 128 ТБ

**PERMANENT - хранение постоянных объектов базы данных, может быть несколько, ОДНО ПО УМОЛЧАНИЮ ЕСТЬ**

**TEMPORARY - хранение временных данных, может быть несколько**

**UNDO - хранение сегментов отката, используется всегда один, ОДИН!!!**

**Базы данных Oracle логически делятся на одно или более табличных пространств**. **Табличное пространство Oracle — это логическая сущность, содержащая физические базы данных**. Табличные пространства хранят все доступные данные базы, и эти данные в табличных пространствах физически хранятся в одном или более файле данных. Файлы данных - это сформированные Oracle файлы операционной системы. **Табличное пространство — просто логическая конструкция, являющаяся первичной логической структурой хранения в базе данных Oracle**

**Табличное пространство состоит из СЕГМЕНТОВ**, сегмент – из экстентов, экстент – из блоков.

**Следующие пять табличных пространств обычно представляют собой табличные пространства по умолчанию,** которые должны иметь все базы данных, даже несмотря на то, что в принципе можно создать базу данных включающую всего два из них:

* **табличное пространство System;**
* [**табличное пространство Sysaux**](https://oracle-patches.com/oracle/prof/2991-sysaux) **– вспомогательное для System**;
* [**табличное пространство Undo**](https://oracle-patches.com/oracle/begin/3024-%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-%D0%BE%D1%82%D0%BC%D0%B5%D0%BD%D1%8B-%D0%B4%D0%BB%D1%8F-%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F-%D1%81%D0%BE%D0%B3%D0%BB%D0%B0%D1%81%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D1%87%D1%82%D0%B5%D0%BD%D0%B8%D1%8F)**;**
* [**табличное пространство Temporary**](https://oracle-patches.com/oracle/prof/2987-%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%87%D0%BD%D1%8B%D0%B5-%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%B0)**;**
* [**постоянное табличное пространство по умолчанию**](https://oracle-patches.com/oracle/prof/2989-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%87%D0%BD%D1%8B%D0%B5-%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%B0-%D0%BF%D0%BE-%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E)**.**

select \* from dba\_tablespaces;

1. Роли и привилегии СУБД Oracle и их основные параметры.

**Роль – поименованный набор привилегий, создается привилегированным администратором.** Применяется для управления привилегиями пользователей. **Роль создается для того, чтобы логически сгруппировать разрешения для пользователей.**

Привилегия – право выполнять конкретный тип предложений SQL или право доступ к объекту другого пользователя.

grant/revoke

Для создания роли используется команда

**CREATE ROLE rolename**

**[idenitifed by 123];**

**Затем права назначаются для роли используя обычные синтаксис команд включая WITH ADMIN или WITH GRANT OPTION, если нобходимо.**

GRANT privileges [ON object] TO role\_name

REVOKE privileges [ON object] FROM role\_name;

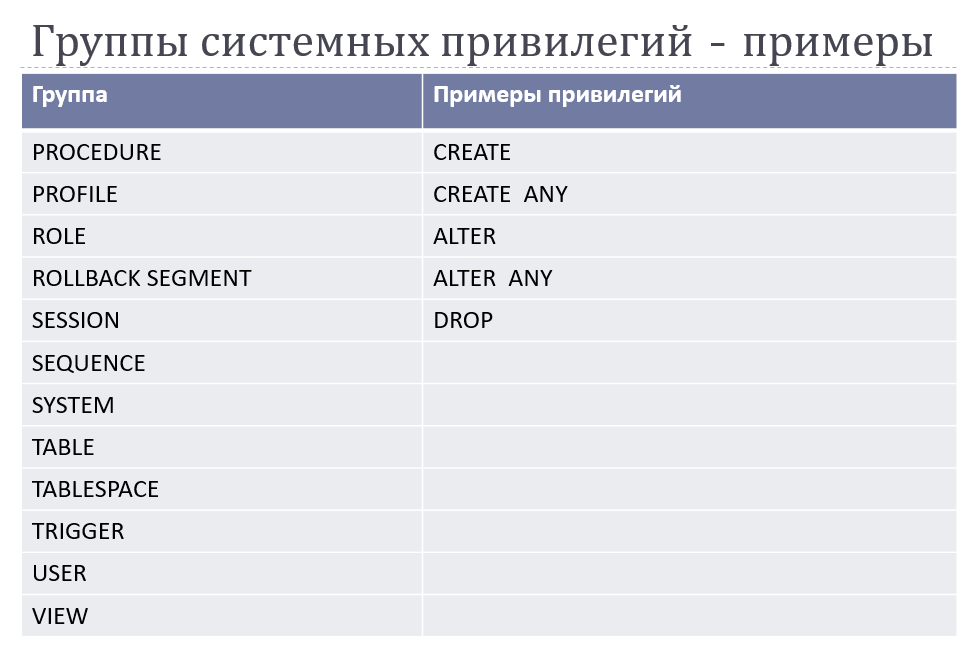
DROP ROLE role\_name;

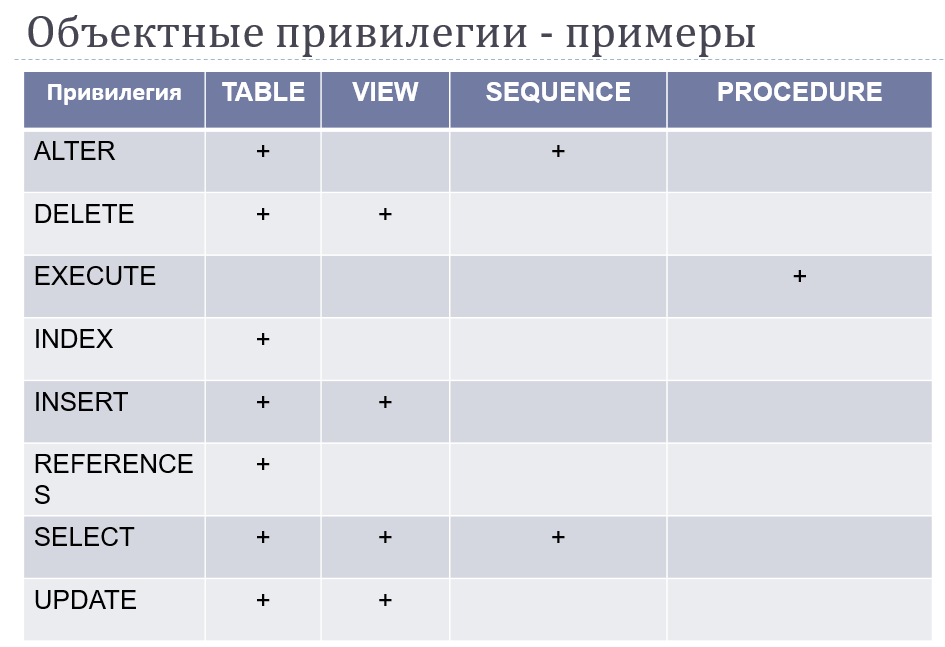
GRANT role\_name TO user\_name;

Системная привилегия – разрешение на выполнение определенных действий

Объектные – на изменение объекта

|  |  |
| --- | --- |
| **системные** | **объектные** |
| на изменение системы: create table / trigger / index / view / procedure … | на изменение объекта: select |
| *WITH ADMIN OPTION* – дают право пользователю также назначать или отзывать СИСТЕМНЫЕ ПРИВИЛЕГИИ:  **alter, analyze, audit, backup, create, drop, select (стараться редко использовать)**  any – для любого объекта в любой схеме; all – для всех объектов | *WITH GRANT OPTION* – дают право назначать или отзывать ОБЪЕКТНЫЕ ПРИВИЛЕГИИ: **alter, delete, execute, insert, update, select, references**  снимает привилегии тот, кто их назначал |
| **объекты грантов:**  database, user, profile, tablespace, role, table, index, trigger, procedure, sequence, view | **объекты грантов:**  table, view, sequence, procedure |





**Для подключения к базе данных создается учетная запись пользователя – имя и пароль.** ***Чтобы, используя учетную запись, пользователь мог подключиться к БД, а затем работать с ее объектами, пользователь (точнее, его учетная запись) должен обладать необходимыми привилегиями***. **Привилегии – это набор разрешений, регламентирующих возможности пользователя выполнять те или иные действия над объектами БД и данными, хранящимися в этих объектах.**

Например, для подключения к БД и создания в ней таблиц необходимо иметь привилегии CREATE SESSION (создание сессии) и CREATE TABLE (создание таблицы). Такого рода привилегии для каждого пользователя назначает администратор

В Oracle имеются два типа привилегий – **системные и объектные.** ***Системные привилегии регламентируют работу с объектами базы данных, например, создание, изменение или удаление таблиц****.*

***Объектные привилегии определяют возможности использования уже существующих объектов,* или, другими словами, доступ к объектам – например удаление данных из таблицы**

1. Пользователь СУБД Oracle и его основные параметры.

Основные аспекты управления безопасностью базы данных Oracle.

* **Управление доступом к данным (авторизация).**
* **Ограничение доступа легальных пользователей (аутентификация).**
* **Обеспечение подотчетности пользователей (аудит).**
* **Защита основных данных в базе данных (шифрование).**
* **Управление безопасностью всей организационной информационной инфраструктуры (безопасность предприятия).**

**create user xxxxx**

**identified by 123**

**default tablespace xxxx**

**quota xM / unlimited on xxxxx**

**temporary tablespace xxxx**

**profile xxxx**

**account unlock**

**password expire**

Чтобы пользователь мог подключиться к базе и начать обмен данными с ней, ему нужно предоставить системные полномочия *CREATE SESSION, или назначить ему роль, для которой уже выданы соответствующие привилегии*

**Если не желаете, чтобы пользователь вообще создавал какие-либо объекты в базе данных, не предоставляйте ему квоту ни в одном из табличных пространств**

ALTER USER xxxx QUOTA 100M ON users;

Как только новому пользователю предоставлена квота в табличном пространстве, он сможет создавать объекты базы данных, такие как таблицы и индексы. **По умолчанию любые объекты, созданные пользователем, будут помещаться в заданное по умолчанию постоянное табличное пространство пользователя (в рассматриваемом примере это *USERS*).** Однако пользователь может создавать объекты в любом табличном пространстве, если только обладает в нем квотой дискового пространства.

1. Профиль безопасности СУБД Oracle и его основные параметры.

Профиль – **коллекция атрибутов, связанных с использованием ресурсов и паролей, которая может быть назначена пользователю.** **Это наложение индивидуальных ограничений пользователю на использование ресурсов Oracle. Позволяет ограничить потребление ресурсов и количество сессий**. Профиль DEFAULT – профиль по умолчанию, если не назначен конкретный

select \* from dba\_profiles;

**create profile xxxx limit**

password\_life\_time 180 -- дней жизни пароля

sessions\_per\_user 3 -- количество сессий

failed\_login\_attempts 7 -- попыток входа

password\_lock\_time 1 -- дней блокировки после ошибки входа

password\_reuse\_time 10 -- через сколько дней повторить пароль

password\_grace\_time default -- количество дней предупреждений о смене пароля

connect\_time 180 -- минут соединения

idle\_time 30 -- минут простоя

**Профиль (profile) — это коллекция атрибутов, связанных с использованием ресурсов и паролей, которая может быть назначена пользователю**. **Несколько пользователей могут разделять один и тот же профиль, и в базе данных Oracle может существовать неограниченное количество профилей**. **Профили накладывают жесткие ограничения на потребление ресурсов различными пользователями в базе данных и помогают ограничивать число одновременно отрытых пользователем сеансов**, их продолжительность и использование ЦП и других ресурсов

1. Язык SQL. Основные операторы и их назначение.

SQL – структурированный язык запросов, Structured Query Language, **декларативный язык программирования (ОПИСЫВАЕТСЯ ОЖИДАЕМЫЙ РЕЗУЛЬТАТ, А НЕ СПОСОБ ЕГО ПОЛУЧЕНИЯ),** применяемый для создания, модификации и управления данными в реляционной базе данных.

**Реляционная база данных хранит информацию в табличной форме со строками и столбцами, представляющими различные атрибуты данных и различные связи между значениями данных**. **Инструкции SQL можно использовать для хранения, обновления, удаления, поиска и извлечения информации из базы данных.**

\* DDL (definition) – create, alter, drop

\* DML (manipulation) – select, insert, update, delete

\* DCL (control) – grant, revoke

\* TCL – commit, rollback, savepoint

**Data Definition Language (DDL)** – *это группа операторов определения данных. Другими словами, с помощью операторов, входящих в эту группы, мы определяем структуру базы данных и работаем с объектами этой базы*, т.е. создаем, изменяем и удаляем их. **CREATE, ALTER, DROP**

**Data Manipulation Language (DML)** – *это группа операторов для манипуляции данными. С помощью этих операторов мы можем добавлять, изменять, удалять и выгружать данные из базы,* т.е. манипулировать ими. **SELECT, INSERT, UPDATE, DELETE**

**Data Control Language (DCL)** – *группа операторов определения доступа к данным. Иными словами, это операторы для управления разрешениями, с помощью них мы можем разрешать или запрещать выполнение определенных операций* над объектами базы данных. **GRANT, REVOKE**

**Transaction Control Language (TCL)** – *группа операторов для управления транзакциями. Транзакция – это команда или блок команд (инструкций), которые успешно завершаются как единое целое, при этом в базе данных все внесенные изменения фиксируются на постоянной основе или отменяются, т.е. все изменения, внесенные любой командой, входящей в транзакцию, будут отменены*. **COMMIT, ROLLBACK, SAVEPOINT**

***Structured Query Language (SQL) — язык структурированных запросов, с помощью него пишутся специальные запросы (SQL инструкции) к базе данных с целью получения этих данных из базы и для манипулирования этими данными.***

1. Таблица и ее основные параметры.

Таблица – основная структура сохранения информации в БД

Типы таблиц:

* + **Традиционные таблицы (heap organized table) хранятся без определенного порядка по принципу КУЧИ, при добавлении данных – они идут в первое свободное место в сегменте**
  + **Индекс-таблицы (index organized table) на основе B-tree сбалансированного дерева поиска упорядоченных ключей** – индексная структура
  + **Кластеризованные индекс-таблицы (index clustered table)**
  + **Кластеризованные хэш-таблицы (hash clustered table)**
  + **Отсортированные кластеризованные хэш-таблицы (sorted hash clustered table)**

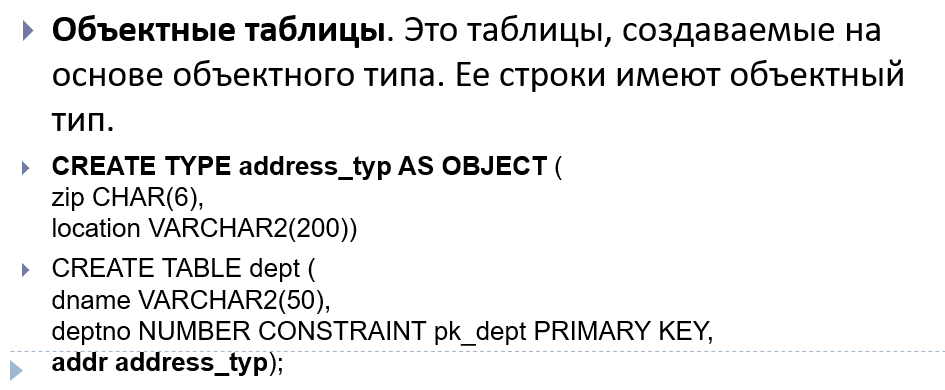
*- кластеризованные таблицы являются частью групп таблиц, которые разделяют между собой одни и те же блоки данных – часто запрашиваются вместе*

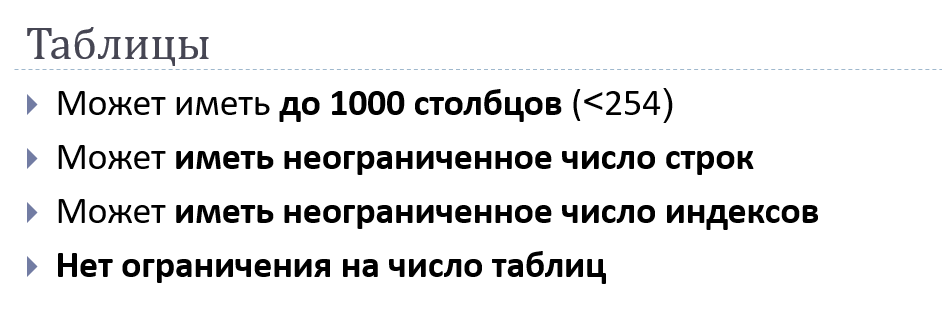
* + **Вложенные таблицы (nested table) – генерируемые и поддерживаемые системой дочерние таблицы**
  + **Временные таблицы (temporary table) – временные данные на время транзакции или сеанса при необходимости выделяются экстенты временного табличного пространства**
  + Объектные таблицы – создаются на основе объектного типа, строки такой таблицы могут иметь объектный тип
  + **Внешние таблицы – используют наборы данных из внешних файлов операционной системы (external tables)**
  + **Секционированные таблицы - позволяет делить большие объемы данных на подтаблицы, именуемые разделами (partition),** **согласно различным критериям**. Секционирование особенно полезно в среде хранилища данных

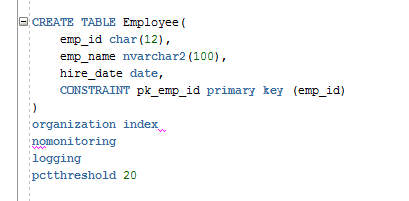
**Временные таблицы создаются и существуют, пока их не удалят** (не удалится, если использует другой пользователь) – **временные таблицы бывают ON COMMIT PRESERVE ROWS (на время сеанса) или ON COMMIT DELETE ROWS (на время транзакции)**

**Временные таблицы глобальны, создаются во временном табличном пространстве, после окончания сеанса (основаны на сессии) или транзакции (основаны на транзакции) таблица очищается, на временных таблицах можно определять ограничения, триггеры, индексы, но нельзя размещать в кластере.** Определены "статически" - создаются один раз для каждой базы данных, а не один раз для хранимой процедуры в базе данных. **Они всегда существуют, но кажутся пустыми, пока не поместим данные**. **Данные зависят от сеанса**. **То есть только мы можем видеть свои данные, даже если 100 пользователей используют одну и ту же таблицу**

**MS SQL:** После создания все временные таблицы сохраняются в таблице tempdb, название временной таблицы начинается со знака решетки # или ##, **# - локальная (на текущую сессию), # - глобальная (на все открытые сессии), удалятся либо, когда отключится пользователь, работающий с ней (если локально), и все пользователи - если глобально**. **Никто, кроме вас, не узнает, что ваша временная таблица существует**







**Create [*global temporary*] table xxxxx**

***[on commit delete rows | on commit preserve rows]***

***(column\_name type, …)***

**segment creation deffered [immediate]**

**organization index [heap | external]**

**cluster xxxx (xxxx)**

**tablespace xxxx**

**nomonitoring (сервер смотрит как заполняется таблица или не следит, слежка за активностью DML в ней)**

**logging**

**pctfree xxxx**

**pctused xxxx**

**PCTFREE – процент памяти блока, резервируемый для возможных обновлений строк.** Чем выше этот процент, тем больше места тратится впустую

**PCTUSED – процент занятой части памяти блока** (automatic segment space management / manual segment space management)

1. Временные таблицы.

**Временные таблицы (temporary table) – *временные данные на время транзакции или сеанса при необходимости выделяются экстенты временного табличного пространства***

**Временные таблицы создаются и существуют, пока их не удалят** (не удалится, если использует другой пользователь) – **временные таблицы бывают ON COMMIT PRESERVE ROWS (на время сеанса) или ON COMMIT DELETE ROWS (на время транзакции)**

**Временные таблицы глобальны, создаются во временном табличном пространстве, после окончания сеанса (основаны на сессии) или транзакции (основаны на транзакции) таблица очищается, на временных таблицах можно определять ограничения, триггеры, индексы, но нельзя размещать в кластере.** Определены "статически" - создаются один раз для каждой базы данных, а не один раз для хранимой процедуры в базе данных. **Они всегда существуют, но кажутся пустыми, пока не поместим данные**. **Данные зависят от сеанса**. **То есть только мы можем видеть свои данные, даже если 100 пользователей используют одну и ту же таблицу**

**MS SQL:** После создания все временные таблицы сохраняются в таблице tempdb, название временной таблицы начинается со знака решетки # или ##, **# - локальная (на текущую сессию), # - глобальная (на все открытые сессии), удалятся либо, когда отключится пользователь, работающий с ней (если локально), и все пользователи - если глобально**. **Никто, кроме вас, не узнает, что ваша временная таблица существует**

**Отличие от обычных: нет flashback**

**Нельзя секционировать их, размещать в кластере**

**На начало сеанса ПУСТЫ**

**МОЖНО: триггеры, индексы, ОГРАНИЧЕНИЯ**

\* **on commit preserve rows – данные существуют на время сеанса, возможны все DML, TCL**

**\* on commit delete rows – данные существуют на время транзакции,** возможны все DML, после выполнения commit / rollback таблица станет пустой

1. Ограничения целостности в таблицах.
2. **data type** – предотвращение появления в столбце значений, не соответствующих заданному типу данных
3. **default** – значение по умолчанию для столбца
4. **pimary key** – первичный ключ, уникален и не null, однозначно идентифицирует запись в таблице
5. **not null** – запрет на неопределенные значения. Указывает, что столбец не может содержать неопределенных значений
6. **check** – обеспечивает соответствие столбца определенным условиям

n number CHECK (n<1)

1. **unique** – гарантирует уникальность, могут быть на комбинации, допускает null

CONSTRAINT n UNIQUE (n, name)

1. **ссылочные** (FK, foreign key) ***– внешний ключ, это ограничение указывает, что столбец или совокупность столбцов принимают значения, которые должны совпадать со значениями столбца (совокупности столбцов) из другого связанной таблицы.*** Это так называемое референциальное ограничение целостности

CONSTRAINT n REFERENCES tabl2 (m);

**Ссылочная целостность - корректность значений**[**внешних ключей**](https://ru.wikipedia.org/wiki/%D0%92%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87)[**реляционной базы данных**](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)**.**

Ограничения ссылочной целостности гарантируют, что значения определенных важных столбцов будут иметь смысл. Предположим, что есть родительская таблица, которая ссылается на значения из другой таблицы, как в случае таблиц *dept* и *employee*.Сотрудник в таблице employee не может быть назначен в департамент, если такой департамент в таблице *dept* не существует. Гарантировать существование действительного департамента можно с помощью ограничения ссылочной целостности.

**Целостность базы данных (**[**англ.**](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)**database integrity) — соответствие имеющейся в**[**базе данных**](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)**информации её внутренней логике, структуре и всем явно заданным правилам**. **Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется**[**ограничением целостности**](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1)**.**

**Нормализация – это процесс удаления избыточных данных. Нормализация – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности.**

Избыточность устраняется, как правило, за счёт декомпозиции отношений (таблиц), т.е. разбиения одной таблицы на несколько.

* Устранения аномалий
* Повышения производительности
* Повышения удобства управления данными
* Избыточность данных – это когда одни и те же данные хранятся в базе в нескольких местах, именно это и приводит к аномалиям.

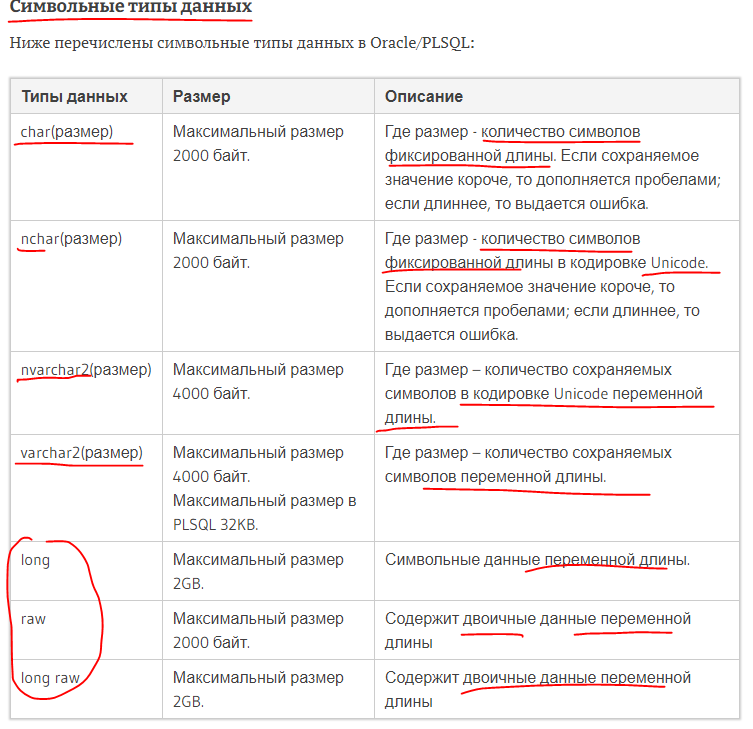
**Нормальная форма базы данных** – это набор правил и критериев, которым должна отвечать база данных.

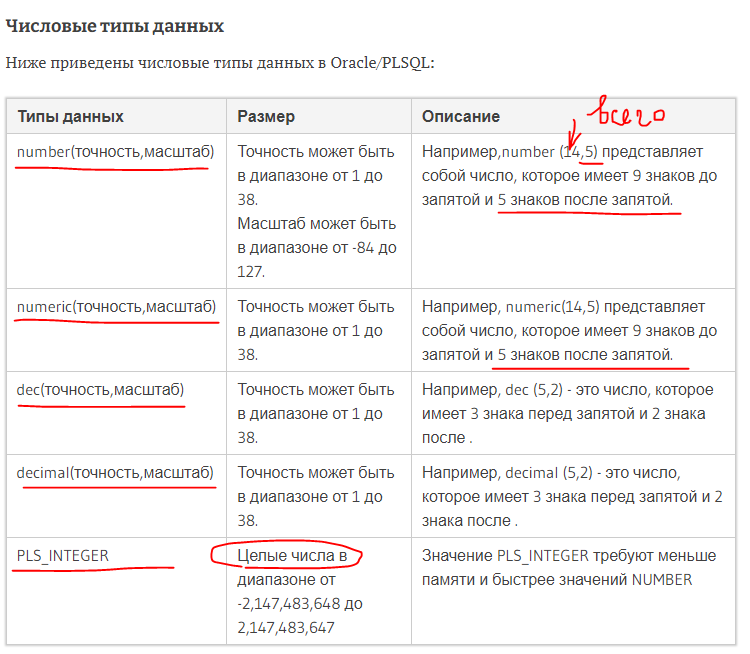
**Процесс нормализации** – это последовательный процесс приведения базы данных к эталонному виду, т.е. переход от одной нормальной формы к следующей.

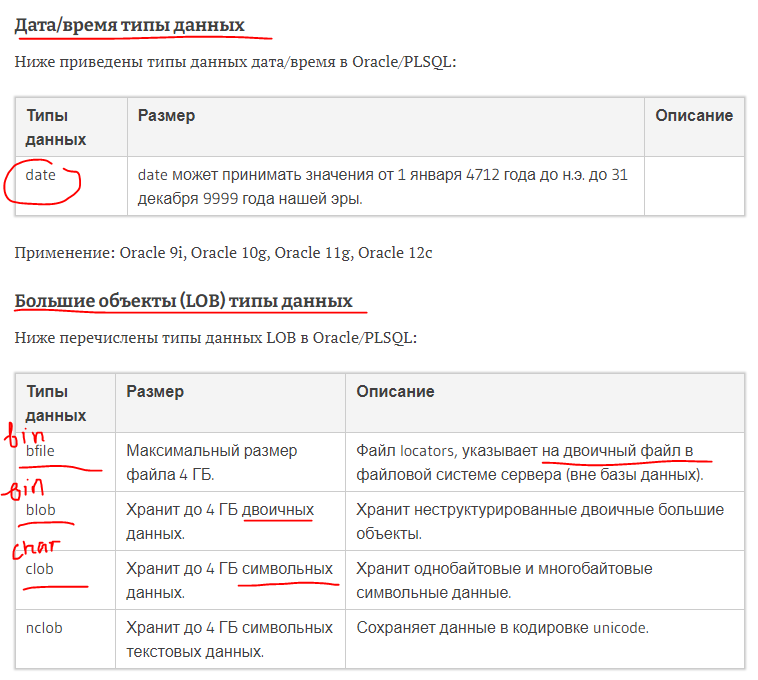
|  |  |  |
| --- | --- | --- |
| 1NF | 2NF | 3NF |
| Все значения полей **НЕДЕЛИМЫ (атомарные) и НЕВЫЧИСЛИМЫ**  **Все записи УНИКАЛЬНЫЕ (нет полностью совпадающих строк)** | **Быть в 1NF**  **Все неключевые поля полностью зависят от ключевого**  Таблица имеет ключ  Ключ – столбец или набор столбцов, по которому можно гарантированно отличить строки друг от друга. Ключ идентифицирует каждую строку таблицы | **Быть в 2NF**  **Все неключевые поля не зависят друг от друга – отсутствуют транзитивные зависимости** |

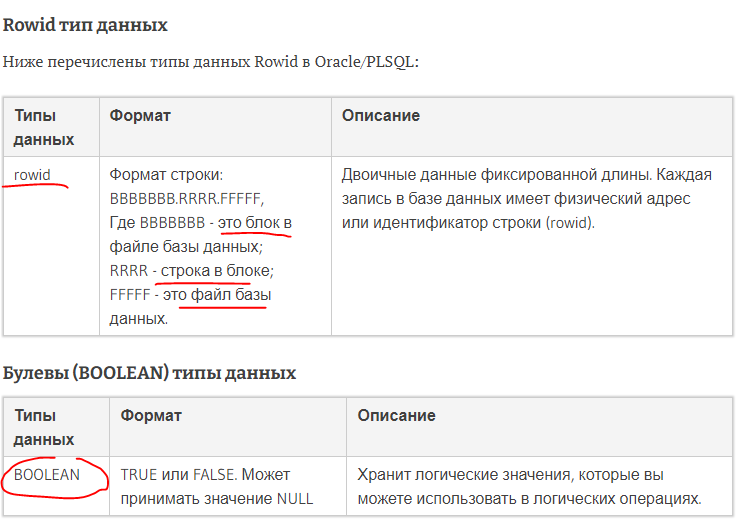
Generated always as identity – значения определяются сервером на основе последовательности

1. Типы данных базы данных.









**ROWID 16-тиричный тип, однозначно идентифицирует строку в таблице**

**UROWID – специальный 16-тиричный тип данных для адресации строк в таблицах, организованных по индексу**

Длина – 18 символов, которая делится на 4 элемента   
(6 + 3 + 6 + 3):

**OOOOOO – уникальный номер объекта в БД, которому принадлежит строка**;

**FFF – уникальный номер файла данных** БД, где хранится строка;

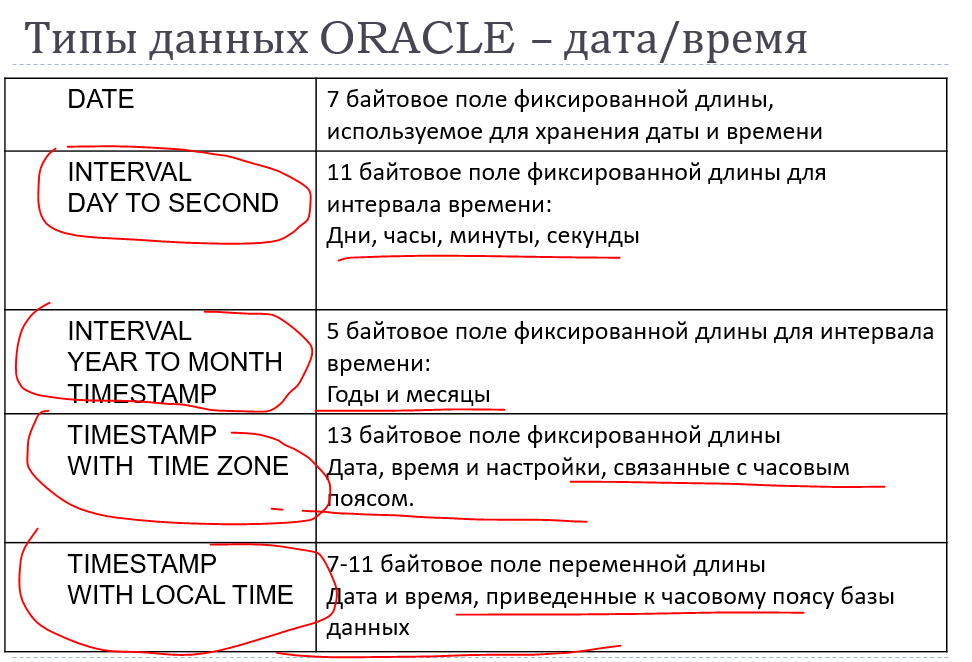
**BBBBBB –номер блока данных**, который хранит строку, уникален на уровне файла данных БД;

**RRR – адрес строки в блоке**

Псевдостолбец ROWID однозначно идентифицирует строку в базе данных**. Для каждой строки в базе данных ROWID псевдостолбец возвращает адрес строки. Значения rowid базы данных Oracle содержат информацию, необходимую для поиска строки:**

Номер строки - **Блок** данных в файле данных, в котором находится строка - **Позиция строки в блоке данных** - **Файл данных, в котором находится строка**

* **INTERVAL YEAR TO MONTH – сохраняет интервалы**, **используя год и месяц.**
* **INTERVAL DAY TO SECOND – сохраняет интервалы, используя дни, часы, минуты и секунды, включая доли секунды.**

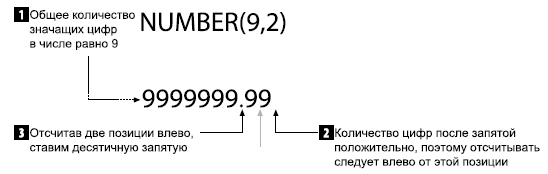


number(n,s), где n – общее число цифр, s – цифр после запятой

**Поясните, какое максимальное значение можно записать в тип данных NUMBER (4, 6)?**

**0,009999**

**Пишем 4 девятки и сдвигаем влево на 6**



**Поясните, какое максимальное значение можно записать в тип данных NUMBER (4, -2)?**

999999

**Пишем 4 девятки и сдвигаем вправо на 2**

**Поясните, какое максимальное значение можно записать в тип данных NUMBER (2, 1)?**

**9,9**

**2 – всего цифр, 1 – цифр после запятой**

1. Индексы базы данных. Виды и особенности применения индексов.

**Индекс – структура базы данных, используемая сервером для быстрого поиска строки в таблице**

**Индексы Oracle обеспечивают быстрый доступ к строкам таблиц, сохраняя отсортированные значения указанных столбцов и используя эти отсортированные значения для быстрого нахождения ассоциированных строк таблицы**. Индексы позволяют находить строку с определенным значением столбца, просматривая при этом лишь небольшую часть общего объема строк таблицы. **Таким образом правильное использование индексов сокращает до минимума количество дорогостоящих операций ввода-вывода**.

create **index <index\_name>** on **<table\_name> ( <column1>, <column2>, … );**

CREATE INDEX employee\_id ON employee(employee\_id)

TABLESPACE MY\_INDEXES;

Применение индексов представляет собой компромисс между ускорением получения результатов запросов и замедлением обновлений и вставок данных. Первая часть этого компромисса – ускорение запросов – довольно очевидна: **если поиск выполняется по отсортированному индексу вместо полного сканирования всей таблиц, то запрос проходит намного быстрее.** Но всякий раз, когда вы обновляете, вставляете или удаляете строку таблицы с индексами, индексы также должны быть обновлены соответствующим образом. То есть такие операции на таблицах с индексами обходятся дороже.

Индексы могут относиться к нескольким типам, наиболее важные из которых перечислены ниже:

* **Уникальные и неуникальные индексы. Уникальные индексы основаны на уникальном столбце** – обычно вроде номера карточки социального страхования сотрудника. Хотя уникальные индексы можно создавать явно, Oracle не рекомендует это делать. **Вместо этого следует использовать уникальные ограничения. Когда накладывается ограничение уникальности на столбец таблицы, Oracle автоматически создает уникальные индексы по этим столбцам.**
* **Первичные и вторичные индексы. Первичные индексы – это уникальные индексы в таблице, которые всегда должны иметь какое-то значение и не могут быть равны null.** Вторичные индексы – это прочие индексы таблицы, которые могут и не быть уникальными.
* **Составные индексы – индексы, содержащие два или более столбца из одной и той же таблицы. Они также известны как сцепленные индексы (concatenated index).** Составные индексы особенно полезны для обеспечения уникальности сочетания столбцов таблицы в тех случаях, когда нет уникального столбца, однозначно идентифицирующего строку.

**В реализации индексов на основе B-деревьев используется концепция сбалансированного (на что указывает буква ‘B’ (balanced)) дерева поиска в качестве основы структуры индекса.** В Oracle имеется собственный вариант B-дерева. Это обычные индексы, создаваемые по умолчанию, когда вы применяете оператора CREATE INDEX.

**Индексы на основе B-деревьев структурированы в форме обратного дерева, где блоки верхнего уровня называются блоками ветвей (branch blocks), а блоки нижнего уровня – листовыми блоками (leaf blocks).** В иерархии узлов все узлы кроме вершины, или корневого узла, имеют родительский узел и могут иметь ноль или более дочерних узлов. Если глубина древовидной структуры , т.е. количество уровней, одинакова от каждого листового блока до корневого узла, то такое дерево называется сбалансированным, или B-деревом.

B-деревья автоматически поддерживают необходимый уровень индекса по размеру таблицы. **B-деревья также гарантируют, что индексные блоки всегда будут заполнены не меньше, чем наполовину, и менее, чем на 100%.** B-деревья допускают операции выборки, вставки и удаления с очень небольшим количеством операций ввода-вывода на один оператор. Большинство B-деревьев имеет всего три и менее уровней. **При использовании B-дерева нужно читать только блоки B-дерева, так что количество операций ввода-вывода будет ограничено числом уровней B-дерева (скажем, тремя) плюс две операции ввода-вывода на выполнение обновления или удаления (одна для чтения и одна для записи).** Для выполнения поиска по B-дереву понадоисят всего три или менее обращений к диску.

Реализация B-дерева от Oracle – всегда сохраняет дерево сбалансированным. Листовые блоки содержат по два элемента: индексированные значения столбца и соответствующий идентификатор ROWID для строки, которая содержит это значение столбца. **ROWID – уникальный указатель Oracle, идентифицирующий физическое местоположение строки и обеспечивающий самый быстрый способ доступа к строке в базе данных Oracle. Сканирование индекса быстро дает ROWID строки, и отсюда можно быстро получить к ней доступ непосредственно.**



**Оптимизатор запроса — встроенное в СУБД программное обеспечение, которое определяет наиболее эффективный способ выполнения SQL-выражения**

* **План выполнения запроса — последовательность шагов или инструкций СУБД, необходимых для выполнения SQL-выражения**
* **Стоимость выполнения запроса — наилучшая оценка времени, необходимого для выполнения оператора, полученная оптимизатором**

**Оптимизатор определяет наилучший план выполнения разными способами:**

* + **полное сканирование таблицы full table scan**
  + **использование индексов**
  + **различные типы соединения**

**Селективность таблицы — значение, представляющее долю строк таблицы, удовлетворяющих определенному условию выбора**

Селективность таблицы связана с условием выбора строк

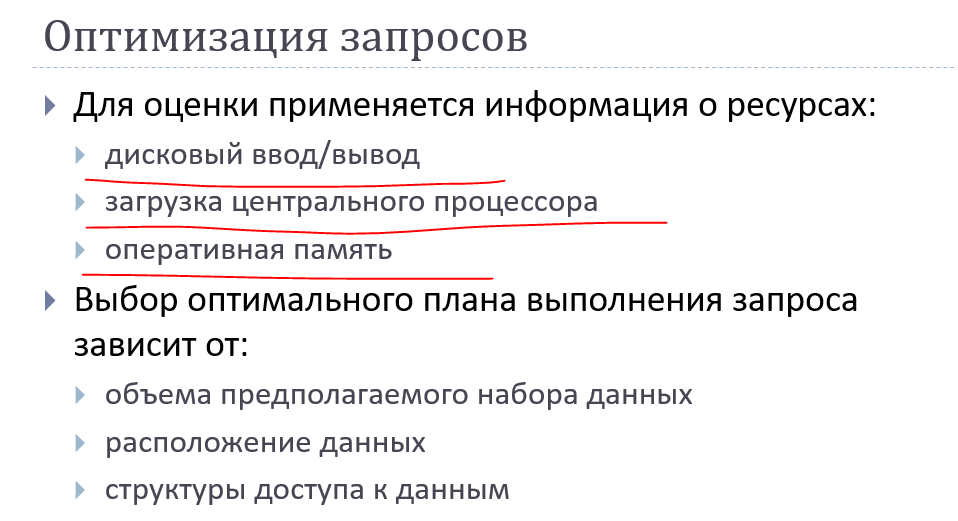
**Селективность индекса — значение, представляющее отношение количества уникальных значений индексируемых столбцов к общему числу строк таблицы**

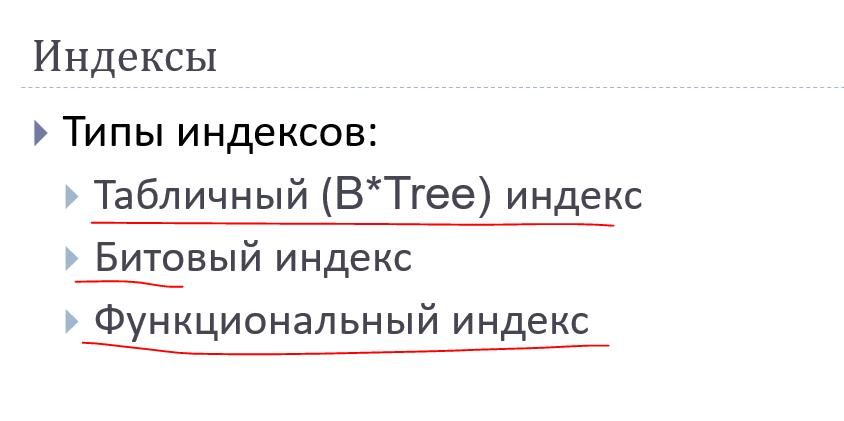
Селективность индекса показывает долю строк от общего числа строк в таблице, которое приходится на одно значение индекса

**Кардинальность — количество строк, возвращаемых после каждой операции плана выполнения запроса**

Значение кардинальности равно произведению селективности на количество обработанных строк

F10 ПЛАН ВЫПОЛНЕНИЯ ЗАПРОСА





**Индекс является структурой на диске, которая связана с таблицей или представлением и ускоряет получение строк из таблицы или представления**. **Индекс содержит ключи, построенные из одного или нескольких столбцов в таблице или представлении. Эти ключи хранятся в виде структуры сбалансированного дерева, которая поддерживает быстрый поиск строк по их ключевым значениям**

|  |  |  |  |
| --- | --- | --- | --- |
| B-tree | Битовый | Функциональный | Кластеризованный (кластерный) |
| Табличный индекс (B\*Tree) структурирован ***в виде сбалансированного дерева***  ***Листовой блок содержит индексированные значения столбца и соответствующий ему идентификатор строки (RowId)***  ***Предназначен для индексирования уникальных столбцов или столбцов с высокой селективностью*** | ***Битовый индекс создает битовые карты для каждого возможного значения столбца***, где каждому биту соответствует строка, а ***значение бита 1 (0) означает, что соответствующая строка содержит (не содержит) индексируемое значение***  ***Предназначен для индексирования столбцов с низкой селективностью***  Не подходит для таблиц с частым обновлением  **Хорошо подходят для хранилищ данных** | ***предварительно вычисляют значения функции по заданному столбцу и сохраняют результат в индексе*** | ***Кластеризованные индексы сортируют и хранят строки данных в таблицах или представлениях на основе их ключевых значений.*** Этими значениями являются столбцы, включенные в определение индекса. Существует только один кластеризованный индекс для каждой таблицы, так как строки данных могут храниться в единственном порядке |

***кластеризованный индекс хранит реальные данные, нельзя создать более одного кластеризованного индекса в таблице.******Некластеризованный индекс не содержит реальных данных таблицы в узлах-листьях. Узлы-листья содержат один из двух типов информации о местоположении строк данных***

1. Последовательность СУБД Oracle и ее параметры.

**Последовательность – объект базы данных, предназначенный для генерации числовой последовательности. Привилегия grant create sequence**

**Этот объект Oracle, который используется для генерации последовательности чисел.** Это может быть полезно, когда нужно создать identity в качестве первичного ключа.

**create sequence xxxxx**

**increment by** \_\_\_ *приращение*

**start with** \_\_\_ *первое генерируемое значение*

**momaxvalue / maxvalue** \_\_\_ *1027*

**nominvalue / minvalue** \_\_\_\_ -1026

**nocycle / cycle** \_\_\_\_\_ *позволяет последовательности повторно использовать созданные значения по достижению maxvalue или minvalue*

**cache 20 / nocache** *сколько элементов последовательности Oracle распределяет заранее и поддерживает в памяти для быстрого доступа*

**order / noorder** *номера последовательности генерируются в порядке запросов*

select **ИМЯ.CURRVAL** from dual;

select **ИМЯ.NEXTVAL** from dual;

select \* from dba\_sequences;

1. Кластер и его параметры

**Кластер – таблицы, с которыми часто работают совместно, можно физически хранить совместно. Создается кластер, который будет их содержать**. Строки из отдельных таблиц сохраняются в одних и тех же блоках. ***Объединяющие эти таблицы запросы выполняются быстрее, что уменьшает количество операций ввода/вывода.***

**Связанные столбцы называются кластерным ключом**.

**Кластер – специализированный объект базы данных, используемый для физически совместного хранения одной или нескольких таблиц, которые часто соединяются вместе в SQL-запросах.**

Если 2 таблицы имеют идентичный столбец, и нам часто приходится соединять таблицы по нему, то тогда выгодно хранить значения общих столбцов в одном и том же блоке данных – КЛАСТЕРЕ

Привилегия grant create cluster

**create cluster xxxx**

**( id number )**

**hashkeys \_\_\_\_**

create table T1

( …., id number)

**cluster xxxx (id)**

create table T2

(id number)

**cluster xxxx(id)**

**T1 и T2 частый join вместе по id**

***Кластер – несколько объектов одного типа, ВМЕСТЕ. Это объект базы данных. Он позволяет быстрее доставать данные из связанных таблиц – он знает, что таблицы связаны,*** и тогда он часть данных хранит в себе, эти данные извлекаются быстрее.

Представление dba\_clusters

select \* from dba\_clusters;

*Хэш-кластеры: используются функции хэширования кластерного ключа строки для определения физической локализации места, где строку следует хранить*

Можно создавать хэш-кластер и поместить в него таблицы. Строки **извлекаются в соответствии с результатом некоторой хэш-функции**. **Параметр hashkeys показывает максимальное количество уникальных хэш-значений, которые могут быть сгенерированы хэш-функцией**

**Ключ кластера – хэшируем его, это связанные столбцы**

**Значение ключа кластера указывает на область диска, где хранятся строки**

**1 операция ввода-вывода**

Преимущество – в запросах, использующих операции равенства:

where id = 999;

1. Представление и его параметры.

**Представление – именованный select-запрос. Обращение происходит как к обычной таблице**. Добавляют уровень защиты. Скрывают сложность данных, имена столбцов таблиц

create view xxx as

select \* from orders;

Команда SELECT на которой базируется представление, может быть чем угодно. **Объединением таблиц, результатом после аггрегации, сортировки. Абсолютно любой валидный запрос может быть основой для представления**.

Добавление к оператору *CREATE VIEW* **конструкции *WITH READ ONLY* гарантирует, что пользователи смогут только осуществлять выборку данных из представления**. ***Это означает, что пользователи не смогут модифицировать представление и тем самым неявно обновлять, вставлять или удалять строки базовых таблиц.***

Хотя представления используются в основном для запросов, при некоторых обстоятельствах их можно также применять в командах *INSERT, DELETE* и *UPDATE*. Например, допускается выполнять операции DML над представлениями, **которые не имеют в своем определении конструкций *GROUP BY,* *START WITH* или *CONNECT BY*, либо каких-то подзапросов в своей конструкции *SELECT***. Однако поскольку представление в действительности не существует как отдельная физическая сущность, **на самом деле происходит модификация данных лежащих в его основе таблиц, и само представление будет, таким образом, субъектом тех же ограничений целостности, что и таблицы, на которых оно основано**.

Параметры:

**Create or replace**

\* **FORCE** – созд предст, неважно сущ. ли таблицы и есть ли права

\* **NOFORCE** – по умолчанию

\* **WITH CHECK OPTION** – нельзя update/insert, кот. не входят в это предст

\* **WITH READ ONLY**

FORCE или NOFORCE – **Использование FORCE приведёт к созданию представления даже если базовые таблицы не существуют**. NOFORCE значение по умолчанию и если таблицы не существуют команды выполняется с ошибкой

WITH CHECK OPTION – эта директива влияет на DML команды. **Если подзапрос включает условие WHERE, тогда эта директива предотвратит возможность вставки строк, которые не видно в представлении (КОТОРЫЕ ПРЕДСТАВЛЕНИЕ НЕ МОЖЕТ ВЫБРАТЬ), или совершать обновления данных которое приведёт к пропаже данных из представления.** По умолчанию эта директива отключена что может приводить к неожидаемым результатам выполнения запросов

**WITH READ ONLY – отключения возможности использование DML команд к представлению. Представление только для чтения**

1. Материализованное представление и его параметры.

**Базовое представление** create view – всякий раз, когда мы обращаемся к представлению, оно рекомпилирует данные для представления актуальной информацией в соответствии с надлежащим запросом. **Мы можем вносить изменения в представление или в базовую таблицу, и данные будут автоматически обновляться в обоих местах. Изменение в базовых таблицах автоматически проникает в представление. По этой причине может потребоваться время для обновления представления**

Всякий раз, когда нужен доступ к представлению, Oracle **должен выполнить запрос, по которому определено представление, и вернуть результат**. **Этот процесс наполнения представления называется разрешением представления (view resolution) и он повторяется при каждом обращении пользователя к представлению.** Если вы имеете дело с представлениями с множеством конструкций *JOIN* и *GROUP BY*, **то этот процесс разрешения представления может потребовать очень длительного времени**. Если нужно часто обращаться к представлению, будет весьма неэффективно каждый раз повторять разрешение представления.

Материализованные представления Oracle предлагают выход из этого затруднения. Упомянутые представления можно воспринимать как специализированные представления, в отличие от обычных представлений, **имеющие физическое воплощение.** *Они занимают место и требуют хранения подобно обычным таблицам.* ***Материализованные представления можно даже секционировать и при необходимости создавать на них индексы.***

**На заметку!** **Представление всегда вычисляется на лету, и его данные не хранятся отдельно от таблиц, на которых оно определено. Таким образом, запросы, использующие представления, по определению гарантированно вернут самые свежие данные**. **Материализованные представления в базе данных Oracle Database, с другой стороны, являются статическими объектами, которые наследуют свои данные от лежащих в их основе базовых таблиц. Если вы будете обновлять свои материализованные представления нечасто, то данные в них могут устареть по отношению к данным таблиц, на которых они основаны.**

**Материализованное представление materialized view – ДАННЫЕ ФИЗИЧЕСКИ СОХРАНЯЮТСЯ В ОПРЕДЕЛЕННЫЙ МОМЕНТ ВРЕМЕНИ, нам не потребуется всякий раз повторно считывать данные, связанные с запросом => недостаток – не гарантируется, что мы видим самые свежие данные => НАДО ОБНОВЛЯТЬСЯ САМИМ**

Привилегия grant create materialized view

В крупных базах данных Oracle с интенсивными действиями, затратными по времени и вычислительной мощности процессоров, такими как соединение таблиц и использование агрегатных функций вроде SUM, материализованные представления ускоряют запросы. **Материализованные представления обеспечивают более быстрое выполнение запросов за счет перерасчета и хранения результатов дорогостоящих соединений и агрегатных операций**. Прелесть материализованных представлений Oracle заключается в том, что при их создании можно указать, что база данных должна автоматически обновлять материализованные представления, когда происходят изменения в положенных в их основу таблицах. **Материализованные представления полностью прозрачны для пользователей. Если пользователи пишут запросы с обращением к лежащим в основе таблицам, то Oracle автоматически переписывает их для использования материализованных представлений, и такая техника оптимизации запросов называется переписыванием запроса (query rewrite).** Стоимостной оптимизатор Oracle автоматически распознает необходимость в переписывании запроса для использования материализованного представления вместо исходных таблиц, если оценочная стоимость такого запроса оказывается ниже. **Под стоимостью запроса здесь подразумевается объем ввода-вывода, а также затраты времени процессора и памяти, связанные с обработкой SQL-запроса. Сложные соединения таблиц обходятся в этом смысле дорого, а применение материализованных представлений позволяет использовать уже сохраненную информацию в предварительно вычисленном виде, и запросы требуют гораздо меньше ресурсов и потому выполняются намного быстрее.**

Прием автоматической оптимизации с переписыванием запроса лежит в основе применения материализованных представлений. Параметр инициализации QUERY\_REWRITE\_ENABLED позволяет включать и отключать это средство на глобальном уровне.

Этот параметр может принимать следующие значения.

* FALSE. База данных не переписывает никакие запросы.
* TRUE. База данных сравнивает стоимость запроса с переписыванием и без, и выбирает наиболее дешевый метод.
* FORCE. База данных всегда переписывает запрос, не оценивая стоимости.Используйте вариант FORCE, если уверены, что это даст выигрыш во времени получения ответа.

**create materialized view xxxx**

**log on \_\_\_\_\_ with rowid, primarykey including new values** – создать журналы МП, вспомогательные таблицы, накапливают сведения об изменениях, которые происходят в базовых таблицах

**build immediate** – создать сразу

**start with** – когда выполнить первый раз, если построено сразу

**next** – когда следующий раз (start with – next = периодичность)

**refresh**

**complete** полное обновление

**fast** использовать журналы фиксации базовой таблицы

**force** – быстрое обновление

**on commit – при commit**

**on demand – по требованию**

**enable query rewrite**  - оптимизатор Oracle прозрачно перепишет все запросы для использования материализованных представлений вместо лежащих в основе главных таблиц

Представления USER\_MVIEWS; USER\_MVIEW\_LOG

**create materialized view** V\_MATERIALIZED\_WORKERS as (select OFFICES.office\_id, CITY, WORKER\_ID, WORKER\_NAME from OFFICES inner join WORKERS on OFFICES.office\_id = WORKERS.office\_id);

select \* from V\_MATERIALIZED\_WORKERS;

alter materialized view V\_MATERIALIZED\_WORKERS **refresh complete on demand next** sysdate + NUMTODSINTERVAL(2, 'MINUTE');

**create materialized view mviewgetdata**

**refresh complete on demand**

**next sysdate + NUMTODSINTERVAL(2, 'MINUTE')**

**as (select \* from T\_EXAM);**

select \* from mviewgetdata;

--CREATE MATERIALIZED VIEW view-name

--BUILD [IMMEDIATE | DEFERRED]

--REFRESH [FAST | COMPLETE | FORCE ]

--ON [COMMIT | DEMAND ]

--[[ENABLE | DISABLE] QUERY REWRITE]

--AS

--SELECT ...;

1. Частные и публичные синонимы СУБД Oracle.

**Синоним – АЛЬТЕРНАТИВНОЕ ИМЯ ОБЪЕКТА, псевдоним объекта базы данных, который служит в основном для облегчения пользовательского доступа к объектам, принадлежащим другим пользователям, а также в качестве безопасности. Это способ обращаться к объекту базы данных без указания полной идентификации объекта (хост – экземпляр – владелец – объект)**

**Представление dba(user|all)\_synonyms;**

**Частный синоним: принадлежит пользователю, который его создал**

create synonym xxxx for \_\_\_

**Публичный синоним: используется совместно ВСЕМИ пользователями базы данных**

create public synonym xxxx for \_\_\_\_

Привилегии grant create (public) synonym

create synonym dasha\_c for C##DASHA.C;

создает синоним dasha\_c для таблицы C, расположенной в схеме C##DASHA

Синонимы **скрывают идентичность лежащих в их основе объектов и могут быть как приватными (private), так и общедоступными (public).** Общедоступные синонимы доступны всем пользователям базы данных, а **приватные синонимы являются составной частью схемы отдельного пользователя,** и другим пользователям базы следует выдавать права доступа для использования приватных синонимов. **Синонимы Oracle могут быть созданы для таблиц, представлений, материализованных представлений, последовательностей и хранимого кода — пакетов и процедур, функций.**

* Для обеспечения прозрачности объектов. Синонимы могут быть созданы для обеспечения прозрачности исходных объектов для пользователя.
* **Для прозрачности расположения. Синонимы могут быть созданы как псевдонимы таблиц и прочих объектов, относящихся к нелокальной базе данных.**

1. Основные характеристики языка PL/SQL.

**Pl/SQL – Procedural Language Extensions to SQL – процедурное языковое расширение языка SQL. Это встроенный (внутренний) язык, работает только в конкретной управляющей среде – SQL Developer, SQL Plus, TOAD**. Это не самостоятельный язык. **Основной язык для программирования хранимых процедур, а также стандартный переносимый язык разработки приложений для базы данных Oracle. «Написать один раз и использовать везде»**

Высокопроизводительный

Высокоинтегрированный

Большой контроль – циклы, условия, ООП концепции

Не чувствителен к регистру

Нету накладных расходов на приведение типов

**Структурное программирование – процедуры и функции**

**Типы модулей – анонимные блоки, функции, пакеты, процедуры, объектные типы**

**PL/SQL функции можно вызывать из select**

**Внутренний язык Oracle**

**Позволяет использовать объектные типы**

**Интерпретация (по умолчанию) и компиляция (промежуточный код на С и конечный объектный код процессора)**

**Приложение может быть проще в реализации при написании бизнес-логики на основе хранимых процедур**

**Позволяет использовать в базе данных такие программные единицы, как ПРОЦЕДУРЫ И ПАКЕТЫ – увеличивает возможность повторного использования кода**

Программа на PL/SQL состоит из [блоков](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (анонимных или поименованных). Блок может содержать вложенные блоки, называемые иногда подблоками. Общая форма PL/SQL-блока:

**DECLARE**

*-- Описания блока, переменные, типы, курсоры и т. п. (опционально)*

**BEGIN**

*-- Непосредственно код программы*

**EXCEPTION**

*-- Обработка исключений (опционально)*

**END**;

*/\* Многострочные*

*комментарии… \*/*

*-- Однострочный комментарий*

Язык PL/SQL позволяет определять следующие типы именованных блоков:

* **процедуры;**
* **функции;**
* **объекты;**
* **пакеты.**

**Все они могут быть**[**скомпилированы**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))**и сохранены как объекты базы данных в некоторой её схеме**. **Все именованные блоки кода, кроме пакетов, не хранят внутреннее состояние от вызова к вызову.**

**Пакеты обеспечивают модульность для больших проектов, позволяя сгруппировать наборы именованных блоков кода, кроме того, в пакетах возможно хранение состояния на время жизни сессии базы данных, доступное для функций и процедур, входящих в пакет**. Пакеты в PL/SQL содержат спецификацию и тело. **Спецификация пакета может содержать определение констант, переменных, типов данных, объявление процедур и функций. Тело пакета определяет объявленные в спецификации процедуры и функции, а также может содержать блок кода инициализации пакета, определения внутренних констант, переменных, типов данных, процедур и функций.** Все компоненты пакета, объявленные в его спецификации, могут быть доступны для использования извне пакета, **в то время как тело пакета**[**инкапсулирует**](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))**реализацию этих компонентов, и извне недоступно**. Тело и спецификация пакета могут модифицироваться, компилироваться и сохраняться независимо друг от друга.

show parameter plsql;

1. Структура программы языка PL/SQL. Анонимные и именованные блоки.

**Программа на PL/SQL состоит из**[**блоков**](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))**(анонимных или поименованных). Блок может содержать вложенные блоки, называемые иногда подблоками**. Общая форма PL/SQL-блока - анонимного блока:

**DECLARE**

*--* ***Описания блока, переменные, типы, курсоры и т. п.*** *(опционально)*

**BEGIN**

*--* ***Непосредственно код программы***

**EXCEPTION**

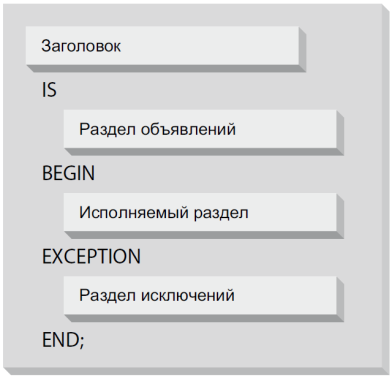
*--* ***Обработка исключений (опционально)***

**END**;

***/\* Многострочные***

***комментарии… \*/***

***-- Однострочный комментарий***



Блок в PL/SQL – это исполняемая программа. Блок состоит из следующих структур:

**Declare**

**Begin ОБЯЗАТЕЛЬНО! Начало исполняемого кода**

**Exception**

**End ОБЯЗАТЕЛЬНО! Конец исполняемого кода**

**Анонимный блок – блок без имени. Другие блоки не могут его вызвать – нет идентификатора обращения:**

- не имеет строки заголовка

- используется как скрипт

- не вызываем из другого блока

- начинается обычно с declare/begin

- варианты использования: триггеры, скрипты

**begin**

**null; --простейший**

**end;**

Исключения возникают, когда механизм PL/SQL встречает инструкцию, которую он не может выполнить из-за ошибки, возникающей во время ее выполнения. Блок обработки исключений **exception**

**when others – обработка любого исключения, УНИВЕРСАЛЬНЫЙ ОБРАБОТЧИК**

**sqlerrm – функция возвращает сообщение об ошибке**

**sqlcode – функция возвращает номер ошибки (обычно 5-значный)**

**raise – распространение возникшего исключения в родительский блок (для вложенного блока) raise\_application\_error(-20001, ‘message’);**

**блоков when сколько угодно, самый последний – when others**

В PL/SQL каждый блок может быть вложен в другой. Вложенные блоки распространены, когда мы хотим выполнить определенный процесс, и в то же время код для него должен быть отделен от логики основного (главного, родительского) блока. Разделить сложные вещи на каждый блок и обрабатывать исключения внутри каждого вложенного блока внутри основного вложенного блока

declare  
….  
begin  
 begin  
 …   
 end;  
end;

Переменные и исключения локальны в рамках блока. **Переменные, объявленные в родительском блоке, видны во вложенном. Вложенных блоков может быть много**

ПРЕДУПРЕЖДЕНИЯ КОМПИЛЯТОРА:

* **ALL (все);**
* **PERFORMANCE (производительность);**
* **INFORMATIONAL (информационные);**
* **SEVERE (логика программы);**
* **Specific error (ошибка);**

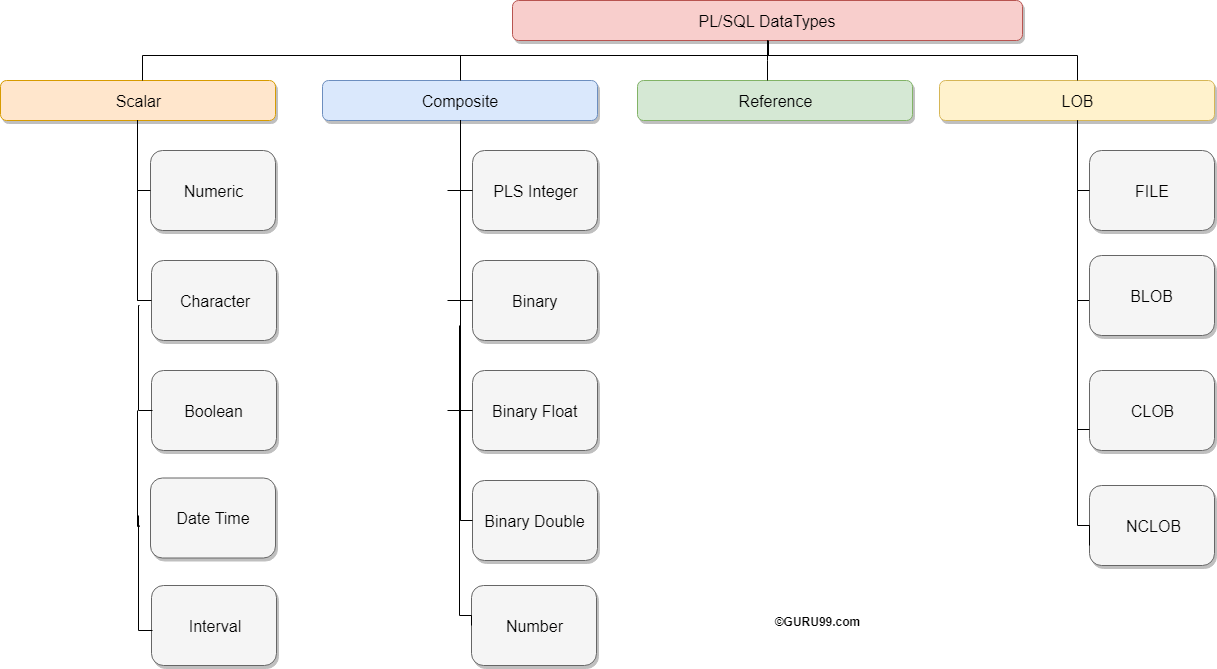
1. Типы данных, основные операции, константы языка PL/SQL.

Язык PL/SQL поддерживает следующие категории типов:

**- встроенные типы данных, включая коллекции и записи;**

* **скалярные;**
* **составные;**
* **ссылочные;**
* [**LOB-типы**](https://ru.wikipedia.org/wiki/LOB)**;**

**- объектные типы данных.**



Character (символьные)

* CHAR фиксированный размер строки
* VARCHAR2 переменный размер строки
* VARCHAR переменный размер строки, аналог varchar2
* NCHAR фиксированный размер строки unciode
* NVARCHAR2 переменный размер строки Unicode
* **LONG большой текст до 2 ГБ**
* **LONG RAW большой текст в двоичном формате**
* **RAW не используется оставлен для совместимости**

LOB

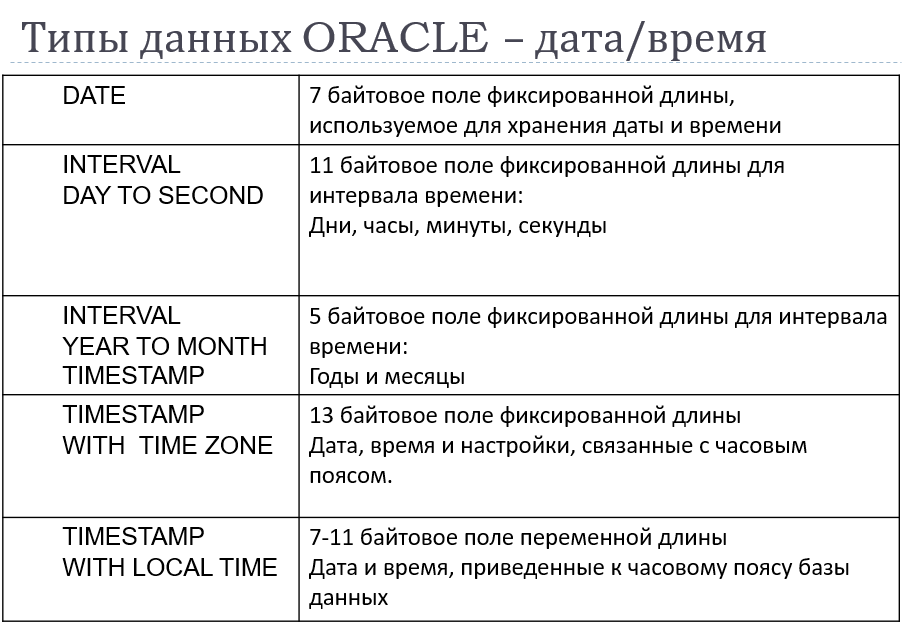
* **CLOB символьный тип большой объект**
* **BLOB двоичный тип большой объект**
* **NCLOB символьный тип большой объект Unicode**
* **BFILE двоичный файл**

Numeric (числовые)

* **Number (p, s)**
* **Binary\_double, binary\_float**
* **PLS\_INTEGER**
* **Natural**

**Логические**

* **boolean**



**имя тип\_данных [NOT NULL] [значение\_по\_умолчанию];**

**имя CONSTANT тип\_данных [NOT NULL] := | DEFAULT значение\_по\_умолчанию;**

**Значение константы задается в момент объявления и впоследствии не может быть изменено.**

**TO\_DATE( string1, [ format\_mask ], [ nls\_language ] )**

SELECT TO\_DATE('2019/07/22', 'yyyy/mm/dd') FROM DUAL;

--Результат:       22.07.2019

SELECT TO\_DATE('072219', 'MMDDYY') FROM DUAL;

--Результат:       22.07.2019

SELECT TO\_DATE('20190722', 'yyyymmdd') FROM DUAL;

--Результат:       22.07.2019

SELECT TO\_DATE('30.01.2019 18:30:52', 'DD.MM.YYYY HH24:MI:SS') FROM DUAL;

--Результат:       30.01.2019 18:30:52

**TO\_CHAR( value, [ format\_mask ], [ nls\_language ] )**

SQL> SELECT TO\_CHAR(1242.78, '9999.9') FROM DUAL;

--Результат:    1242.8

SQL> SELECT TO\_CHAR(-1242.78, '9999.9') FROM DUAL;

--Результат:   -1242.8

SQL> SELECT TO\_CHAR(1242.78, '9,999.99') FROM DUAL;

--Результат:    1,242.78

SQL> SELECT TO\_CHAR(1242.78, '$9,999.00') FROM DUAL;

--Результат:   $1,242.78

SQL> SELECT TO\_CHAR(18, '000099') FROM DUAL;

--Результат:    000018

**select** concat('Дарья ','Шейбак') **from** dual; --Дарья Шейбак

**select** lower(‘ДАРЬЯ ШЕЙБАК’) **from** dual; --дарья шейбак

**select** upper(‘дарья шейбак’) **from** dual; --ДАРЬЯ ШЕЙБАК

**select** initcap('дарья') **from** dual; --Дарья

**select** trim(' Дарья ') **from** dual; --Дарья, вырезает пробелы с начала и с конца строки, ltrim () вырежет слева, rtrim () вырежет справа

REPLACE( исходная\_строка, исходная\_подстрока, [ новая\_подстрока ] ) --Заменяет некоторую последовательность символов в строке другим набором символов

SUBSTR( **исходная\_строка, начальная\_позиция, [ длина\_подстроки ]** )

-- Если начальная\_позиция отрицательная - то отсчет идет от конца строки

-- Если начальная\_позиция равна 0 то считается что начальная позиция равна 1

-- Если начальная\_позиция положительная - то отсчет идет от начала строки

-- Если длина\_подстроки отрицательная то функция вернет NULL

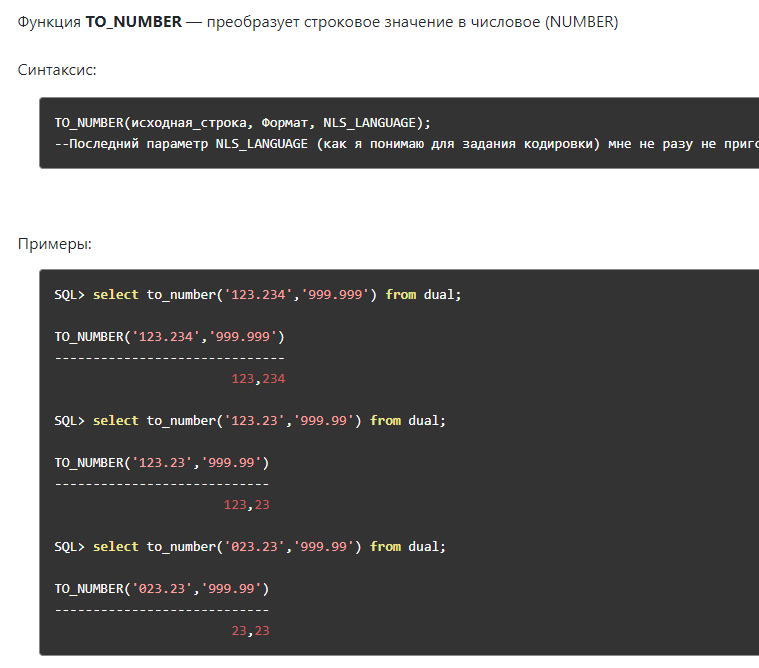
**select** substr('000111qa00zx0222333', 3, 5) **from** dual;

Строка первый параметр. С какой позиции – 3. Сколько символов - 5

**select** length(‘Дарья’) **from** dual; --возвращает длину строки

**VSIZE** — Возвращает длину строки в байтах

**select** ascii('2') **from** dual; --вернет код ASCII символа



1. Поддержка национальных языков в СУБД Oracle. Наборы символов. Байтовая и символьная семантика символов.

**NLS – *National Language Support позволяет хранить, обрабатывать и извлекать данные на национальных языках***. ***Это гарантирует, что утилиты базы данных и сообщения об ошибках, порядок сортировки, даты, время, денежные, числовые и календарные соглашения автоматически адаптируются к национальному языку и локали.***

3 компоненты NLS\_LANG (переменная окружения):

1. **Language язык – месяцы, дни, направления текста, сокращения дат и времени**
2. **Territory территория – календарь, формат дат и денежных единиц**
3. **Charset set набор символов – отображение символов**

NLS - National Language Support, далее Globalization Support

**Можно хранить данные множества национальных языков, используя Unicode или специальные кодировки – наборы символов** (character set)

Символы хранятся как коды символов, зависящие от выбранного набора символов

В одной БД могут использоваться два набора символов: **основной (database character set) и дополнительный (national character set)**

Устанавливаются при создании БД

**Изменяются alter database (national) character set**

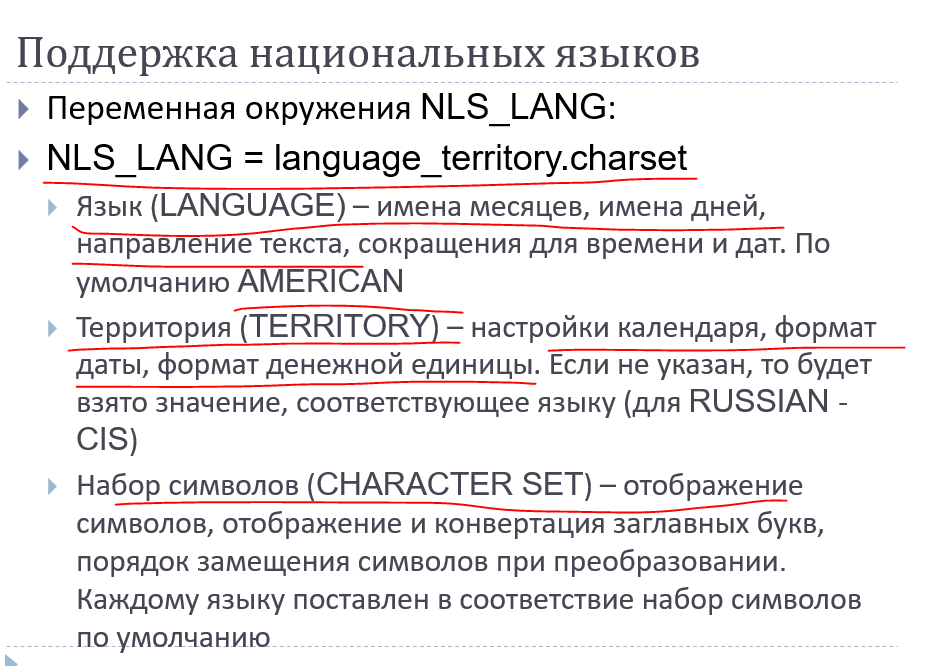
Основной набор символов используется для:

* + **хранения символьных типов** char, varchar2, clob и long
  + **описания имен объектов, переменных**
  + **Ввода и хранения PL/SQL модулей**

Дополнительный набор символов используется для:

* + **хранения символьных типов nchar, nvarchar2, nclob**

**Кроме символов алфавита в набор включаются знаки препинания, числа, символы денежных единиц и** **пр.**



**show parameter nls;**

**show parameter NLS\_LANG\_SEMANTICS;**

Представления словаря:

* **NLS\_SESSION\_PARAMETERS**
* **NLS\_INSTANCE\_PARAMETERS**
* **NLS\_DATABASE\_PARAMETERS**

|  |  |
| --- | --- |
| Символьная семантика CHAR | Байтовая семантика BYTE |
| Строка – набор символов  Можно задавать явно:  Varchar2(20 char)  ***Глобализованные данные и многобайтовые наборы*** | По умолчанию  Строка – последовательность байт  Можно задавать явно:  varchar2 (20 byte)  ***однобайтовые наборы символов*** |

1. Связанные объявления переменных: инструкция %TYPE, инструкция %ROWTYPE.

Тип переменной основан на известной структуре данных, если мы, например, не знаем заранее ее тип данных.

**Скалярная ссылка %TYPE для определения переменной на основе другой переменной или поля в таблице**

**Ссылка на запись %ROWTYPE для определения структуры записи на основе таблицы или курсора**

**Атрибут %TYPE позволяет объявить константу, переменную, поле записи или параметр подпрограммы так, чтобы они имели тот же тип данных, что и ранее объявленная переменная или столбец (не зная, что это за тип).**

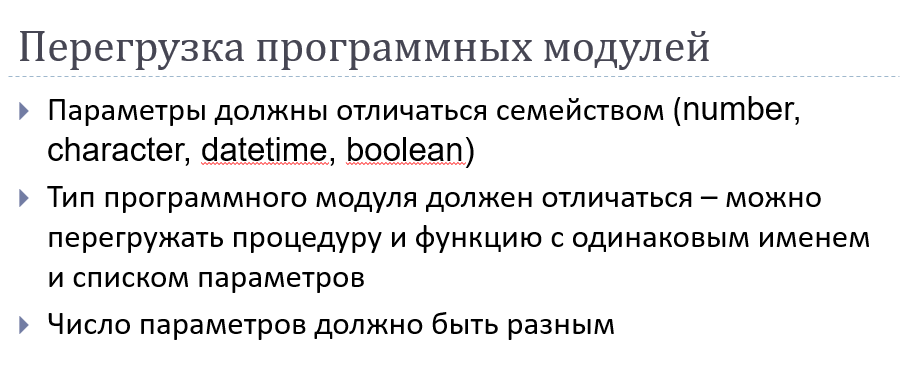
1. Локальные процедуры и функции языка PL/SQL.

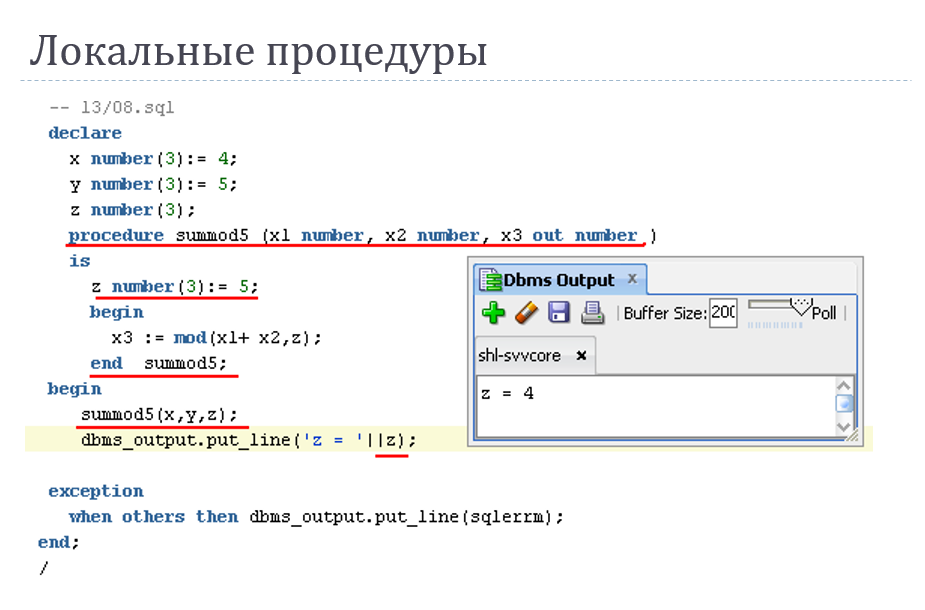
**Локальный** (или **вложенный**) **модуль** — **это**[**процедура**](https://oracle-patches.com/db/sql/3611-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D1%8B-pl-sql-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BD%D0%B0-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85)**или**[**функция PL/SQL**](https://oracle-patches.com/db/sql/3612-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8-pl-sql-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BD%D0%B0-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85)**, определяемая в разделе объявлений блока PL/SQL (анонимного или именованного).** Локальным такой модуль называется из-за того, **что он определяется только внутри родительского блока PL/SQL и не может быть вызван из другого блока, определенного вне родительского.**

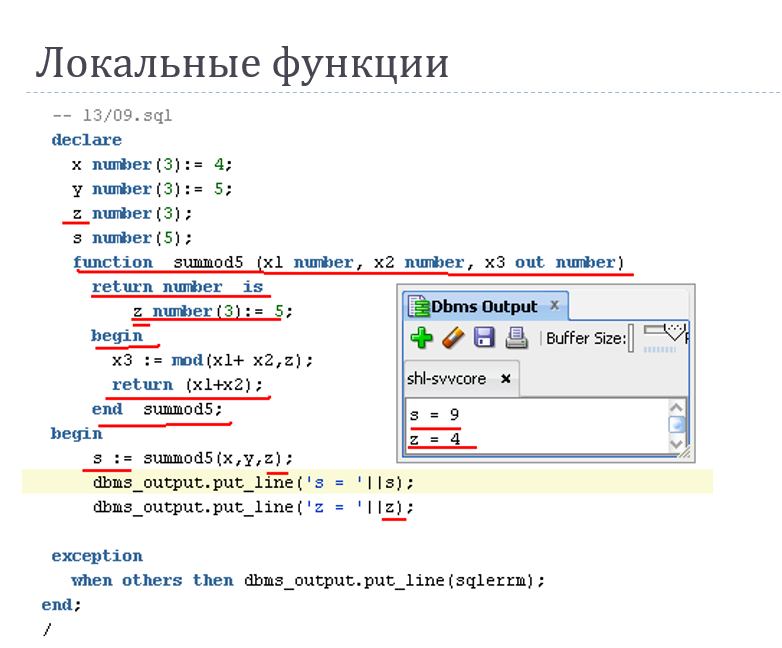
**- Определены в секции декларации pl/sql блока**

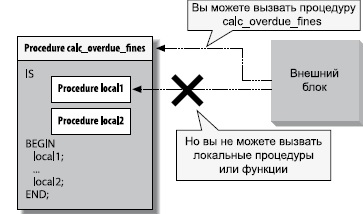
**- Их объявление добавляется в конце секции декларакции ПОСЛЕ всех типов записей, курсоров, переменных, исключений**

**- Могут быть использованы ТОЛЬКО в рамках блока, где они объявлены**









* **Уменьшение объема кода за счет исключения повторяющихся фрагментов**.
* **Улучшение удобочитаемости кода**. Даже если модуль не содержит повторяющихся блоков кода, взаимосвязанные операторы лучше поместить в локальный модуль. В этом случае вам будет легче отслеживать логику родительского модуля.

1. Использование записей в PL/SQL. Вложенные записи.

**Запись (record) – структура данных, составленная из нескольких частей информации, называемых полями**

**Для объявления записи вначале надо определить как тип, а потом объявить переменную типа «запись»**

Типы записей:

* + **Табличные**
  + **Курсорные**
  + **Программно-определенные (определяются программистом)**

Каждая строка таблицы состоит из одного или нескольких столбцов, которые могут содержать данные разных типов. **Аналогичным образом запись состоит из одного или нескольких полей.**

**На основе таблицы:**

declare one\_book books%ROWTYPE;

**На основе курсора:**

declare cursor mycur is select \* from books;

one\_sf\_book mycur%ROWTYPE;





Определяемая программистом:

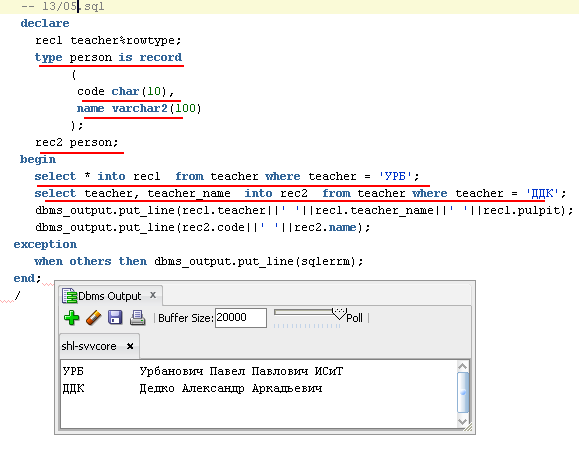
declare **type** book\_rec **record**

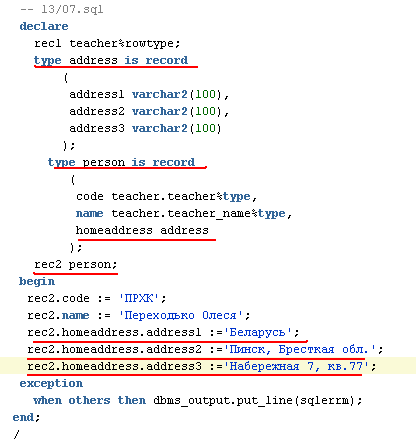
(author books.author%type, …); - перечисляем поля

Запись (record) **представляет собой составную структуру данных**; другими словами, **запись состоит из нескольких полей, каждое из которых обладает собственным значением.**

Записи очень похожи на строки в таблицах баз данных.

**Запись как целое не имеет собственного значения; однако значение имеет каждый ее компонент, или поле, а объединение их в единую запись позволяет хранить и обрабатывать все значения как одно целое.**





**Вложенные записи - одно из полей внешней записи в действительности является полем другой записи.**

1. Операторы управления, операторы цикла языка PL/SQL.

Условный оператор if

1. if – then – end if;
2. if – then – else – end if;
3. if – then – elsif – end if;
4. if – then – elsif – else – end if;

Условный оператор case

1. case – when – then – end;
2. case – when – then – else – end;

**case x**

**when … then …;**

**when … then …;**

**when x between 13 and 20 then ….;**

**else ….;**

**end case;**

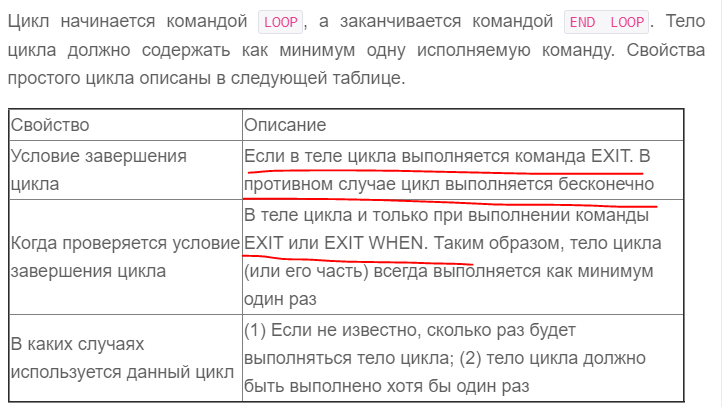
Отличие: в 1 выбираем значение и сравниваем его с чем-то. Во 2 – проверяем условия

1. **Простейший цикл**

loop

исполняемые\_команды

end loop;



Выход из цикла exit или exit when

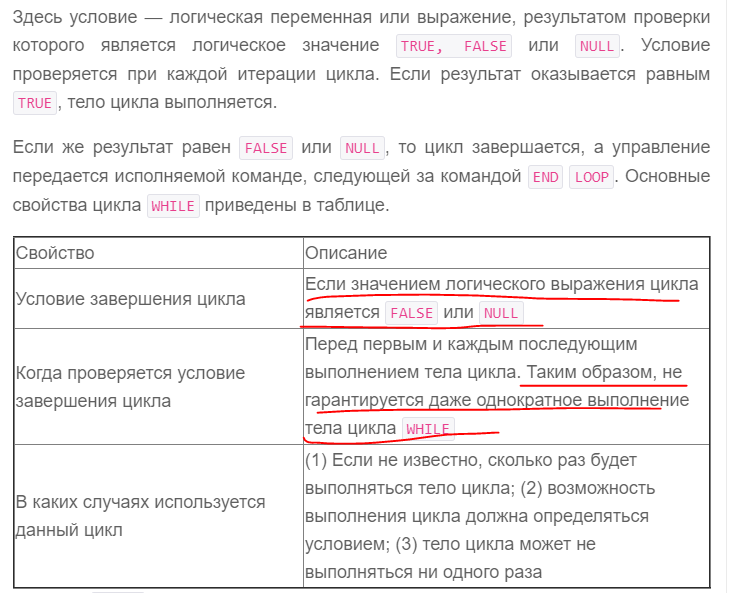
1. **Цикл while**

WHILE условие

LOOP

исполняемые\_команды

END LOOP;



1. **Цикл for**

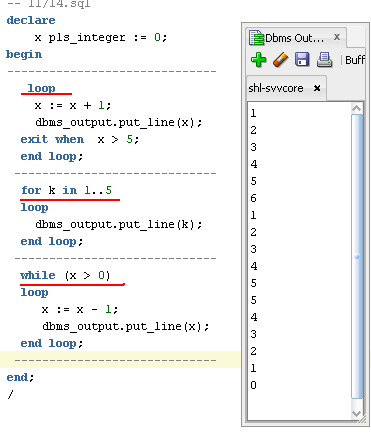
FOR **счетчик** **IN [REVERSE] начальное\_значение .. конечное\_значение**

LOOP

исполняемые\_команды

END LOOP;

* **Не объявляйте счетчик цикла**. **PL/SQL автоматически неявно объявляет локальную переменную с типом данных INTEGER. Область действия этой переменной совпадает с границей цикла; обращаться к счетчику за пределами цикла нельзя.**



1. Курсоры. Виды курсоров. Схемы обработки курсора.

**Курсор** — **объект БД, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных Oracle. Позволяет проходить по результатам select-запроса запись за записью. ЭТО УКАЗАТЕЛЬ НА ОБЛАСТЬ PGA, В КОТОРОЙ ХРАНЯТСЯ СТРОКИ ЗАПРОСА, ЧИСЛО СТРОК, УКАЗАТЕЛЬ НА РАЗОБРАННЫЙ ЗАПРОС В ОБЩЕМ ПУЛЕ**

**Курсор Oracle – указатель на область в PGA, в которой хранится:**

* + **1)строки запроса,**
  + **2)число строк,**
  + **3)указатель на разобранный запрос в общем пуле.**

**Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.**

Поддерживаются два типа курсоров:

* + **явный — объявляется разработчиком**;
  + **неявный — не требует объявления**

**Курсор может возвращать одну строку, несколько строк или ни одной строки**

Для **повторного создания результирующего набора для других значений параметров курсор следует закрыть, а затем повторно открыть**

**Точная выборка: возвращается точно 1 строка**, в PL/SQL select into

Присваивается локальной переменной в блоке данных. Это способ получить строку в переменную

**Неточная выборка: возвращает select into неопределенное количество строк, когда возвращает более одной строки – исключение TOO\_MANY\_ROWS; нет строк вообще – NO\_DATA\_FOUND;**

* **DECLARE** — **выполняет объявление явного курсора**.
* **OPEN** — **открывает курсор, создавая новый результирующий набор на базе указанного запроса**.
* **FETCH** — **выполняет последовательное извлечение строк из результирующего набора от начала до конца.**
* **CLOSE — закрывает курсор и освобождает занимаемые им ресурсы**.
* **Неявные курсоры** **- простой оператор SELECT ... INTO извлекает одну строку данных непосредственно в переменные локальной программы.** Это удобный (и часто наиболее эффективный) способ доступа к данным, использование которого, однако, может приводить к необходимости повторного кодирования оператора SELECT (или похожих операторов) в нескольких местах программы.

Ошибки неявного курсора:

- no\_data\_found – не возвращает строк вообще

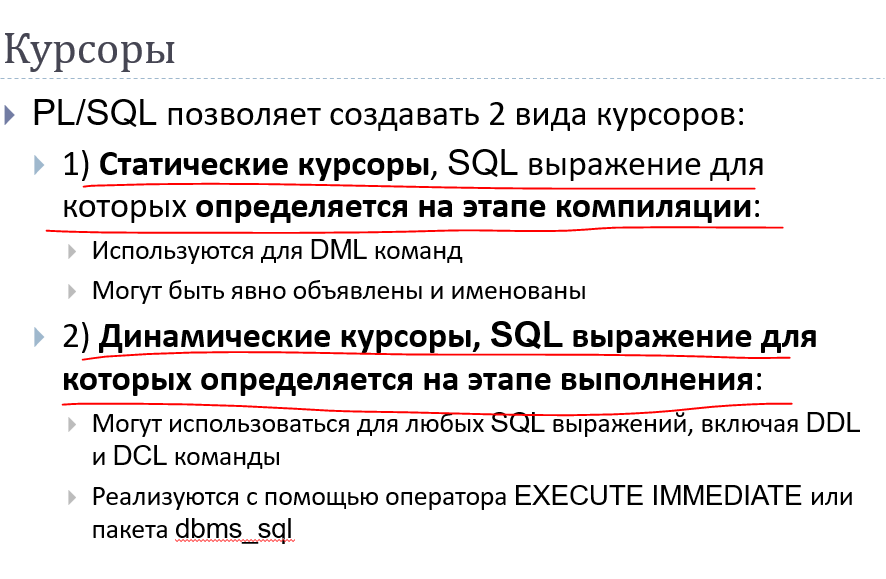
- too\_many\_rows – более 1 строки вернул select … into

**select into чтобы вернуть ровно 1 строку – точную выборку!**

* **Явные курсоры** - **вы можете явно объявить курсов в разделе объявлений (локального блока или пакета). В этом случае курсор можно будет открывать и извлекать данные в одной или нескольких программах, причем возможности контроля будут шире, чем при использовании неявных курсоров. ОБЪЯВЛЯЕТСЯ РАЗРАБОТЧИКОМ! Управление – declare, open, fetch, close**

**CURSOR employee\_cur IS SELECT \* FROM employee;**

* ***Статический – выражение определяется при компиляции***
* ***Динамический – выражение определяется при выполнении***



**Неявные курсор – выполнение SQL выражения в секции исполнения или в секции исключений блока**

Операторы **INSERT, UPDATE, DELETE, MERGE, SELECT INTO**

* **Не требуют объявления**
* **Не требуют OPEN, FETCH, CLOSE**
* **Всякий раз, когда выполняется оператор DML (INSERT, UPDATE и DELETE), с этим оператором связывается неявный курсор.** Для операций INSERT курсор содержит данные, которые необходимо вставить. Для операций UPDATE и DELETE курсор определяет строки, которые будут затронуты.

Когда неявный курсор не возвращает строк вообще, PL/SQL **генерирует исключение NO\_DATA\_FOUND** и передает управление в секцию исключений

**Когда SELECT INTO возвращает более одной строки, PL/SQL генерирует исключение TOO\_MANY\_ROWS** и также передает управление в секцию исключений

**SELECT INTO предназначен исключительно для того, чтобы возвращать ровно 1 строку – точную выборку**

1. Курсоры. Атрибуты курсора. Курсоры с параметрами.

**Курсор** — **объект БД, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных Oracle. Позволяет проходить по результатам select-запроса запись за записью. ЭТО УКАЗАТЕЛЬ НА ОБЛАСТЬ PGA, В КОТОРОЙ ХРАНЯТСЯ СТРОКИ ЗАПРОСА, ЧИСЛО СТРОК, УКАЗАТЕЛЬ НА РАЗОБРАННЫЙ ЗАПРОС В ОБЩЕМ ПУЛЕ**

**Курсор Oracle – указатель на область в PGA, в которой хранится:**

* + **1)строки запроса,**
  + **2)число строк,**
  + **3)указатель на разобранный запрос в общем пуле.**

**Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.**

Атрибут курсора – хранит информацию о состоянии курсора и результате его работы

|  |  |
| --- | --- |
| НЕЯВНЫЙ КУРСОР | ЯВНЫЙ КУРСОР |
| sql%**found** – null перед выполнением, true если были манипуляции **sql%isopen ВСЕГДА FALSE ДЛЯ НЕЯВНЫХ** – он сразу открывается и закрывается sql%**notfound** – null перед выполнением, false если были манипуляции sql%**rowcount** – количество строк, выбранных в курсоре | Переменная\_курсора%**found** – true успешно выбрана, false не выбрана Переменная\_курсора%**isopen** – true открыт, false закрыт Переменная\_курсора%**notfound** – false выбрана, true не выбрана Переменная\_курсора%**rowcount** – количество строк, выбранных в курсоре |

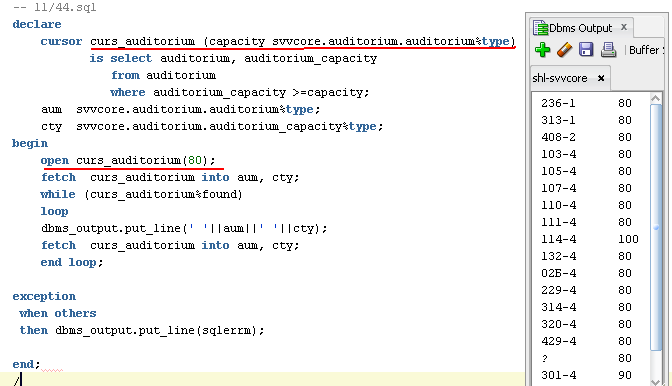
**При объявлении курсора в скобках указывается имя параметра**

**declare**

**cursor curs\_audiotorium (capacity number) is select \* from auditorium where auditorium\_capacity > capacity (capacity – ПАРАМЕТР)**

**open curs\_auditorium(80); --передали параметр**

**fetch curs\_auditorium into aum, ….**



1. Курсоры. Курсорные переменные. Параметры инстанса, связанные с курсорами.

**Курсор** — **объект БД, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных Oracle. Позволяет проходить по результатам select-запроса запись за записью. ЭТО УКАЗАТЕЛЬ НА ОБЛАСТЬ PGA, В КОТОРОЙ ХРАНЯТСЯ СТРОКИ ЗАПРОСА, ЧИСЛО СТРОК, УКАЗАТЕЛЬ НА РАЗОБРАННЫЙ ЗАПРОС В ОБЩЕМ ПУЛЕ**

**Курсор Oracle – указатель на область в PGA, в которой хранится:**

* + **1)строки запроса,**
  + **2)число строк,**
  + **3)указатель на разобранный запрос в общем пуле.**

**Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.**

Курсорная переменная – структура данных, которая указывает на курсорный объект. Используется для передачи курсора в качестве параметра или чтобы отложить связь курсора с select до выполнения open

Объявленная нами курсорная переменная указывает на объект курсора в базе данных

declare

**type имя\_курсорного\_типа is ref cursor;**

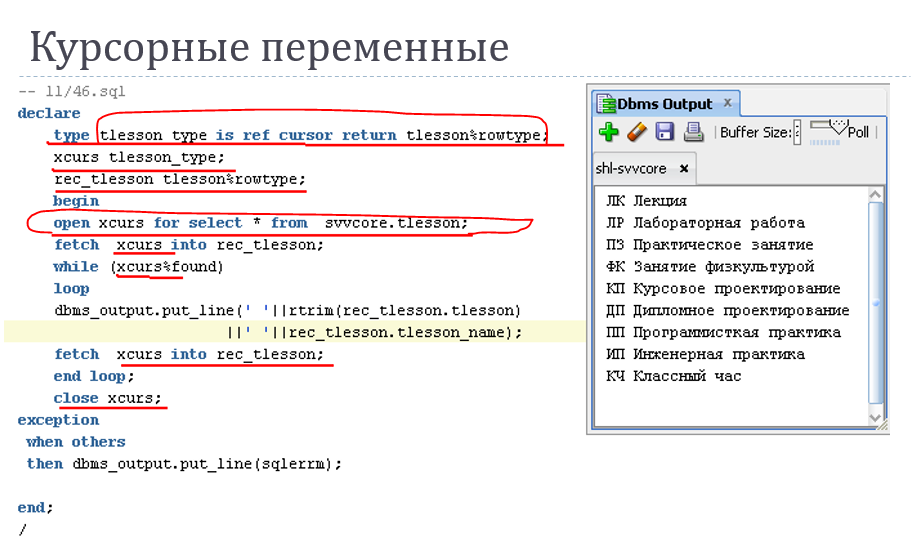
begin

**open имя\_курсорного\_типа for …. – и** **работаем как с обычным курсором**

end;

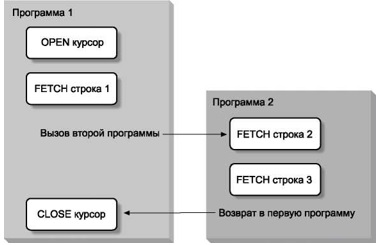
**TYPE имя\_типа\_курсора IS REF CURSOR [ RETURN возвращаемый\_тип];**

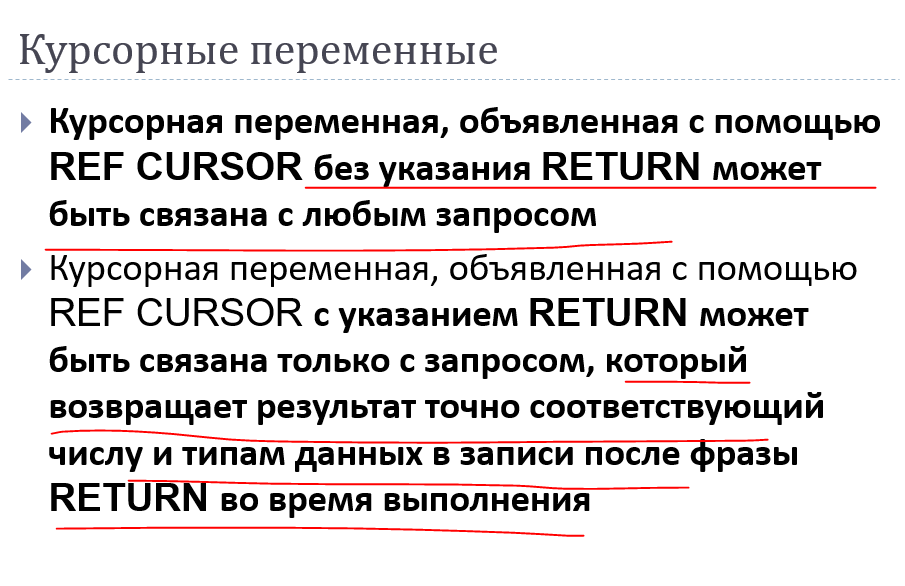
Здесь имя курсора — имя типа курсора, а ***возвращаемый тип — спецификация воз­вращаемых данных курсора. Это может быть любая структура данных, использование которой допускается в секции RETURN, определенная с помощью атрибута %ROWTYPE или путем ссылки на ранее объявленный тип записи.***

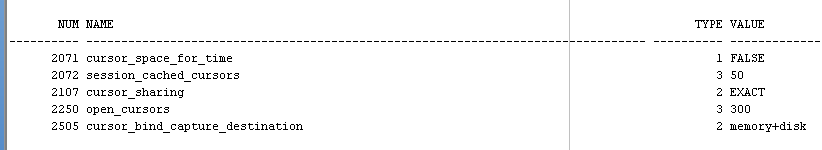


Курсорная переменная ссылается на курсор. **В отличие от**[**явного курсора**](https://oracle-patches.com/db/sql/3565-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D1%8F%D0%B2%D0%BD%D1%8B%D0%BC%D0%B8-%D0%BA%D1%83%D1%80%D1%81%D0%BE%D1%80%D0%B0%D0%BC%D0%B8-%D0%B2-pl-sql)**, имя которого в PL/SQL используется как идентификатор рабочей области результирующего набора строк, курсорная переменная содержит ссылку на эту рабочую область.**

Явные и [неявные курсоры](https://oracle-patches.com/db/sql/3562-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%BD%D0%B5%D1%8F%D0%B2%D0%BD%D1%8B%D0%BC%D0%B8-%D0%BA%D1%83%D1%80%D1%81%D0%BE%D1%80%D0%B0%D0%BC%D0%B8-%D0%B2-pl-sql) имеют статическую природу, поскольку они жестко привязаны к конкретным запросам. **С помощью же курсорной переменной можно выполнить любой запрос и даже несколько разных запросов в одной программе.**







select \* from v$parameter where NAME like '%cursor%';

Параметр Cursor Space for time используется для сохранения SQL до тех пор, пока его курсор не будет закрыт.

CURSOR\_SHARING определяет, какие операторы SQL могут совместно использовать одни и те же курсоры.

* FORCE

Позволяет создать новый курсор при совместном использовании существующего курсора или если план курсора не оптимален.

* EXACT

Только операторы с идентичным текстом могут использовать один и тот же курсор.

CURSOR\_BIND\_CAPTURE\_DESTINATION определяет расположение, в котором доступны переменные связывания, захваченные из курсоров SQL.

В **Oracle9i появился предопределенный слабый тип REF CURSOR с именем SYS\_REFCURSOR. Теперь программисту не нужно определять собственный слабый тип — достаточно ис­пользовать стандартный тип Oracle:**

**DECLARE**

***my\_cursor SYS\_REFCURSOR;***

1. Курсоры. Курсорные подзапросы.

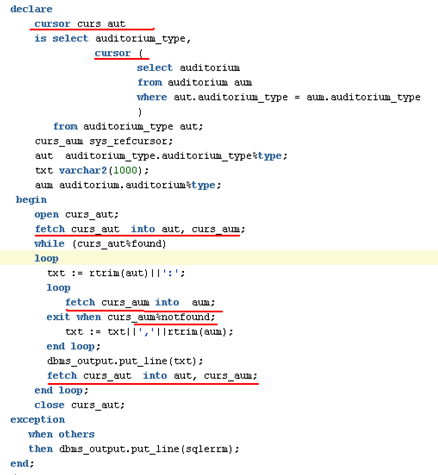
**Курсор** — **объект БД, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных Oracle. Позволяет проходить по результатам select-запроса запись за записью. ЭТО УКАЗАТЕЛЬ НА ОБЛАСТЬ PGA, В КОТОРОЙ ХРАНЯТСЯ СТРОКИ ЗАПРОСА, ЧИСЛО СТРОК, УКАЗАТЕЛЬ НА РАЗОБРАННЫЙ ЗАПРОС В ОБЩЕМ ПУЛЕ**

**Курсор Oracle – указатель на область в PGA, в которой хранится:**

* + **1)строки запроса,**
  + **2)число строк,**
  + **3)указатель на разобранный запрос в общем пуле.**

**Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.**

**Курсорные подзапросы или вложенные курсорные выражения – механизм встраивания курсоров в операторы SQL.**



Мы объявляем курсор в рамках другого курсора.

Declare

**cursor curs\_aut**

**is select auditorium\_type,**

**cursor** *(select auditorium from auditorium aum where aut.auditorium\_type=aum.auditorium\_type)*

**from auditorium\_type aut;**

**curs\_aum sys\_refcursor;**

begin

open curs\_aut;

**fetch curs\_aut into aut, curs\_aum;**

while curs\_aut%found

loop

loop

**fetch curs\_aum into …**

**exit when curs\_aum%notfound; ВЛОЖЕННЫЙ КУРСОР**

end loop;

**fetch curs\_aut into aut, curs\_aum; ОСНОВНОЙ КУРСОР**

end loop;

end;

Курсорное выражение (подзапрос) – это выражение со специальным оператором cursor, используется в SQL-запросе и определяет вложенный курсор. **ТАКИЕ НАБОРЫ ОБРАБАТЫВАЮТСЯ ВО ВЛОЖЕННЫХ ЦИКЛАХ! Главный цикл выбирает строки основного курсора, вложенный – строки вложенного курсора.**

1. Курсоры. Использование конструкции CURRENT OF в курсорах.

**Курсор** — **объект БД, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных Oracle. Позволяет проходить по результатам select-запроса запись за записью. ЭТО УКАЗАТЕЛЬ НА ОБЛАСТЬ PGA, В КОТОРОЙ ХРАНЯТСЯ СТРОКИ ЗАПРОСА, ЧИСЛО СТРОК, УКАЗАТЕЛЬ НА РАЗОБРАННЫЙ ЗАПРОС В ОБЩЕМ ПУЛЕ**

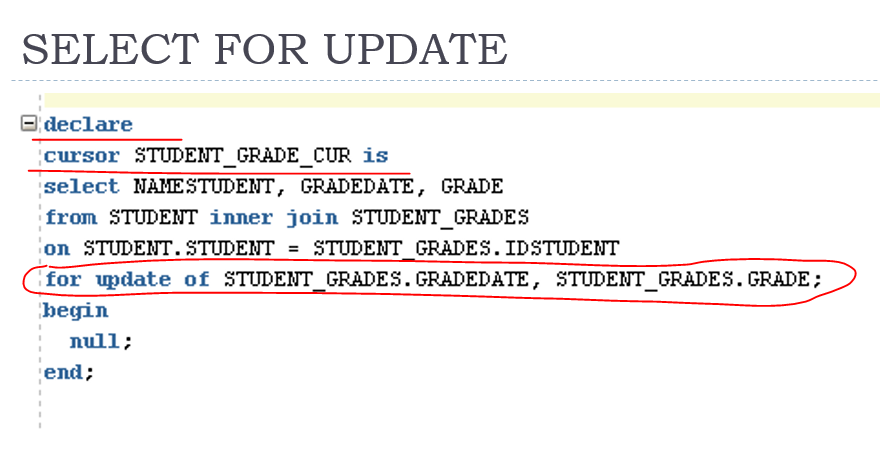
**Курсор Oracle – указатель на область в PGA, в которой хранится:**

* + **1)строки запроса,**
  + **2)число строк,**
  + **3)указатель на разобранный запрос в общем пуле.**

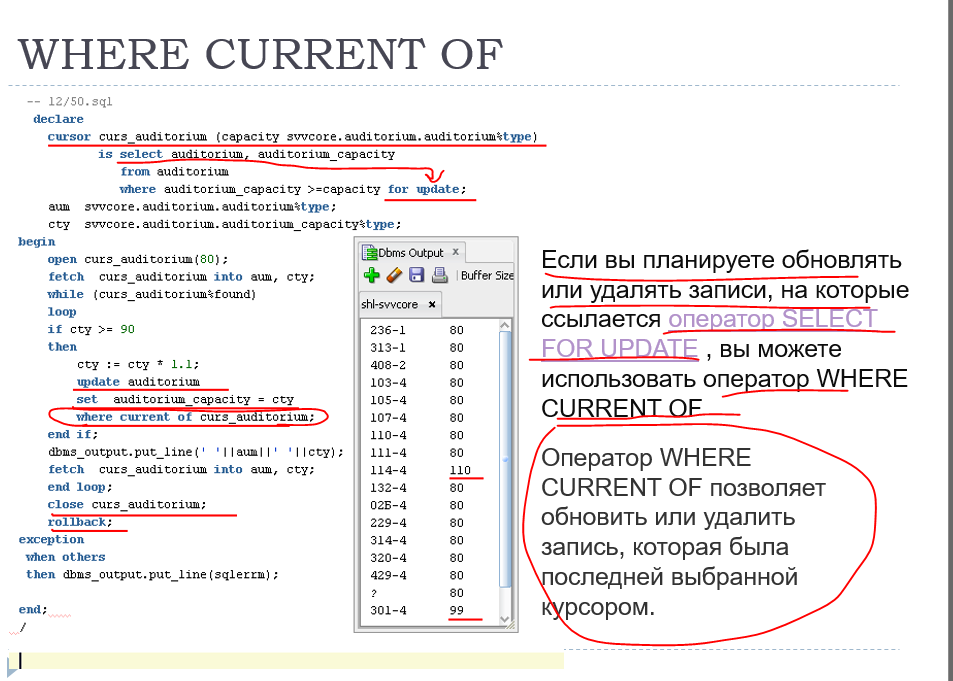
**Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.**

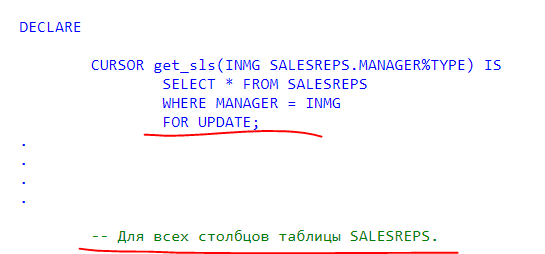
**Оператор WHERE CURRENT OF позволяет обновить или удалить запись, которая была в курсоре последней.**

**Иногда, при выборке из курсора бывает ситуация, что какой-либо столбец или строки результирующего набора необходимо обновить**. То есть, изменить их содержимое. **Для того, чтобы это осуществить, непосредственно при объявлении курсора необходимо использовать конструкцию - FOR UPDATE (для обновления ..).** **А, так же конструкцию, WHERE CURRENT OF (где текущая строка ..) в операторах UPDATE, DELETE**. Собственно конструкция FOR UPDATE, является частью оператора SELECT и объявляется последней









***Так как обычный запрос с помощью оператора SELECT, при выполнении получает строки таблицы и при этом сама таблица выборки не блокируется***, то есть ***любой другой пользователь может выполнить запрос к той же таблице, получив при этом данные.***

В Oracle при выполнении запроса, т.е. при извлечении активного набора SELECT, **производится моментальный снимок таблицы (snapshot), при этом все изменения, сделанные до этого момента кем-либо еще отражаются в данном наборе.** А**, после того как snapshot получен все изменения, произведенные в данной таблице выборке, даже если они зафиксированы оператором COMMIT, отражаться не будут!!!** Для того, чтобы их отразить нужно закрыть и снова открыть курсор, загрузив данные заново!

Это и есть алгоритм согласованного чтения данных. **А вот когда мы объявляем FOR UPDATE - строки активного набора данных блокируются до момента выполнения COMMIT**. Таким образом **мы запрещаем изменение данных другим сеансам.**

1. Курсоры. Динамические курсоры.

**Курсор** — **объект БД, который позволяет приложениям работать с записями построчно. Курсор – это средство извлечения данных из базы данных Oracle. Позволяет проходить по результатам select-запроса запись за записью. ЭТО УКАЗАТЕЛЬ НА ОБЛАСТЬ PGA, В КОТОРОЙ ХРАНЯТСЯ СТРОКИ ЗАПРОСА, ЧИСЛО СТРОК, УКАЗАТЕЛЬ НА РАЗОБРАННЫЙ ЗАПРОС В ОБЩЕМ ПУЛЕ**

**Курсор Oracle – указатель на область в PGA, в которой хранится:**

* + **1)строки запроса,**
  + **2)число строк,**
  + **3)указатель на разобранный запрос в общем пуле.**

**Открытие курсора – создание контекстной области PGA – создается моментальный снимок (snapshot) данных запроса.**

**EXECUTE IMMEDIATE - однострочные запросы и DDL команды**

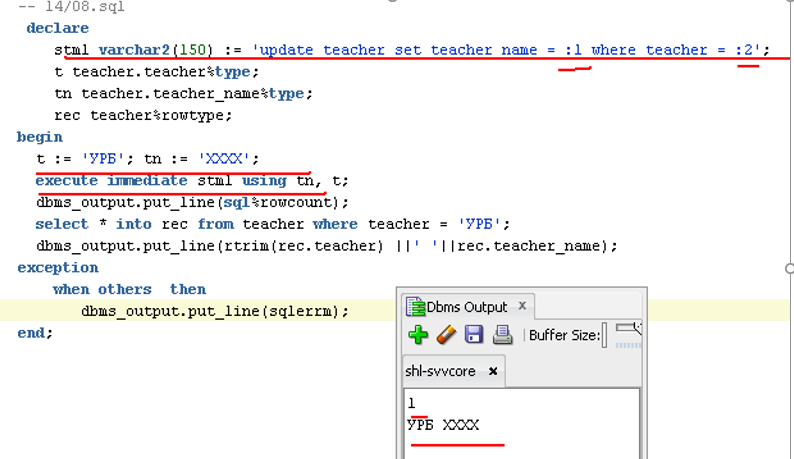
**OPEN FOR, FETCH и CLOSE - динамические многострочные запросы**

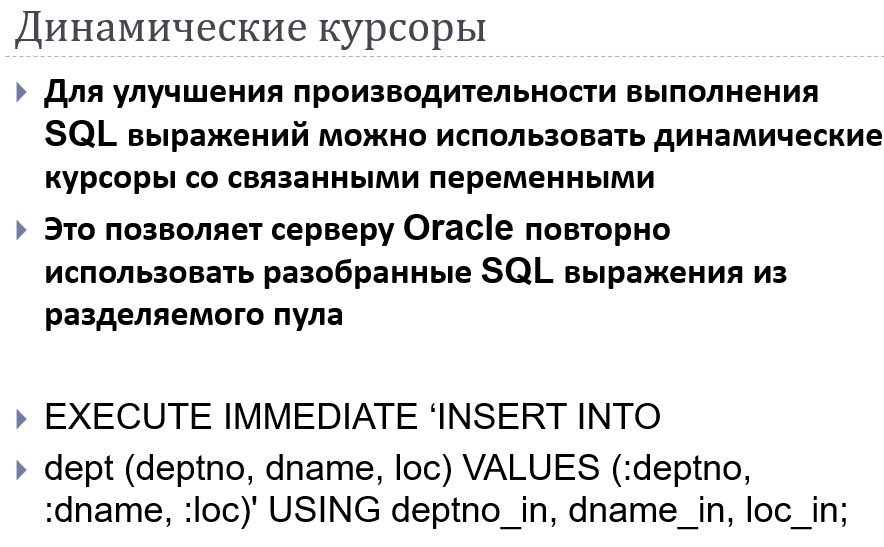
Синтаксис EXECUTE IMMEDIATE:

EXECUTE IMMEDIATE sql\_statement

[INTO {variable [,variable ...] | record}] [USING [IN | OUT | IN OUT]

bind\_argument [, [IN | OUT | IN OUT] bind\_argument . ..] ] [{RETURNING | RETURN} INTO bind\_argument [,bind\_argument]...];





Оператор EXECUTE IMMEDIATE выполняет динамический оператор SQL или анонимный блок PL/SQL.

**Термином «динамический SQL» обозначаются команды SQL, которые конструируются и вызываются непосредственно во время выполнения программы. Статическими называются жестко закодированные команды SQL, которые не изменяются с момента компиляции программы**. **«Динамическим PL/SQL» называют целые блоки кода PL/SQL, которые строятся динамически, а затем компилируются и выполняются.**

BEGIN

***EXECUTE IMMEDIATE 'CREATE INDEX emp\_u\_1 ON employees (last\_name)';***

END;

1. Применение псевдостолбцов ROWID, ROWNUM в PL/SQL.

Псевдостолбцами в Oracle принято ***называть столбцы, которые отсут­ствуют в таблицах в явном виде, но могут быть использованы в запросах.***

Наиболее употребимым и важным из них является **ROWID — псевдостол­бец, являющийся уникальным идентификатором строки. Он не просто га­рантированно уникален в рамках таблицы — более того: он уникален в рамках базы данных.** С физической точки зрения ROWID является свое­образной **координатой записи в базе**.

**ROWID — это псевдостолбец, который является уникальным идентификатором строки в таблице и фактически описывает точное физическое расположение данной конкретной строки.** На основе этой информации Oracle впоследствии может найти данные, связанные со строкой таблицы.

Идентифицирует строку однозначно, имеет двоичное значение.

**1-7 сегмент (объект)**

**8-10 файл**

**11-16 блок**

**17-18 адрес строки в блоке**

**ПОИСК ПО НЕМУ САМЫЙ БЫСТРЫЙ!!!!!!**

**6-3-6-3**

Ускоряет обращение. Он ссылается непосредственно на строку таблицы. Это «псевдостолбец» - при выборке его фактически не существует

 Для хранения данных **ROWID** требуется 80 бит (10 байт). Идентификаторы **ROWID** состоят из четырех компонентов: **номера объекта (32 бита), относительного номера файла (10 бит), номера блока (22 бита) и номера строки (16 бит).** Эти идентификаторы отображаются как 18-символьные последовательности, указывающие местонахождение данных в БД, **причем каждый символ представлен в формате base-64, состоящем из символов A-Z, a-z, 0-9, + и /.** Первые шесть символов – это номер объекта данных, следующие три – относительный номер файла, следующие шесть – номер блока, последние три – номер строки.

**Псевдостолбец ROWNUM возвращает номер текущей строки запроса**. **Логический номер записи в запросе**. **Возвращает число, представляющее порядок, в котором Oracle выбирает строку из таблицы.** 1 строка ROWNUM = 1

Позволяет вводить ограничение на количество выводимых записей

**select \* from students where rownum < 10;**

Для справочки – какие еще могут быть псевдостолбцы?

CURRVAL и NEXTVAL

***Псевдостолбцы CURRVAL (текущее значение) и NEXTVAL (следующее значение) применяются в последовательностях***.

Последовательность (sequ­ence) — это объект Oracle, который используется для генерирования уникальных чисел. Последовательность создается с помощью DDL-команды CREATE SEQUENCE. После того как последовательность создана, к ней можно обратиться.

**select ИМЯ\_ПОСЛЕД.CURRVAL from dual;**

**select ИМЯ\_ПОСЛЕД.NEXTVAL from dual;**

1. Обработка исключений в PL/SQL, стандартные исключения, генерация и обработка исключения.

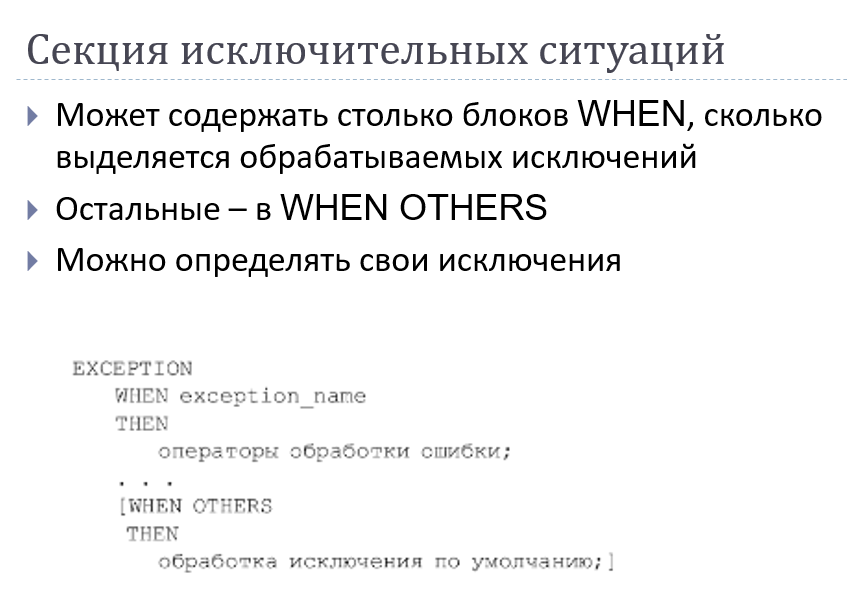
**Исключительная ситуация – событие, возникающее в программе и требующее незамедлительной обработки.**

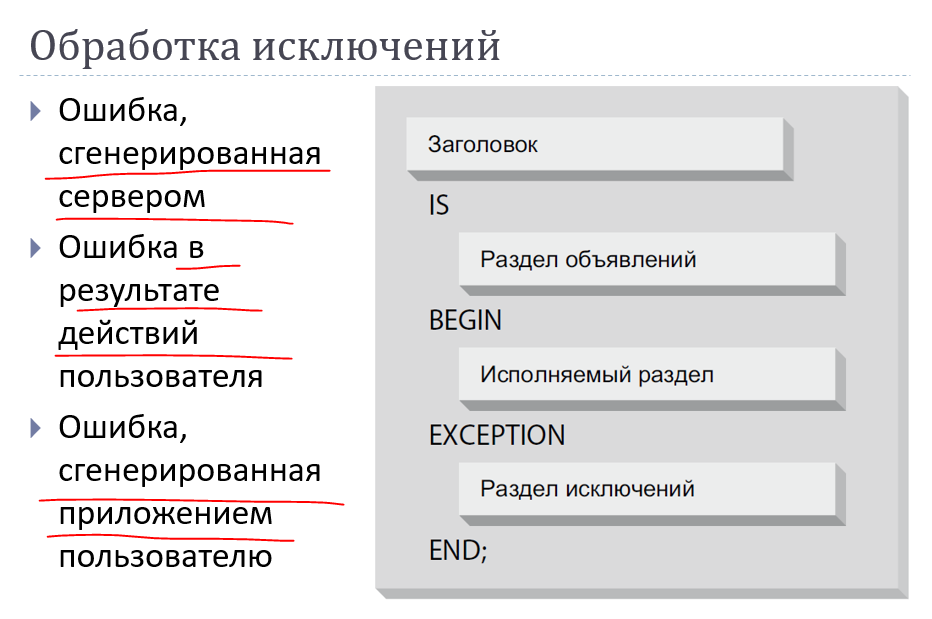
1. **Программно-определяемые**
2. **Предопределенные (стандартные**)

**Sqlerrm – функция возвращает сообщение об ошибке**

**Sqlcode – функция возвращает номер ошибки**

Обработка исключений – перехват ошибки в секции exception





**Секция исключений exception – необязательная секция в PL/SQL блоке, которая содержит один или несколько обработчиков исключений**

RAISE (RAISE\_APPLICATION\_ERROR) – команда, которая прерывает выполнение текущего блока

**Обработка исключений – перехват ошибки в секции исключений**

**Область действия – часть кода, в рамках которого может быть сгенерировано исключение**

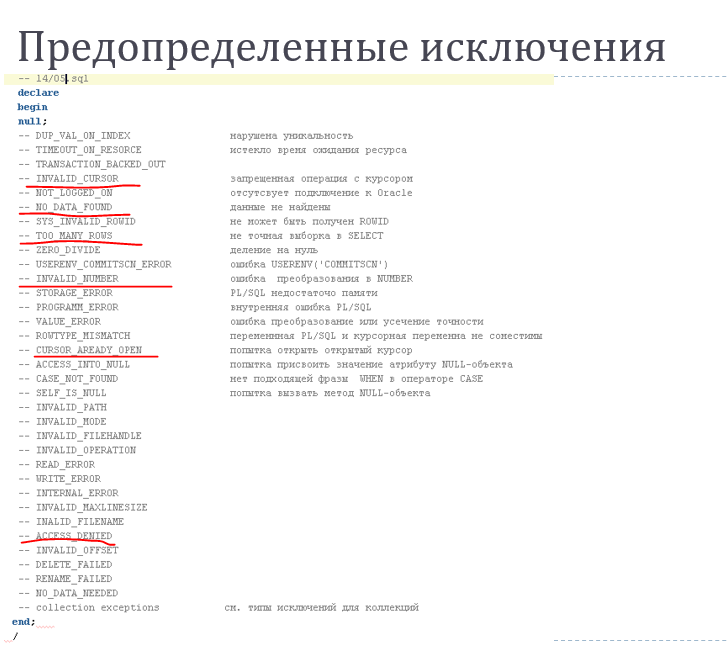
**Распространение исключения – процесс передачи исключений от одного блока другому, если исключение не было обработано ЭСКАЛАЦИЯ!**

* Необработанное исключение – исключение становится необработанным, **если оно не обработано блоком самого верхнего уровня**
* **Неименованное исключение – исключение, которое имеет код ошибки и сообщение, но не имеет наименования**, не может быть использовано в команде RAISE или в секции WHEN
* **Именованное исключение – исключение, которому было определено наименование**

имя\_исключения EXCEPTION;

RAISE имя\_исключения;

 WHEN имя\_исключения THEN



**raise\_application\_error(**

**error\_number,**

**message**

**[, {TRUE | FALSE}]**

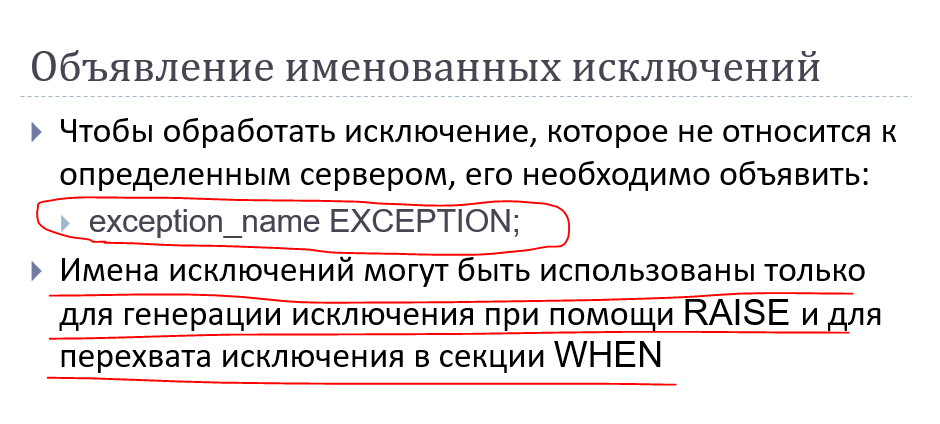
**);**

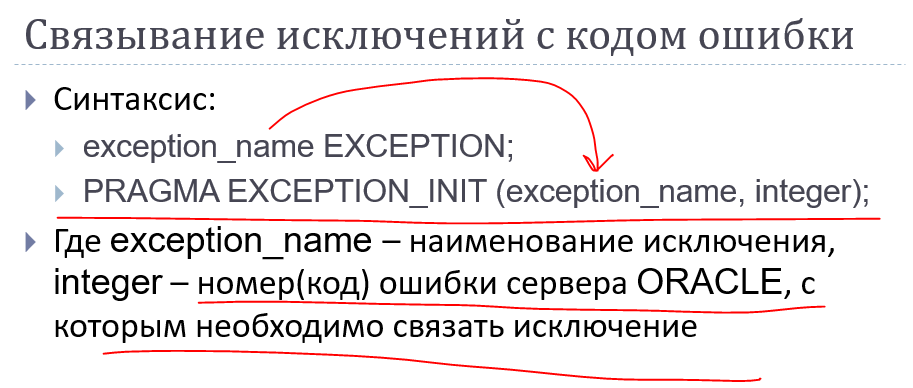
Существует два типа исключений:

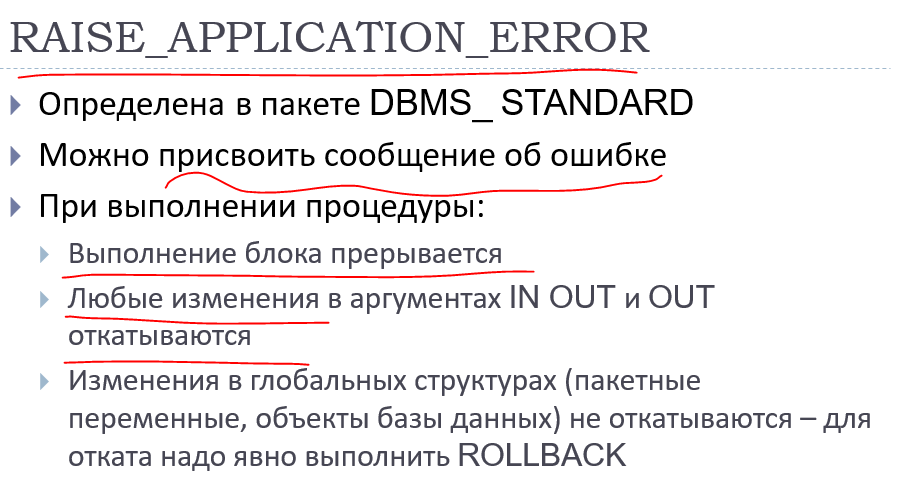
* **Системное исключение определяется в Oracle и обычно инициируется исполняемым**[**ядром PL/SQL**](https://oracle-patches.com/db/sql/3231-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0-pl-sql-%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%B2%D0%BE%D0%B5-%D1%81%D0%BB%D0%BE%D0%B2%D0%BE-pragma)**, обнаружившим ошибку. Одним системным исключениям присваиваются имена (например, NO\_DATA\_FOUND), другие ограничиваются номерами и описаниями.**
* Исключение, определяемое программистом, актуально только для конкретного приложения.

Имя исключения можно связать с конкретной ошибкой Oracle с помощью директивы компилятора EXCEPTION\_INIT или же назначить ошибке номер и описание процедурой RAISE\_APPLICATION\_ERROR.

**Инициировать исключение с номером ошибки (в диапазоне от –20 999 до –20 000) может и разработчик приложения, воспользовавшись для этой цели процедурой RAISE\_APPLICATION\_ERROR**







1. Принцип распространения исключений в PL/SQL. Инструкция RAISE\_APPLICATION\_ERROR.

Существует два типа исключений:

* **Системное исключение определяется в Oracle и обычно инициируется исполняемым**[**ядром PL/SQL**](https://oracle-patches.com/db/sql/3231-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0-pl-sql-%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%B2%D0%BE%D0%B5-%D1%81%D0%BB%D0%BE%D0%B2%D0%BE-pragma)**, обнаружившим ошибку. Одним системным исключениям присваиваются имена (например, NO\_DATA\_FOUND), другие ограничиваются номерами и описаниями.**
* **Исключение, определяемое программистом, актуально только для конкретного приложения.**

Имя исключения можно связать с конкретной ошибкой Oracle с помощью директивы компилятора EXCEPTION\_INIT или же назначить ошибке номер и описание процедурой RAISE\_APPLICATION\_ERROR.

**Инициировать исключение с номером ошибки (в диапазоне от –20 999 до –20 000) может и разработчик приложения, воспользовавшись для этой цели процедурой RAISE\_APPLICATION\_ERROR**

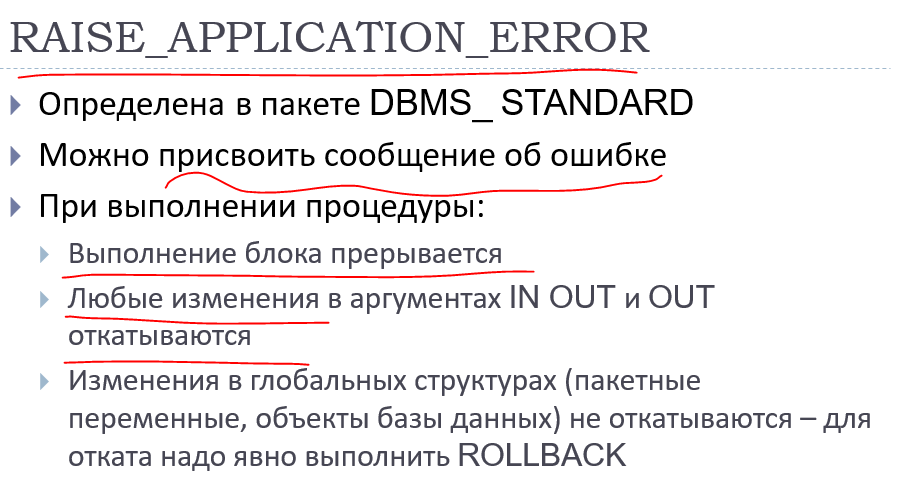
**raise\_application\_error(**

**error\_number,**

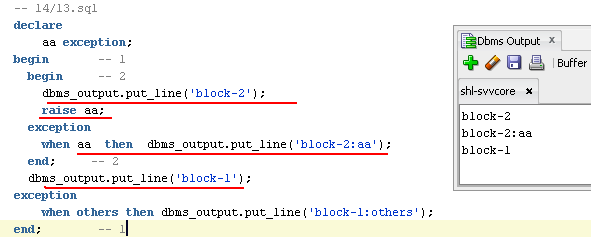
**message**

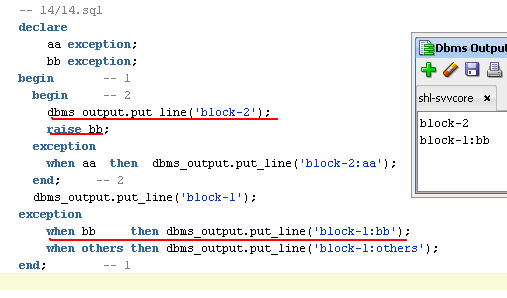
**[, {TRUE | FALSE}]**

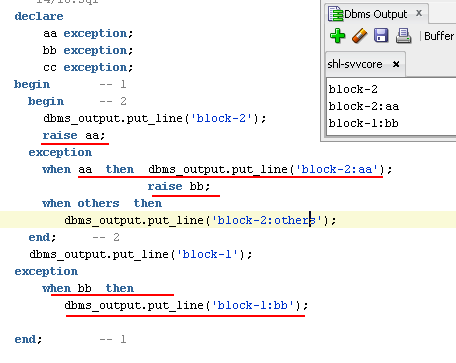
**);**



**Распространение исключений – процесс передачи исключения от одного блока другому, если исключение не было обработано! С помощью raise**







1. Встроенные функции языка PL/SQL. Функции работы с датами, текстом и числами.

**TO\_DATE( string1, [ format\_mask ], [ nls\_language ] ) приводит к дате**

SELECT TO\_DATE('2019/07/22', 'yyyy/mm/dd') FROM DUAL;

--Результат:       22.07.2019

SELECT TO\_DATE('072219', 'MMDDYY') FROM DUAL;

--Результат:       22.07.2019

SELECT TO\_DATE('20190722', 'yyyymmdd') FROM DUAL;

--Результат:       22.07.2019

SELECT TO\_DATE('30.01.2019 18:30:52', 'DD.MM.YYYY HH24:MI:SS') FROM DUAL;

--Результат:       30.01.2019 18:30:52

**TO\_CHAR( value, [ format\_mask ], [ nls\_language ] ) приводит к символьной строке**

SQL> SELECT TO\_CHAR(1242.78, '9999.9') FROM DUAL;

--Результат:    1242.8

SQL> SELECT TO\_CHAR(-1242.78, '9999.9') FROM DUAL;

--Результат:   -1242.8

SQL> SELECT TO\_CHAR(1242.78, '9,999.99') FROM DUAL;

--Результат:    1,242.78

SQL> SELECT TO\_CHAR(1242.78, '$9,999.00') FROM DUAL;

--Результат:   $1,242.78

SQL> SELECT TO\_CHAR(18, '000099') FROM DUAL;

--Результат:    000018

**Работа со строками и символами**

**select** concat('Дарья ','Шейбак') **from** dual; --Дарья Шейбак

**select** lower(‘ДАРЬЯ ШЕЙБАК’) **from** dual; --дарья шейбак

**select** upper(‘дарья шейбак’) **from** dual; --ДАРЬЯ ШЕЙБАК

**select** initcap('дарья') **from** dual; --Дарья

**select** trim(' Дарья ') **from** dual; --Дарья, вырезает пробелы с начала и с конца строки, ltrim () вырежет слева, rtrim () вырежет справа

REPLACE( исходная\_строка, исходная\_подстрока, [ новая\_подстрока ] ) --Заменяет некоторую последовательность символов в строке другим набором символов

SUBSTR( **исходная\_строка, начальная\_позиция, [ длина\_подстроки ]** )

-- Если начальная\_позиция отрицательная - то отсчет идет от конца строки

-- Если начальная\_позиция равна 0 то считается что начальная позиция равна 1

-- Если начальная\_позиция положительная - то отсчет идет от начала строки

-- Если длина\_подстроки отрицательная то функция вернет NULL

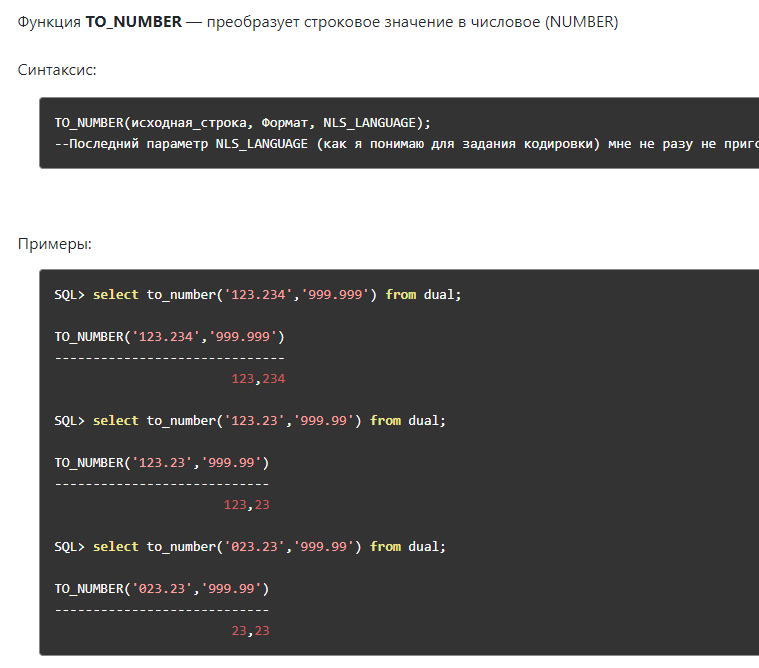
**select** substr('000111qa00zx0222333', 3, 5) **from** dual;

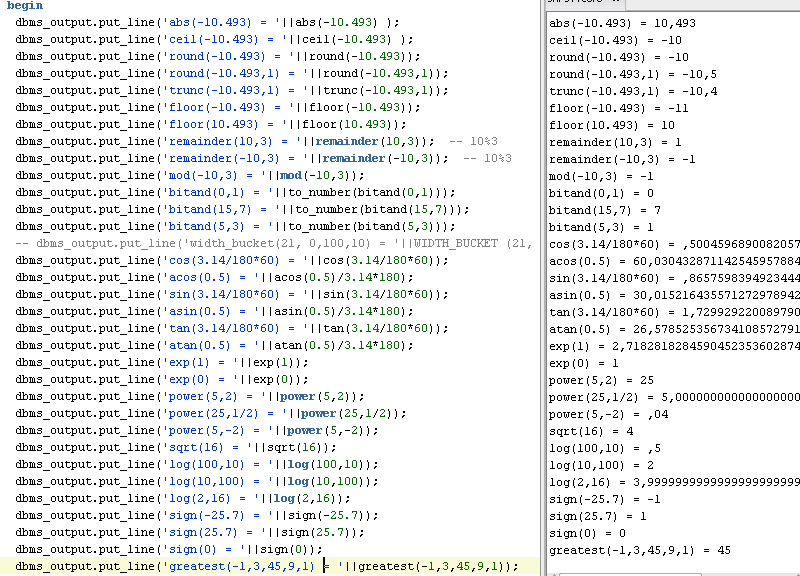
Строка первый параметр. С какой позиции – 3. Сколько символов - 5

**select** length(‘Дарья’) **from** dual; --возвращает длину строки

**VSIZE** — Возвращает длину строки в байтах

**select** ascii('2') **from** dual; --вернет код ASCII символа





**Числовые**

Abs(-10.493) будет 10.493 это взятие модуля

**Ceil(-10.493) будет -10 округляет ВВЕРХ до целых**

**Round(-10.493) будет -10 округляет до определенного числа знаков после запятой**

**Round (-10.493, 1) будет -10.5 округляет до определенного числа знаков после запятой, в данном случае один знак**

**Trunc (-10.493, 1) усекает до определенного знака после запятой, будет -10.4**

**Floor(-10.493) будет 11, округляет ВНИЗ до целых**

Remainder(10, 3) аналогично 10%3

Mod(-10, 3) деление по модулю

Cos(), acos(), sin(), asin(), tan(), atan(), exp(), power(число, степень), sqrt(число), log(число, основание)

**Работа с датами**

**current\_date**

**sysdate ТЕКУЩИЕ ДАТА И ВРЕМЯ**

**dbtimezone, sessiontimezone**

localtimestamp

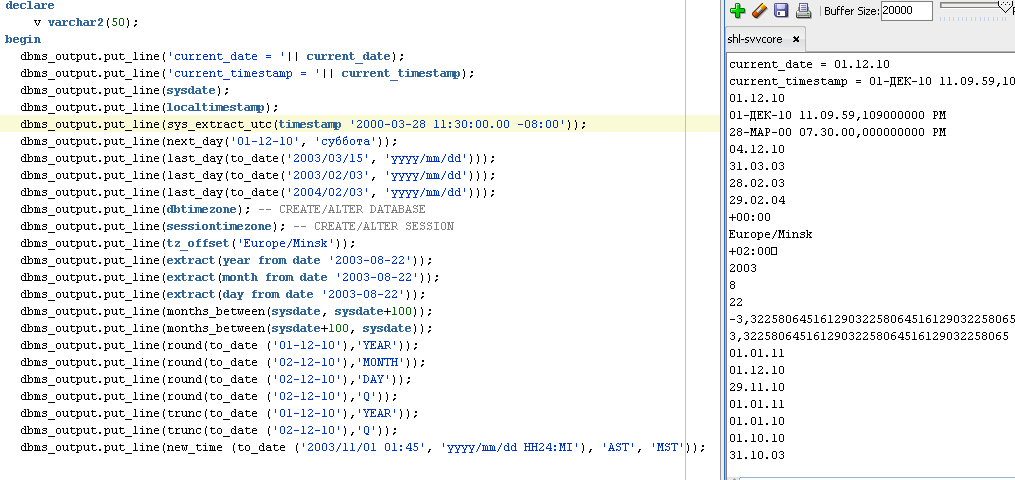
**extract(year from date ‘2003-08-22’)**

**extract(month from date ‘2003-08-22’)**

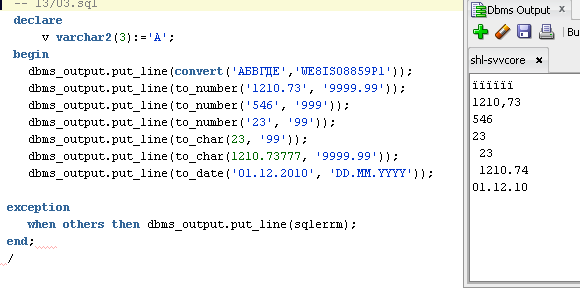
**extract(day from date ‘2003-08-22’)**

**month\_between (startdate, enddate)**

**trunc (d, [формат]) – возвращает дату d, усеченную до указанной единицы измерения**



**Конвертирование**

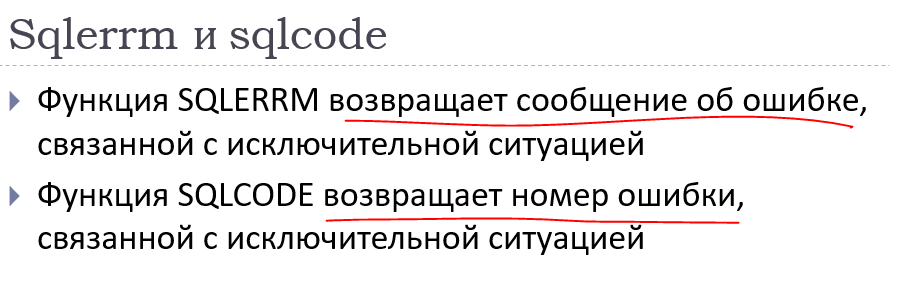


**to\_number(строка, формат), to\_char(строка, формат), to\_date(строка, формат), convert(строка, формат)**

**Convert (строка, формат) конвертирует строку из одного набора символов в другой**

**Обработка ошибок**

sqlcode, sqlerrm



1. Встроенные функции языка PL/SQL. Функции регулярных выражений

Регулярные выражения - **формальный язык поиска и осуществления манипуляций с подстроками в тексте**, **основанный на использовании метасимволов**

**REGEXP\_LIKE выбирает все строки, соответствующие заданному шаблону**

**REGEXP\_LIKE (source\_char, pattern)**

**REGEXP\_INSTR определяет местоположение вхождения шаблона в строку**

REGEXP\_INSTR(source\_char, pattern)

**REGEXP\_REPLACE заменяет шаблон выражения на заданный**

**REGEXP\_REPLACE(source\_char, pattern, replace\_char)**

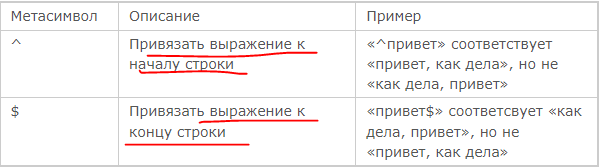
**REGEXP\_SUBSTR выделяет из строки шаблон**

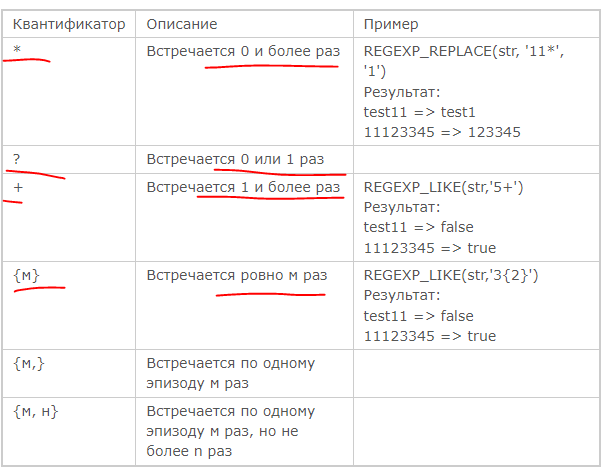
REGEXP\_SUBSTR(source\_char, pattern)

REGEXP\_COUNT определяет количество вхождений

REGEXP\_COUNT (source\_char, pattern)







|  |  |
| --- | --- |
| **Значение** | **Описание** |
| **^** | **Соответствует началу строки** |
| **$** | **Соответствует концу строки** |
| **\*** | **Соответствует 0 или более вхождений.** |
| **+** | **Соответствует 1 или более вхождений.** |
| **?** | **Соответствует 0 или 1 вхождению.** |
| **.** | **Соответствует любому символу, кроме NULL.** |
| **|** | **Используется как "OR", чтобы указать более одной альтернативы.** |
| **[ ]** | **Используется для указания списка совпадений, где вы пытаетесь соответствовать любому из символов в списке.** |
| [^ ] | Используется для указания списока nonmatching, где вы пытаетесь соответствовать любому символу, за исключением тех кто в списке. |
| ( ) | **Используется для групповых выражений в качестве подвыражений.** |
| {m} | **Соответствует m раз.** |
| {m,} | **Соответствие как минимум m раз.** |
| {m,n} | **Соответствие как минимум m раз, но не более n раз.** |
| \n | n представляет собой число от 1 до 9. Соответствует n-му подвыражению находящемуся в ( ) перед \n. |
| [..] | Соответствует одному сопоставлению элемента, который может быть более одного символа. |
| [::] | Соответствует классу символов. |
| \d | **Соответствует цифровому символу.** |
| \D | Соответствует не цифровому символу. |
| \w | **Соответствует текстовому символу.** |
| \W | Соответствует не текстовому символу. |
| \s | **Соответствует символу пробел.** |
| \S | Соответствует не символу пробел. |
| \A | Соответствует началу строки или соответствует концу строки перед символом новой строки. |
| \Z | Соответствует концу строки. |
| {n}? | Соответствует предыдущему шаблону n раз. |
| {n,}? | Соответствует предыдущему шаблону, по меньшей мере n раз. |
| {n,m}? | Соответствует предыдущему шаблону, по меньшей мере n раз, но не более m раз. |

В целом регулярное выражение предусматривает поиск в символьных строках определенных образцов. Символьная строка может относиться к типу CHAR, VARCHAR2, NCHAR или NVARCHAR2, а регулярное выражение — содержать одну из следующих функций:

* REGEXP\_LIKE
* REGEXP\_REPLACE
* REGEXP\_INSTRING
* REGEXP\_SUBSTRING

1. Коллекции. Массивы переменной длины.

**Коллекция – структура данных, содержащая элементы одного типа**

**Элементом коллекции может быть как скалярная величина, так и композитные данные**

Элементы коллекций можно сравнивать между собой на эквивалентность

Можно передавать параметром

Одномерная, но можно создавать коллекции коллекций

* + **Массив переменной длины VARRAY** – **ОГРАНИЧЕНЫ – заранее определены границы. ВСЕГДА ПЛОТНЫЕ (все элементы имеют значение, в том числе null)!**
  + Вложенная таблица (nested tables) – НЕОГРАНИЧЕНЫ. СНАЧАЛА ПЛОТНЫЕ, ПОТОМ МОГУТ РАЗРЕДИТЬСЯ
  + Ассоциативный массив (associative array) – НЕОГРАНИЧЕНЫ. МОГУТ БЫТЬ ПЛОТНЫЕ И РАЗРЕЖЕННЫЕ

Коллекция состоит из **набора** элементов, причем каждый элемент находится в определенной позиции (имеется **индекс** элемента)

Необходимо объявить **тип коллекции** – командой TYPE

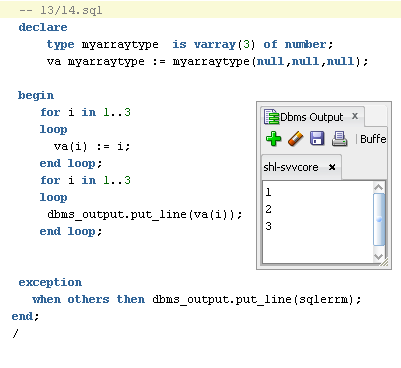
Необходимо объявить **коллекцию** – переменную этого типа для дальнейшего использования

Коллекция называется **ограниченной**, если заранее определены границы возможных значений индексов ее элементов, иначе **неограниченной**

**Коллекция называется плотной, если все ее элементы, определены и каждому из них присвоено некоторое значение** (таковым **может быть и NULL**)

**Массивы переменной длины Varray – одномерные, связанные коллекции однотипных элементов. ЗАРАНЕЕ ЗНАЕМ РАЗМЕР (ОГРАНИЧЕНЫ), ВСЕ ЭЛЕМЕНТЫ ИМЕЮТ ЗНАЧЕНИЕ, В ТОМ ЧИСЛЕ NULL (ПЛОТНЫЕ)**

* Доступны в рамках PL/SQL и в БД
* Являются плотными



declare

type **myarraytype** **is varray(3) of number;**

**va muarraytype := myarraytype(null, null, null)**

**va.extend; -добавляет один нулевой элемент**

**va.extend(n); -добавляет n нулевых элементов**

**va.delete(idx); - удаляет по индексу**

**va.count; вернет количество элементов**

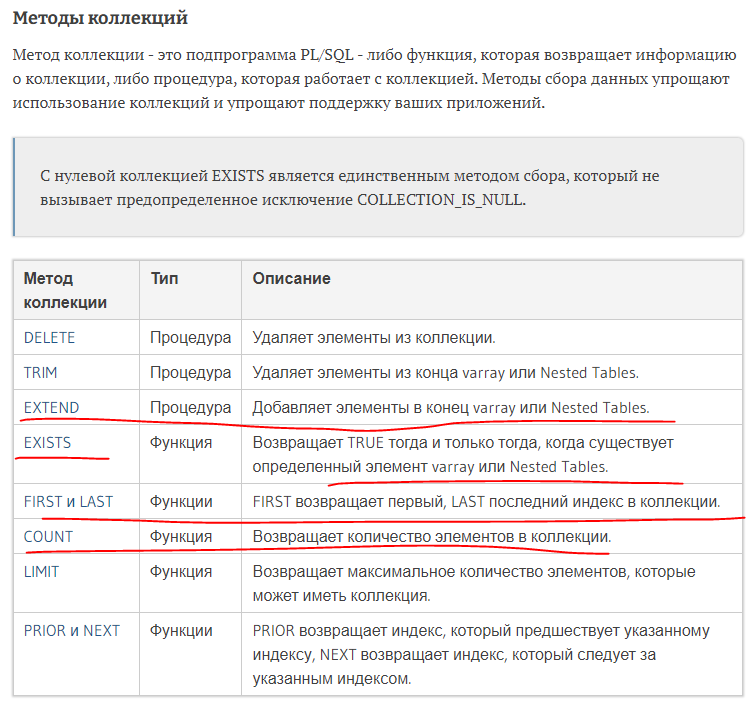
**В Oracle PL/SQL Varray (массив с переменным размером) - это массив, число элементов которого может варьироваться от нуля (пусто) до объявленного максимального размера.**

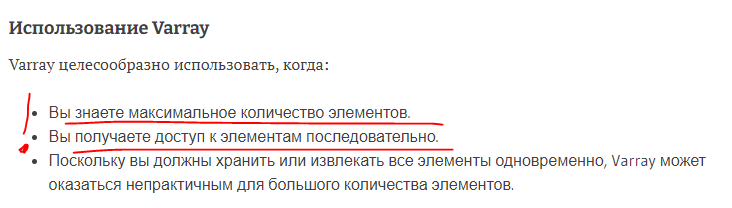
**Чтобы получить доступ к элементу переменной Varray, используйте синтаксис variable\_name(index).**

**Нижняя граница** index равна 1; верхняя граница - это текущее количество элементов.

**Верхняя граница** изменяется при добавлении или удалении элементов, но она не может превышать максимальный размер.

Когда вы храните и извлекаете varray из базы данных, его индексы и порядок элементов остаются стабильными.





1. Коллекции. Вложенные таблицы.

**Коллекция – структура данных, содержащая элементы одного типа**

**Элементом коллекции может быть как скалярная величина, так и композитные данные**

Элементы коллекций можно сравнивать между собой на эквивалентность

Можно передавать параметром

Одномерная, но можно создавать коллекции коллекций

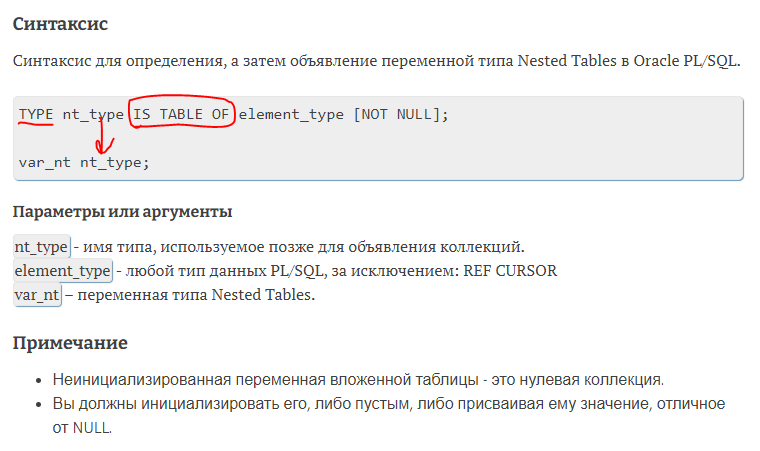
* + Массив переменной длины VARRAY – ОГРАНИЧЕНЫ – заранее определены границы. ВСЕГДА ПЛОТНЫЕ (все элементы имеют значение, в том числе null)!
  + **Вложенная таблица (nested tables) – НЕОГРАНИЧЕНЫ. СНАЧАЛА ПЛОТНЫЕ, ПОТОМ МОГУТ РАЗРЕДИТЬСЯ**
  + Ассоциативный массив (associative array) – НЕОГРАНИЧЕНЫ. МОГУТ БЫТЬ ПЛОТНЫЕ И РАЗРЕЖЕННЫЕ

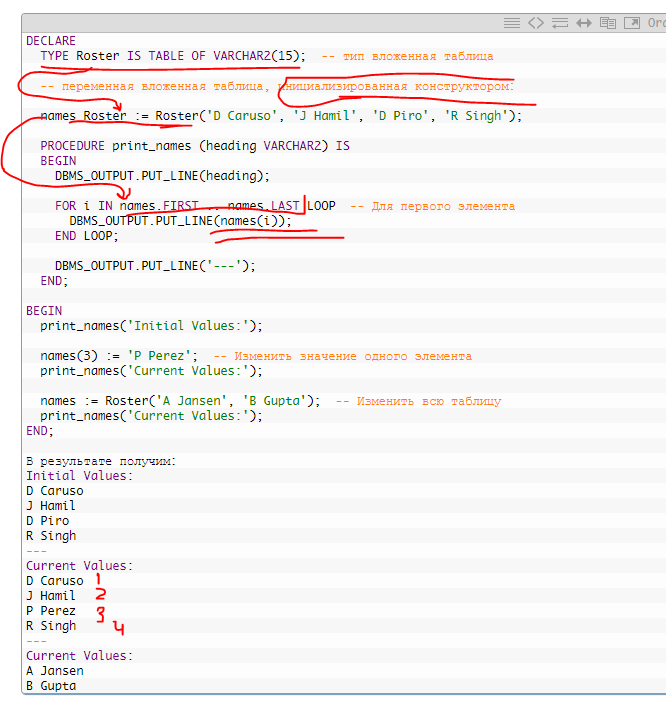
**Вложенные таблицы** Nested Tables – одномерные, **несвязанные** коллекции однотипных элементов. НЕОГРАНИЧЕНЫ! Элементы могут содержать значения, а могут не содержать (потом вложенная таблица разредиться)

Доступны в рамках PL/SQL и как поля таблицы в БД

Изначально являются плотными, но могут впоследствии становиться разреженными

В **Oracle PL/SQL Nested Tables** представляет собой **тип столбца, который хранит неограниченный набор строк в определенном порядке**. Когда вы извлекаете значение вложенной таблицы из базы данных в переменную Nested Tables, PL/SQL выдает строки последовательных индексов, начиная с 1. Используя эти индексы, вы можете обращаться к отдельным строкам переменной Nested Tables.





1. Коллекции. Ассоциативные массивы.

**Коллекция – структура данных, содержащая элементы одного типа**

**Элементом коллекции может быть как скалярная величина, так и композитные данные**

Элементы коллекций можно сравнивать между собой на эквивалентность

Можно передавать параметром

Одномерная, но можно создавать коллекции коллекций

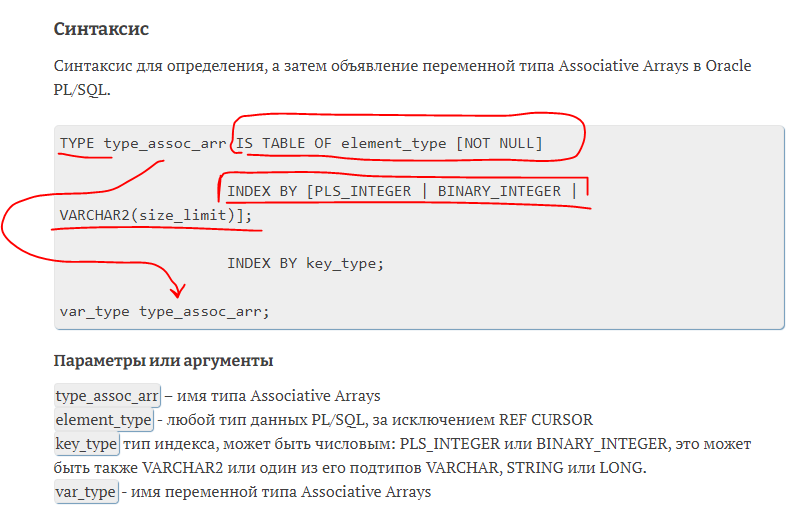
* + Массив переменной длины VARRAY – ОГРАНИЧЕНЫ – заранее определены границы. ВСЕГДА ПЛОТНЫЕ (все элементы имеют значение, в том числе null)!
  + Вложенная таблица (nested tables) – НЕОГРАНИЧЕНЫ. СНАЧАЛА ПЛОТНЫЕ, ПОТОМ МОГУТ РАЗРЕДИТЬСЯ
  + **Ассоциативный массив (associative array) – НЕОГРАНИЧЕНЫ. МОГУТ БЫТЬ ПЛОТНЫЕ И РАЗРЕЖЕННЫЕ**

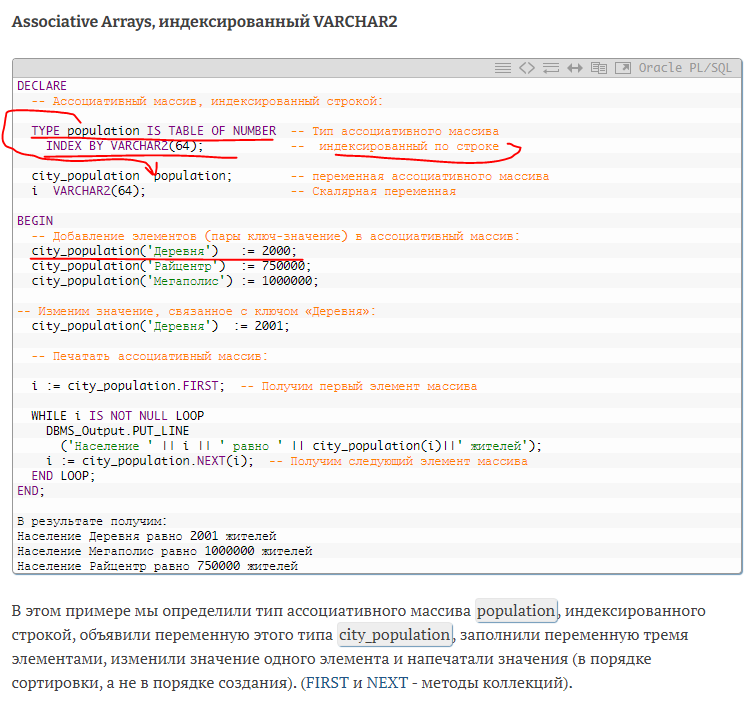
Ассоциативные массивы – одномерные, неограниченные (по максимальному количеству элементов при создании) коллекции элементов

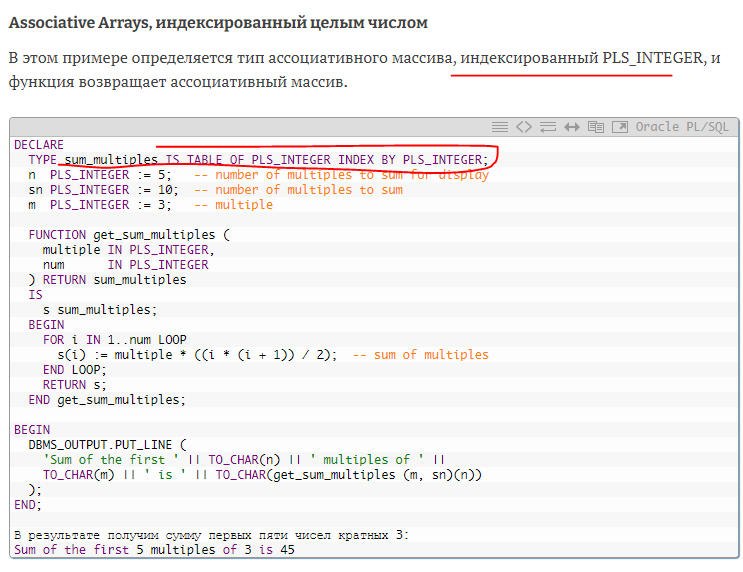
Доступны только в рамках PL/SQL

Изначально являются разреженными, индекс может принимать непоследовательные значения

В Oracle PL/SQL **Associative Arrays**, также известные как индексные таблицы, в которых для значений индекса используя произвольные числа и строки. **Associative Arrays - это набор пар ключ-значение, где каждый ключ уникален и используется для нахождения соответствующего значения в массиве.**







Представляют собой множество пар ключ-значение

**type array\_name is table of datatype index by value;**

|  |  |  |
| --- | --- | --- |
| **Массив переменной длины Varray** | **Вложенная таблица Nested table** | **Ассоциативный массив Associative Arrays** |
| **Ограничен**  **Всегда плотный**  type **myarraytype is varray(3) of number;**  va muarraytype := myarraytype(null, null, null) | **Неограничена**  **Поначалу плотная, может становиться разреженной**  type table\_name is table of datatype; | **Неограничен**  **Может быть плотный или разреженный**  type array\_name is table of datatype index by value; |

1. Процедурные объекты. Хранимые процедуры. Вызов процедур. Входные и выходные параметры, позиционный и параметрический форматы передачи фактических параметров. Значения параметров по умолчанию.

**- Функция – именованный модуль, возвращает значение (результат работы) в вызывающую программу через return, выполняет 0 или несколько действий**

**- Процедура – именованный модуль, выполняет 1 или несколько действий и может принимать, возвращать значения через список параметров**

- **Пакет – коллекция PL\SQL объектов, сгруппированных вместе, набор переменных, процедур, функций, связанных функциональным замыслом, состоит из спецификации (заголовка) и тела (реализация)**

- **Триггер – хранимая процедура, которая автоматически запускается при событии DML или DDL,** обычно связаны с insert, update, delete в некоторой таблице

Процедура – именованный модуль, который выполняет одно или несколько выражений и может принимать или возвращать значения через список параметров

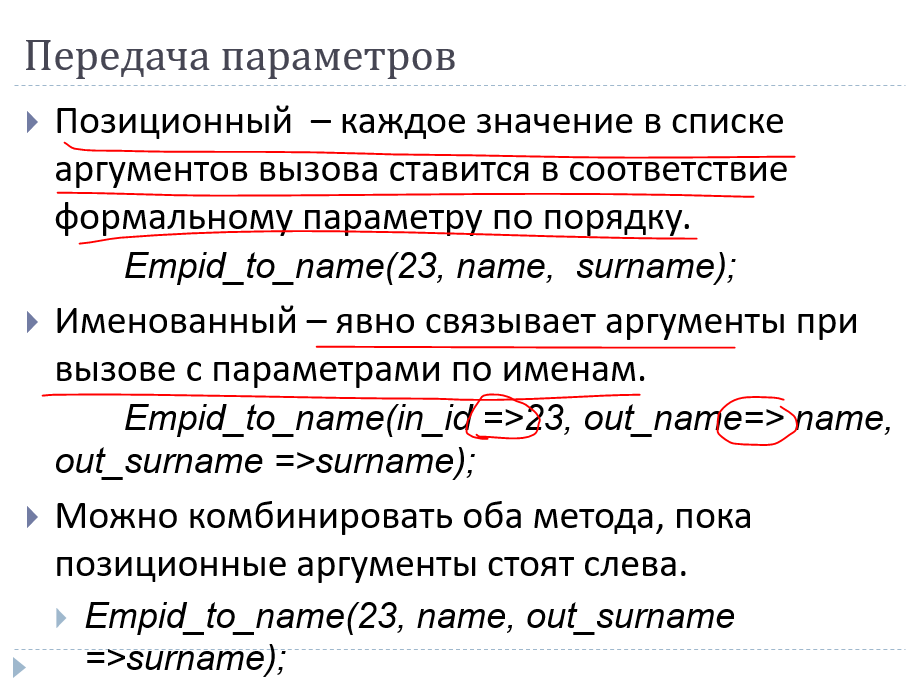
**Привилегия create procedure**

**Параметры: наименование – режим передачи – тип данных – начальное значение**

Типы параметров:

1. **In – передача в процедуру, передача по ссылке**
2. **Out – получение из вызова, возврат, передача по ссылке**
3. **In out – передается и возвращается обновленное значение, передача по значению**

По умолчанию IN, IN OUT. МОЖНО НЕ УКАЗЫВАТЬ! Возьмется по умолчанию



create procedure xxx (параметр тип данных тип передачи, ….)

as/is

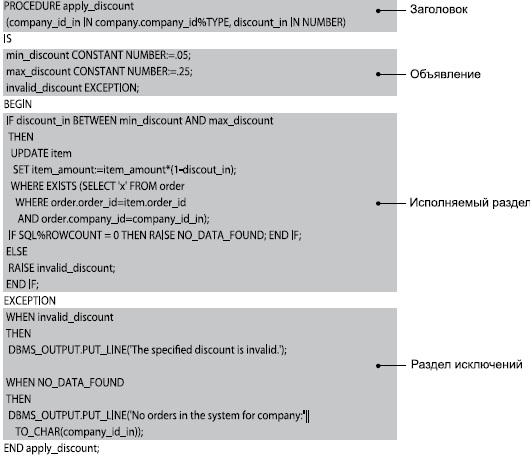
begin

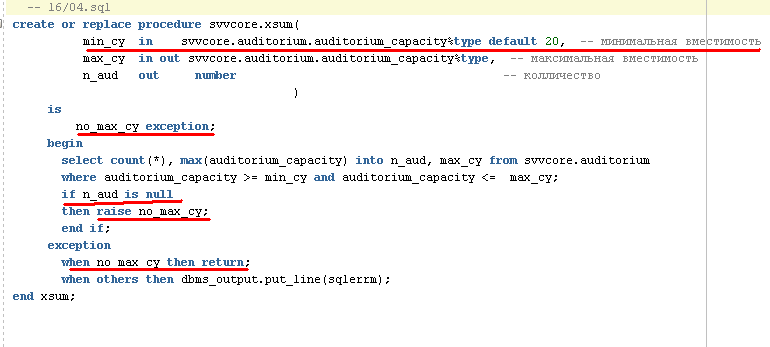
…..

end xxx;

**Концептуально: процедуры изменяют состояние объектов или среды, манипулируют данными. А функции не меняют состояние и данные, от них ожидается вычисление данных**

Вызов осуществляется из блока PL/SQL имя(параметры) или через execute, ну или через меню connection -> procedures -> run





**Значения по умолчанию**

**Имя\_параметра тип\_передачи тип\_данных default значение**

1. Процедурные объекты. Хранимые функции. Параметры функции. Вызов функций. Понятие детерминированной функции. Понятие pipeline функции. Значения параметров по умолчанию.

**Функция – именованный модуль, выполняет 0 или несколько действий, а через фразу return всегда возвращает результат своей работы**.

Вызов:

1. Присвоение начального значения переменной
2. В присвоении
3. В булевом выражении
4. В SQL запросе
5. Как аргумент другой функции или процедуры

Create function xxx () return тип

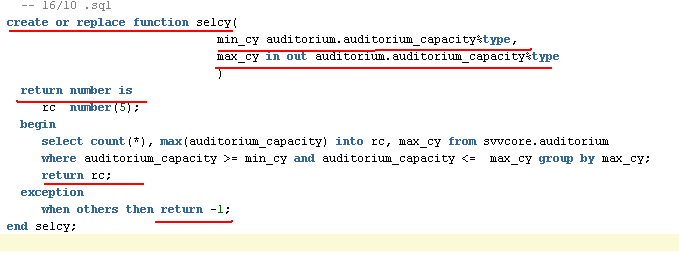
Is

Begin

….

End xxx;

Функции можно использовать в select, insert, update, delete



Параметры функции: in, out, in out

Формат записи: имя – формат передачи – тип данных

Функция может не принимать параметров:



* DETERMINISTIC – **функция детерминирована, если она возвращает одно и то же значение при вызове с теми же параметрами**
* AGGREGATE USING – **используется для агрегатных функций.**



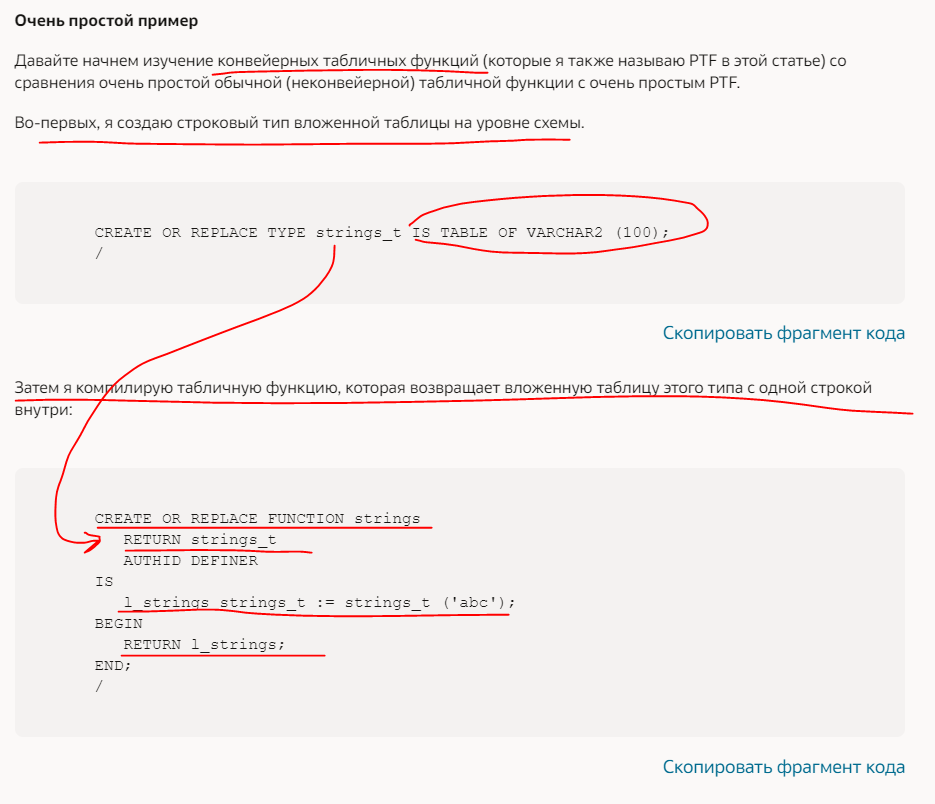
**Конвейерные pipeline табличные функции — это табличные функции, которые возвращают или «направляют» строки обратно в вызывающий запрос по мере того, как функция создает данные в нужной форме** — **и *до* того, как функция завершит всю свою обработку**.

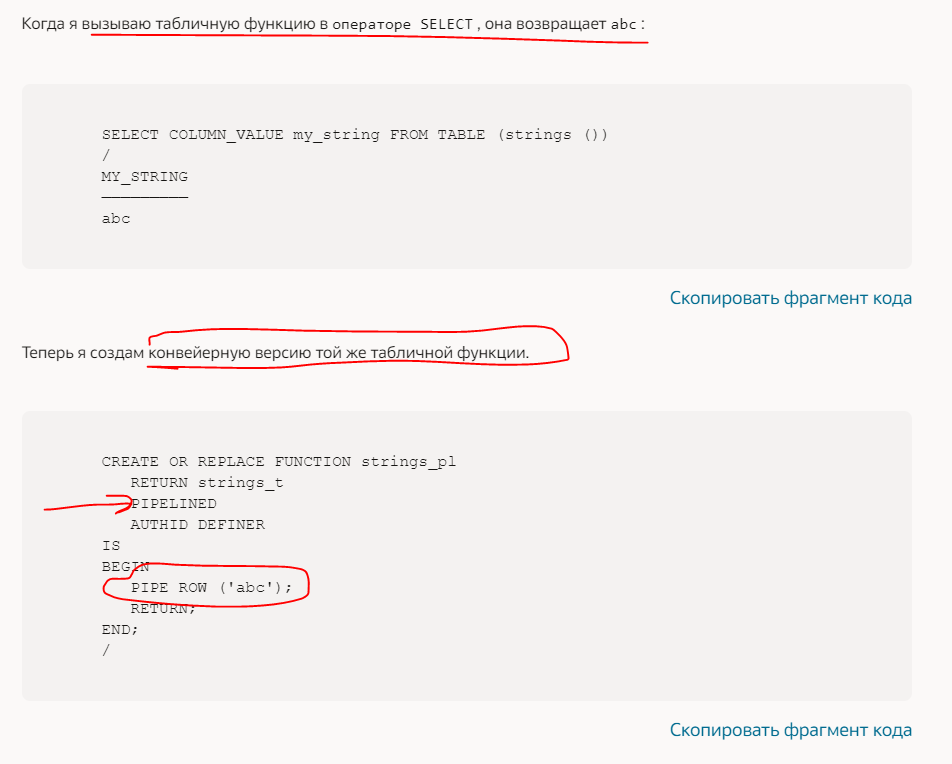
**PL/SQL не является многопоточным языком. Как правило, когда блок PL/SQL (анонимный, вложенный, процедурная функция и т. д.) вызывается, дальнейшая обработка в этом сеансе «приостанавливается»** (приостанавливается) до тех пор, пока блок не вернет управление хосту, вызвавшему блок — будь то этот хост является другим блоком PL/SQL, оператором SQL или базовым языком, таким как Java.

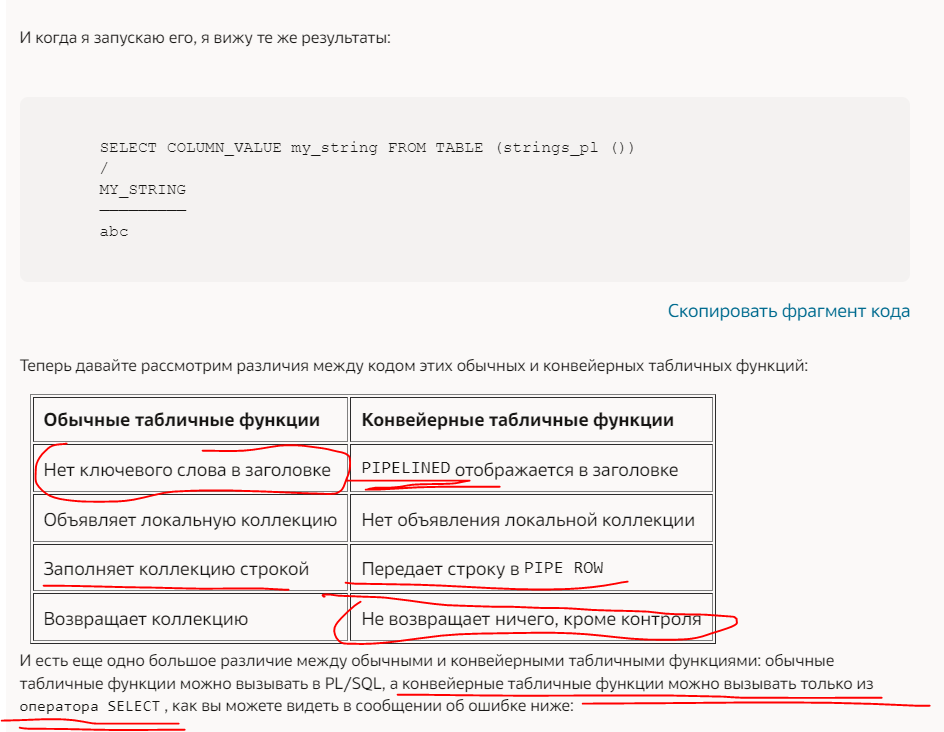
Обычные (неконвейерные) табличные функции действуют именно таким образом. **Каждый раз, когда табличная функция вызывается в предложении FROM, механизм SQL должен ждать, пока не будет выполнен оператор RETURN, чтобы передать коллекцию обратно оператору SELECT для преобразования в строки и столбцы.**

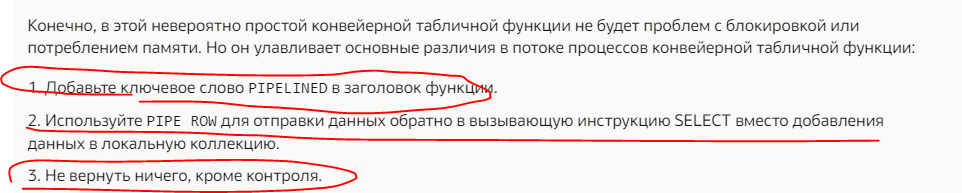
**Такое блокирующее поведение может отрицательно сказаться на общей производительности инструкции SELECT** , особенно в операциях ETL (извлечение, преобразование и загрузка) хранилища данных. Кроме того, с каждым элементом, добавляемым в коллекцию в табличной функции, потребляется все больше и больше памяти Program Global Area (PGA). Для очень больших наборов данных это может привести к дальнейшему снижению производительности и даже ошибкам.

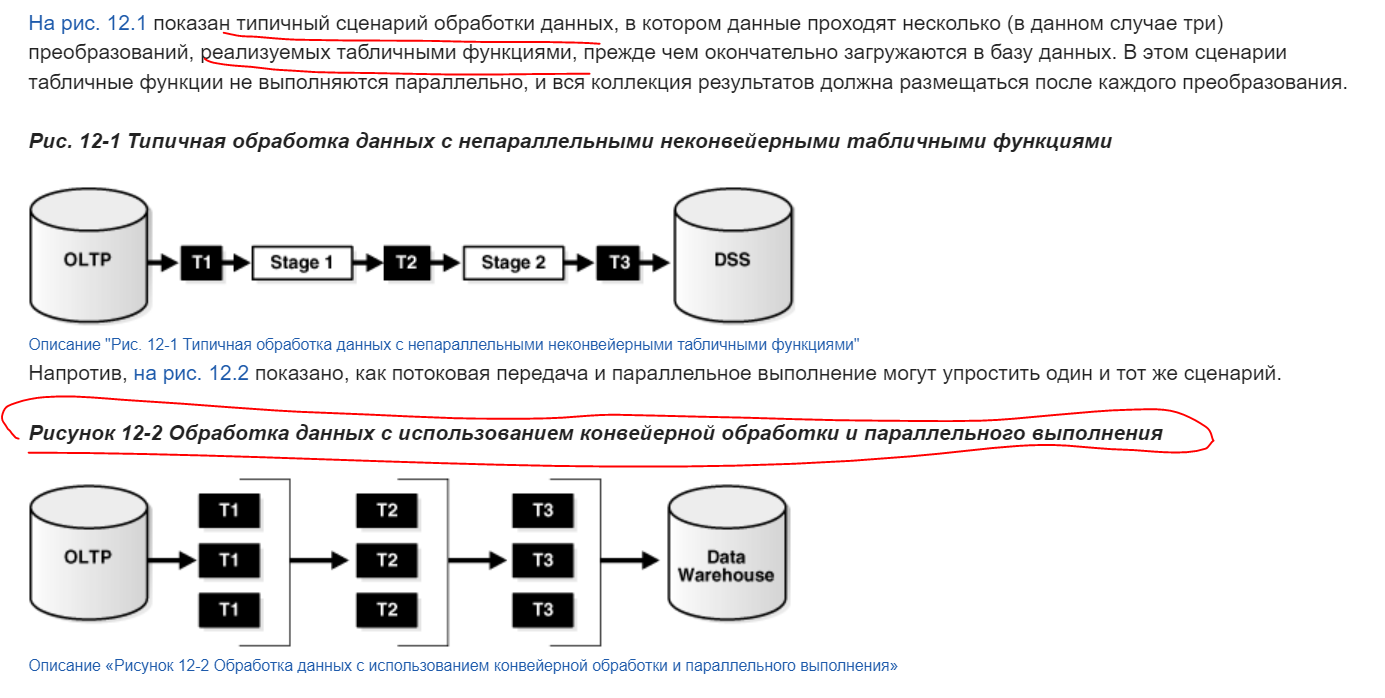
Конвейерные табличные функции решают обе эти проблемы.











Данные называются конвейерными, **если они потребляются потребителем (преобразованием) сразу после того, как производитель (преобразование) их производит, без помещения в таблицы или кэш перед входом в следующее преобразование.**

**Конвейерная обработка позволяет табличной функции быстрее возвращать строки и может уменьшить объем памяти, необходимый для кэширования результатов табличной функции.**

**Конвейерная функция или pipelined-функция** – табличная функция, которая **поставляет свои результаты клиенту по мере их подготовки. Не материализуют весь результирующий набор.**

**Профит**

**- можно сократить выполнение sql-запросов и pl/sql-кода, ЛИБО поставить результат гораздо раньше, чем это возможно с обычными функциями**

**- уменьшает затраты ресурсов при материализации результата (PGA)**

(PGA может забиться одним лишь результирующим набором функций, когда, например, у нас коллекция состоит из объекта, а объект состоит из нескольких полей, причем строковые и длинные и все они заполнены + работают параллельные сессии)

**Синтаксис**

create or replace function func\_name(…) return тип\_коллекция

**pipelined**

is

begin

…цикл…

**pipe row** (элемент\_коллекции);

…

end;

**От обычной функции отличается ключевой конструкцией pipelined.**

**Обязательно должна возвращаться коллекция**

**Её заполнение происходит по мере выполнения через pipe row (в pipe row указывается элемент коллекции)**

Важный момент: результаты должны отдаваться итеративно в цикле. Это м.б., например, проход по курсору (так обычно бывает при обработке входных данных), либо просто цикл

Как только программа достигает участка с pipe row, результат будет передан в вызывающую среду. При этом вызывающая сторона уже может что-то делать с этой строкой.

**Конвейерная функция - табличная функция, которая поставляет свои результаты в вызывающую среду по мере их подготовки.**

1. **Когда вам нужно вернуть большой набор результатов из блока PL/SQL:**

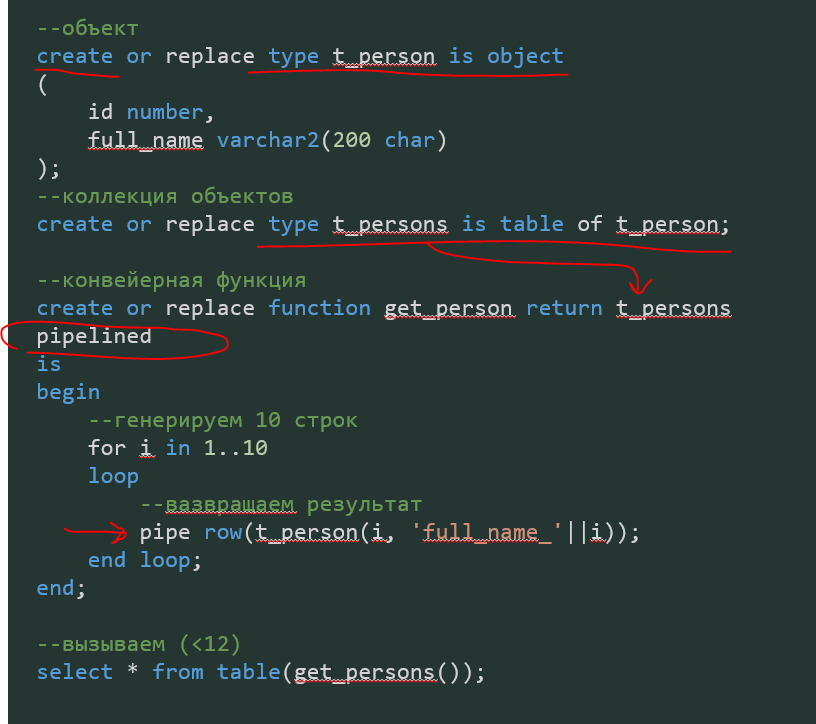
**Конвейерные функции позволяют вам вернуть большой набор результатов из блока PL/SQL в оператор SELECT, что может быть более эффективным, чем использование неконвейерной функции или курсор**

**2) Когда вам нужно преобразовать данные или манипулировать ими перед их возвратом:**

**Конвейерные функции позволяют выполнять преобразования или манипуляции с данными по мере их извлечения и возвращать измененные данные в вызывающий запрос.**

**3) Когда вам нужно вернуть данные из нескольких таблиц или других источников данных:**

**Конвейерные функции можно использовать для извлечения данных из нескольких таблиц или других источников данных и возврата их в виде единого результирующего набора для вызывающего запроса**



1. Процедурные объекты. Пакеты. Спецификация и реализация пакета.

**Пакет – коллекция PL\SQL объектов, сгруппированных вместе. Содержит спецификацию – заголовок, а также реализацию – тело.**

**Служит для:**

**- Улучшенной производительности**

**- Сокрытия информации**

**- Объектно-ориентированного дизайна**

**- Группирует вместе объекты с функциональным смыслом**

**- Постоянство объектов в транзакциях**

**Может содержать:**

1. **Процедуры**
2. **Функции**
3. **Константы**
4. **Исключения**
5. **Курсоры**
6. **Записи**

**Спецификация, заголовок, header**: обязателен, содержит список объектов для общего доступа из других программных модулей или приложений.

**Create or replace package xxxx is**

***…. Пишем определение общих типов и объектов***

**End xxxx;**

**Реализация, тело, body**: содержит программный код для реализации процедур или функций из спецификации, приватные объекты, секцию инициализации

**Create or replace package body xxxx is**

**…. *Пишем тела подпрограмм, реализации***

**End xxxx;**

**Вызов пакета:**

* + **Package\_name.package\_element;**

**Структуры данных, объявленные в пакете, называются пакетными данными**

**Пакетные переменные сохраняют свое состояние от одной транзакции к другой и являются глобальными данными**

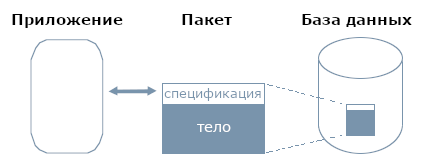
* Словарь: USER\_PROCEDURES, USER\_SOURCE

**Видимым и доступными являются объявления в спецификации (заголовке)**

Пакетированные общие переменные продолжают существовать в течение всей сессии – они могут совместно использоваться всеми процедурами в данном окружении

**В Oracle PL/SQL набор элементов: процедур, функций, определения типов; объявления переменных, констант можно объединить в пакет.** После написания **пакет PL/SQL компилируется, а затем сохраняется в базе данных Oracle, где его содержимое может использоваться многими приложениями**.

Как показано на рисунке, **вы можете думать о спецификации как о рабочем интерфейсе, а о теле - как о «черном ящике». Вы можете отлаживать, улучшать или изменять тело пакета без изменения интерфейса (спецификации) пакета**.



**Спецификация пакета содержит публичные объявления, которые видны вашему приложению.** Вы должны объявить подпрограммы в конце спецификации после всех других элементов (кроме прагм, которые вызывают конкретную функцию; такие прагмы должны следовать спецификации функции).

**Тело пакета содержит детали реализации и приватные объявления, которые скрыты от вашего приложения**. За декларативной частью тела пакета следует необязательная часть инициализации, которая обычно содержит **операторы, которые инициализируют переменные пакета**.

1. Процедурные объекты. Триггеры. Виды триггеров. Классификация, порядок выполнения и предикаты триггеров. Триггеры замещения. Привилегии. Включение/отключение триггеров. Псевдозаписи old и new.

**Триггер – именованный блок (модуль) PL/SQL, который хранится в базе данных и который может быть вызван повторно – триггер можно включать или отключать, но он НЕ ВЫЗЫВАЕТСЯ НАПРЯМУЮ** (по умолчанию триггер включен. **Триггер происходит в ответ на события, возникающие в базе данных**

**Create trigger xxxx**

**Before/after insert/delete/update on xxxx ИЛИ instead of … on …**

**[for each row]**

**Begin**

**….**

**End;**

Grant create trigger to username – привилегия – НАДО НАПРЯМУЮ ПОЛЬЗОВАТЕЛЮ!

Указываем инициирующее событие в терминах запускающих операторов и объекта, на который они будут действовать

Триггеры:

1. DML – на таблицу или представление; insert, update, delete
2. System – на схему или базу данных
3. Instead of – триггер на представление

**По привязке к объекту:**

**- на таблицу**

**- на представление INSTEAD OF**

По событиям запуска триггера:

- insert добавление строки в таблицу или представление

- update / update of, of указывает на поля, update это обновление данных

**- delete (не truncate – удаляет минуя корзину)**

По времени срабатывания:

**- before перед оператором, до записи в журнал**

**- after после оператора, после записи в журнал**

По области действия:

**- уровня оператора ПО УМОЛЧАНИЮ, 1 раз на триггерное событие**

**- уровня строк, СТРОКОВЫЙ for each row СРАБОТАЕТ ДЛЯ КАЖДОЙ ИЗМЕНЕННОЙ СТРОКИ, применяются :new.\_\_\_; :old.\_\_\_\_\_\_**

**- составной compound**

**Триггер – особый вид процедур, которые срабатывают по запускающему их событию**

* **для реализации сложных ограничений целостности базы данных**;
* **для аудита** (контроля хранимой и изменяемой информации);
* **для автоматического оповещения программ** о произошедших событиях;



Триггер – часть транзакции, **ошибка в триггере откатывает операцию, изменения таблиц в триггере становятся частью транзакции.**

Если откатывается транзакция, изменения триггера тоже откатываются.

***Не может выдавать COMMIT/ROLLBACK (исключение - только, если в теле триггера есть автономная транзакция)***

**Может выдавать RAISE\_APPLICATION\_ERROR**

Порядок выполнения DML-триггеров

* операторные operator BEFORE;
* для каждой строки row BEFORE;
* выполняется оператор operator;
* для каждой строки row AFTER;
* операторные operator AFTER.

**Триггер замещения**

Instead of триггер – выполнится ВМЕСТО оператора, для которого создается триггер. ТОЛЬКО УРОВНЯ СТРОК!!!!

Служит для замещения DML операций НАД ПРЕДСТАВЛЕНИЯМИ своим функционалом.

create or replace trigger view\_teacher\_insteadof\_insert

**instead of insert** **on teachers\_view**

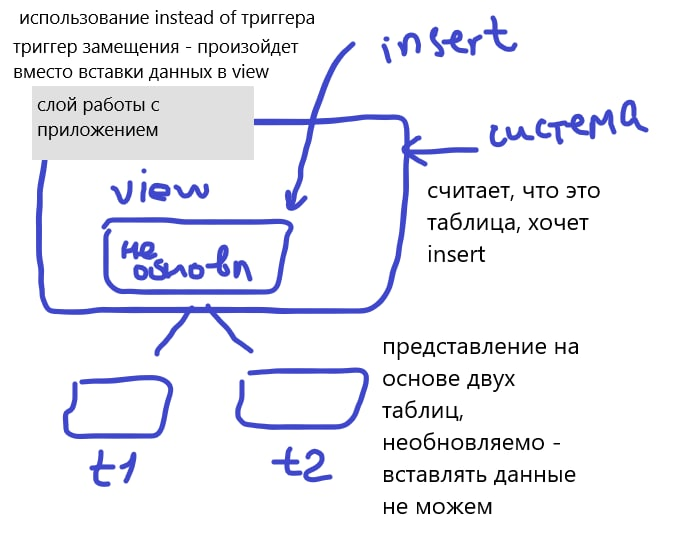
for each row

begin

dbms\_output.put\_line('Новый набор данных: ' || :new.t || ' ' || :new.tname || ' ' || :new.pulp || ' ' || :new.bday || ' ' || :new.slry);

--insert into TEACHER values ('ГИИ', 'Гур Иван Иванович', 'ИСиТ', '20.09.1960', 4590);

end;



**Предикаты триггера**

***Чтобы различать DML команды и события, которые выполняют триггер, используются триггерные предикаты INSERTING, UPDATING, and DELETING в условиях IF***

Create or replace trigger xxxx

After insert or update or delete on tablename

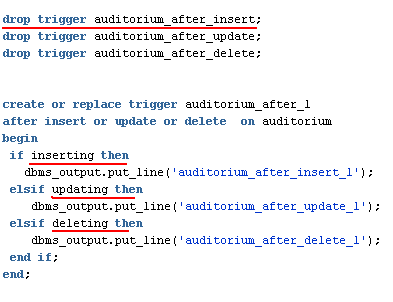
Begin

If inserting then ….

If deleting then ….

If updating then ….

End;





create or replace trigger teacher\_before\_insertROW

before insert on TEACHER

for each row

declare

datastr varchar2(600);

begin

dbms\_output.put\_line('teacher\_before\_insertROW');

datastr:= 'Вставляемый набор данных: ' || :new.TEACHER || ' ' || :new.TEACHER\_NAME || ' ' || :new.PULPIT || ' ' || :new.BIRTHDAY || ' ' || :new.SALARY;

insert into MYAUDIT (TRIGGER\_NAME, OPERATION\_TYPE, OPERATION\_DATA)

values ('teacher\_before\_insertROW', 'I', datastr);

end;

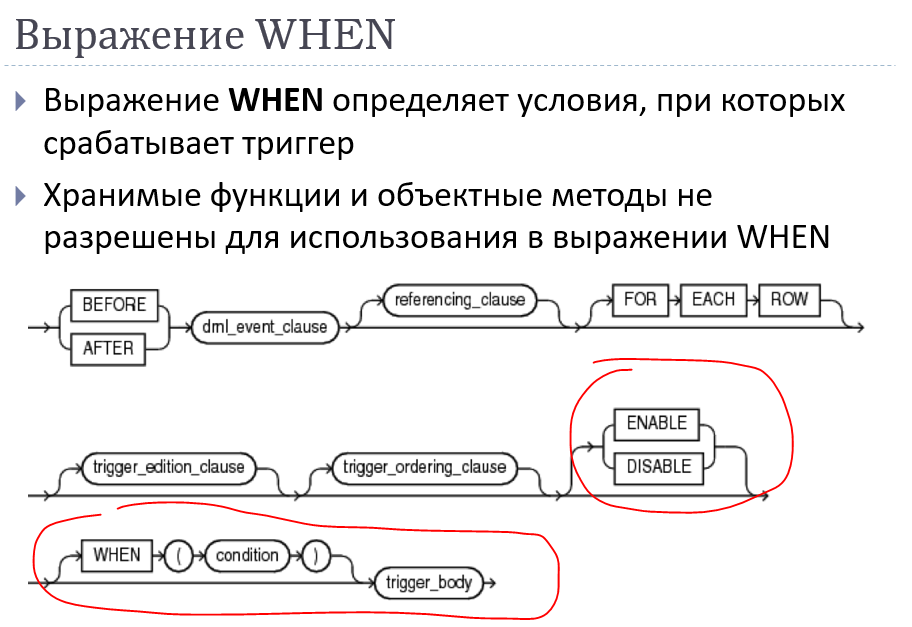
Включение или отключение триггеров

\* **alter trigger { disable | enable }**

\* всех для таблицы: **alter table .. { enable | disable } all triggers;**

\* **компиляция триггера : alter trigger .. compile**

**\* переименование триггера**



* dba\_triggers
* dba\_source
* dba\_objects

create or replace trigger tr\_drop\_table

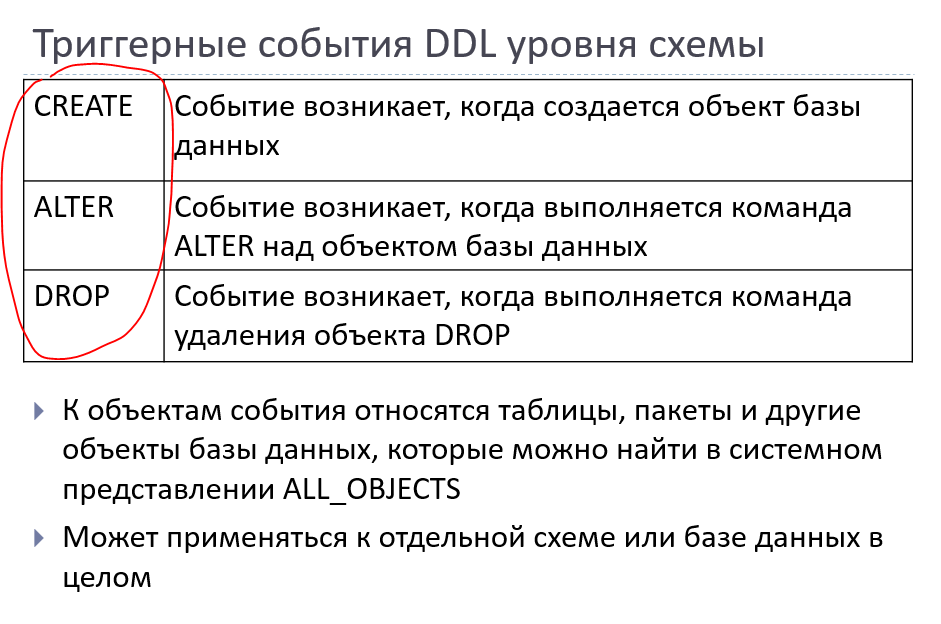
**before drop on SCHEMA**

begin

if **ORA\_DICT\_OBJ\_NAME = 'TEACHER'**

then raise\_application\_error(-20000, 'Удаление таблицы запрещено. Объект: ' || ORA\_DICT\_OBJ\_TYPE || ' ' || ORA\_DICT\_OBJ\_NAME);

end if;



По времени срабатывания:

* + BEFORE, AFTER

По уровню триггера:

* + SCHEMA, DATABASE

По виду события:

1)серверные события

2) DDL-события

3)события сбора статистики

4) события аудита (диагностическая информация об объектах базы данных)

5) DCL-события

**СИСТЕМНЫЕ - срабатывают на события уровня схемы или уровня базы данных**

**Триггер может следить за конкретной схемой, некоторые не бывают before/after, по-разному**

у **нас например нет before startup - нет такого !!! или after logoff**

Можно указать группу событий

**- сервер (logon, logoff, включение или выключение сервера)**

**- DDL (создание объектов, таблиц, процедур, ролей, профилей безопасности)**

**- события сбора статистики**

**- события аудита**

**- DCL события - выдача или отзыв прав**

**Уровня схемы: create/alter/drop + указание объекта, на которые сработает событие**

**grant, revoke, любые события DDL, отдельно идет truncate**

SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN

Все кроме LOGOFF работают в автофиксации. LOGOFF входит в транзакцию отключения

Системный триггер может генерировать исключение raise

**create or replace trigger name**

**after logon on guest.schema**

**begin**

**//пишем в таблицу например входа пользователей или в аудит**

**end name;**

***Если мы хотим триггеры на таблицы, за которыми надо следить - юзаем DML***

***Если хотим поставить нормальный аудит - используем системные триггеры, по системным событиям, чаще всего ставят на:***

***- изменение прав!!! оно редчайше - триггер всегда***

***- триггер на работу вне рабочее время - отслеживать время события, если входы в систему вне рабочее время - отслеживаем и фиксируем в спец таблице,*** она делается таким образом, что к ней нельзя получить доступ без ведома админов, в специальной схеме

1. Секционирование таблиц. Виды секционирования.

Секционирование - **метод, позволяющий хранить сегмент данных, такой как таблица, в виде нескольких сегментов, сохраняя логическую монолитную структуру – данные остаются непротиворечивы, нет нарушений целостности, выполняются нормальные формы**

***Все секции таблицы могут располагаться в разных табличных пространствах – это позволит разместить их на разных дисках для увеличения производительности и степени параллелизма***



**Секционирование данных: разбить по горизонтали, какие-то строки в одну секцию, другие - в другую**

ЛОГИЧЕСКИ – один объект. ФИЗИЧЕСКИ – объект состоит из разных частей

**Ключ секционирования – набор столбцов, которые определяют, в какую секцию будет попадать та или иная строка. Секция – единица параллелизма**

**чем хорошо? мы можем обратиться ко всей таблице и осуществлять сбор по всей таблице, а можем по секции - только то, что надо**

user/all/dba

\_tables

\_part\_tabes информация о секционированных таблицах

\_tab\_partitions секции таблиц

\_part\_key\_columns по какому ключу секционировались - выбор диапазона, по которому будет происходить деление на секции

\_segments информация о сегментах хранения - какие сегменты существуют

\_objects информация об объектах бд

**Секция - отдельный сегмент**

**Можно сегмент класть в одно ТП, а можно в разные ТП на разных дисках**

Когда применить?

* 1. **Повышение доступности данных – секция является отдельной сущностью, если недоступна одна, не значит, что недоступен весь объект и другие секции. Если какое-то ТП в режиме offline, работоспособность не нарушается**
  2. **Повышение производительности в операциях – распараллеливание DML**
  3. **Упрощение администрирования**
  4. **Балансировка дисковой нагрузки**
  5. **Упрощение управления крупными объектами хранения – крупный объект разбивается на индивидуально управляемые части**
  6. Сокращение времени обслуживания
  7. Нечувствительность к сбоям

В каждой секции, при том что это как будто одна общая таблица, данные могут храниться по-разному: **разный размер блока, проценты заполнения... - имеют собственные ФИЗИЧЕСКИЕ атрибуты, но ОБЩИЕ ЛОГИЧЕСКИЕ**

**Что в каждой секции свое? Параметры табличного пространства, параметры блока, атрибуты сжатия и логирования**

Секция - единица параллелизма. **Если есть параллельные операции, секции могут загружаться и обновляться параллельно - можно использовать параллельную обработку данных секций**

**Если одна секция в состоянии offline, и если мы попытаемся извлечь данные из таблицы, возникнет ОШИБКА ЧТЕНИЯ**

**А если обращаемся к другим секциям - они работают и доступны**

Блокировка таблицы: если есть незавершенная транзакция

Типы секционирования:

1. **Диапазонное RANGE – секционирование по диапазону ключа секционирования, обычно по диапазону значений или диапазону дат**

create table T\_RANGE

(

**current\_id number,**

word varchar2(255)

)

**partition by range (current\_id)**

(

partition T\_RANGE\_q1 **values less than (100**),

partition T\_RANGE\_q2 values less than (200),

partition T\_RANGE\_q3 values less than (300),

partition T\_RANGE\_q4 **values less than (maxvalue)**

);

select \* from T\_RANGE partition (T\_RANGE\_q1);

**Минус: при загрузке новых данных надо постоянно расщеплять maxvalue**

1. **Интервальное INTERVAL**

create table T\_INTERVAL

(

productid number,

timeid date

)

**partition by range (timeid)**

**interval (interval '1' month) store in (users)**

(

partition T\_INTERVAL\_q1 values less than (TO\_DATE('01.12.2022', 'DD.MM.YYYY'))

);

**'01.12.2022' – ТОЧКА ПЕРЕХОДА!**

**При интервальном секционировании после того как данные в базе данных пересекают точку перехода, автоматически создаются новые секции**

В примере выше – идем по интервалу 1 месяц, создаем секцию для записей, где значение даты раньше, чем 01.12.2022 – все до этой даты влетят в эту секцию

Значит если появится 01.01.2023, создастся секция автоматически. 01.02.2023 – еще одна секция (интервально идем по месяцам)

**НОВЫЕ СЕКЦИИ СОЗДАЮТСЯ АВТОМАТИЧЕСКИ ПО ПЕРВОЙ ВСТАВКЕ, КОТОРАЯ НЕ ПОПАДАЕТ В ДИАПАЗОН СУЩЕСТВУЮЩИХ СЕКЦИЙ!!!**

Когда секционирование интервальное, мы добавляем новую строку - может создаться новая секция. А как генерируется имя? По какому принципу?

**Партиции SYS\_P472, SYS\_P473, SYS\_P474 - выдаются в порядке появления предыдущих записей**

Номера идут последовательно, а секции создаются как надо

**Самое первое название секции - из определения таблицы**

**Все остальные выдает сервер, при этом с какого имени начнет : SYS\_Pномер**

**номер в зависимости от того, что он раньше выдавал**

**определенный набор номеров последовательно - для этого есть системная последовательность**

****

1. Хэш-секционирование – **реализация хэш-функции неизвестна, строки таблицы РАВНОМЕРНО (НЕ СЛУЧАЙНО) распределяются между секциями на основании внутренних алгоритмов хэширования Oracle**

create table T\_HASH

(

cityid number,

word varchar2(255)

)

**partition by hash (word) word должно быть с высокой уникальностью!!!!**

**partitions 4; рекомендуется степень 2 = 2, 4, 8, 16, 32 ….**

***Реализация хэш-функции не известна***

***Заранее не сказать, куда попадут данные. Задача - равномерно распределить данные***

1. **Списочное – разбивает таблицу на секции по списку дискретных значений**

create table T\_LIST

(

operid number primary key,

opertype char(2)

)

**partition by list (opertype)**

(

partition T\_LIST\_q1 **values ('d', 'u'),**

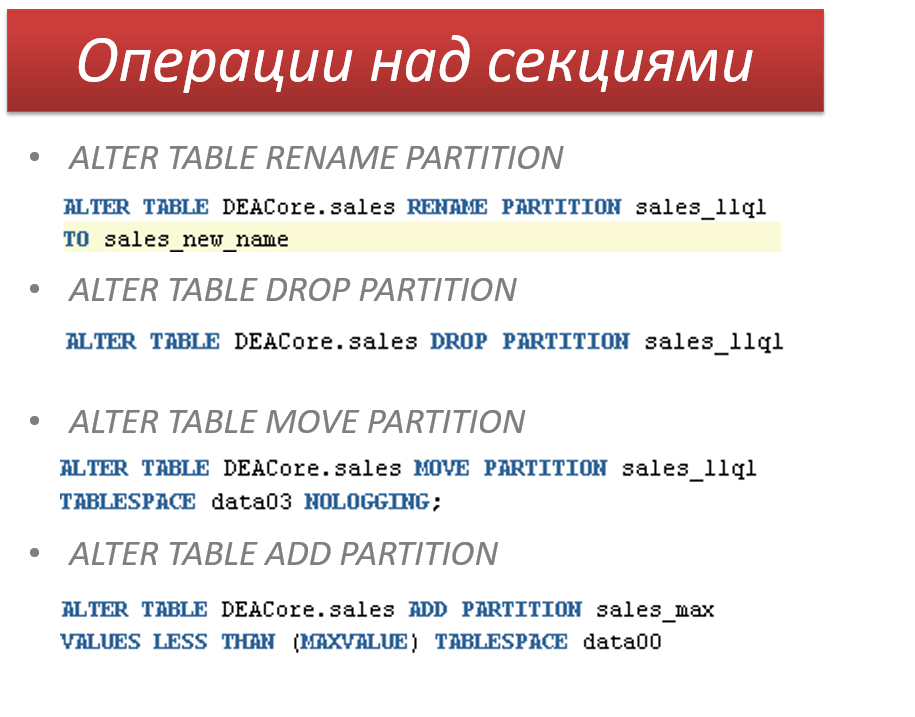
partition T\_LIST\_q2 **values ('i')**

);

1. **Композитное – 2 уровня, секции и подсекции**
2. **Эквисекционирование** – секционирование объектов одинаковым образом, одинаковое количество секций, метод секционирования, ключи и границы
3. **Ссылочное** – если таблицы связаны FK, метод и ключ секционирования наследуются из главной таблицы, partition by reference
4. **По виртуальному столбцу**

**Виртуальный столбец – значение вычисляется на лету (динамически), детерминистически (не зависит от времени). ВИРТУАЛЬНЫЙ СТОЛБЕЦ НЕ МОЖЕТ ССЫЛАТЬСЯ НА ВИРТУАЛЬНЫЙ!!!!!!!**

**Виртуальными могут быть столбцы только в обычных таблицах. ВИРТУАЛЬНЫЙ СТОЛБЕЦ ВЫЧИСЛЯЕТСЯ ПО ОБЫЧНЫМ СТОЛБЦАМ!!!!**





**У 1 таблицы есть секция и данные**

**У 2 нет секции и есть данные**

**Обмен секциями – во 2 появятся данные из секции 1ой таблицы, а первая таблица получит данные 2ой**

**И теперь в секции 1ой таблицы данные из 2ой**

* Для ускорения доступа таблицы могут индексироваться
* Индексы могут секционироваться

1. Транзакции. Виды транзакций. Понятие автономной транзакции.

**Транзакция – это логическая единица работы в базе данных Oracle, состоящая из одного или более операторов SQL.** Транзакция начинается с первого исполняемого опертартора SQL и завершается, когда вы фиксируетет или отказываете транзакцию. Фиксация (commiting) транзакции закрепляет проведенные вами изменения, а откат (roll back) – конечно же, отменяет их. Как только вы зафиксировали транзакцию, все прочие транзакции других пользователей, которые начались после нее, смогут видеть изменения, проведенные вашими транзакциями.

**Транзакция начинается немедленно в начале первого исполняемого оператора SQL и продолжается во время выполнения всех последующих операторов SQL до тех пор,пока не произойдет одно из перечисленных ниже событий.**

* **COMMIT**. Если транзакция встречает оператор COMMIT, все изменения, произведенные до этой точки, фиксируются в базе данных и становятся постоянными.
* **ROLLBACK**. Если транзакция встречает оператор ROLLBACK, то все изменения, произведенные до этой точки, отменяются.
* **Оператор DDL. Если пользователь запускает оператор DDL, такой как CREATE,DROP, RENAME или ALTER, то, прежде чем выполнить и зафиксировать результаты оператора DDL, Oracle сначала фиксирует все текущие операторы DML, которые являются частью транзакции.** Это называется неявной фиксацией (implicit commit),поскольку фиксация операторов DML, предшествующих операторам DDL, не выполняется пользователем явно.
* **Нормальное завершение программы. Если программа завершается без ошибок, то все изменения неявно фиксируются в базе данных.** В случае нормального “чистого” выхода из SQL\*PLus база данных автоматически фиксирует все изменения,которые были произведены в данных на протяжении сеанса.
* **Ненормальное завершение программы. Если программа терпит крах или прерывается, то все проведенные ею изменения неявно отменяются базой данных**.

**Оператор *COMMIT* успешно завершает транзакцию. Все изменения, произведенные всеми операторами SQL с начала транзакции, записываются на постоянное хранение в базу данных**. До выдачи оператора *COMMIT* изменения в данных другим транзакциям не видны.

Когда транзакция Oracle фиксируется, происходят описанные ниже события.

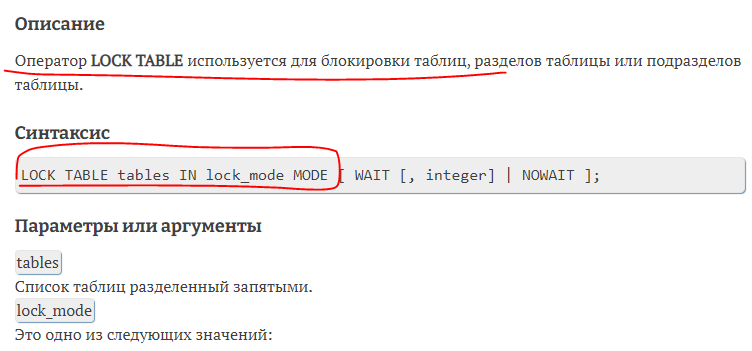
1. **Таблицы транзакций в записях повторного выполнения помечаются уникальным системным номером изменения (system change number — SCN) зафиксированной транзакции.**

2. **Процесс-писатель журнала записывает информацию в журнал повторного выполнения на диске из буфера журнала повторного выполнения, вместе с SCN транзакции**. В этот момент фиксация считается завершенной в Oracle.

3. **Все удерживаемые Oracle блокировки освобождаются, и Oracle помечает транзакцию как зафиксированную.**

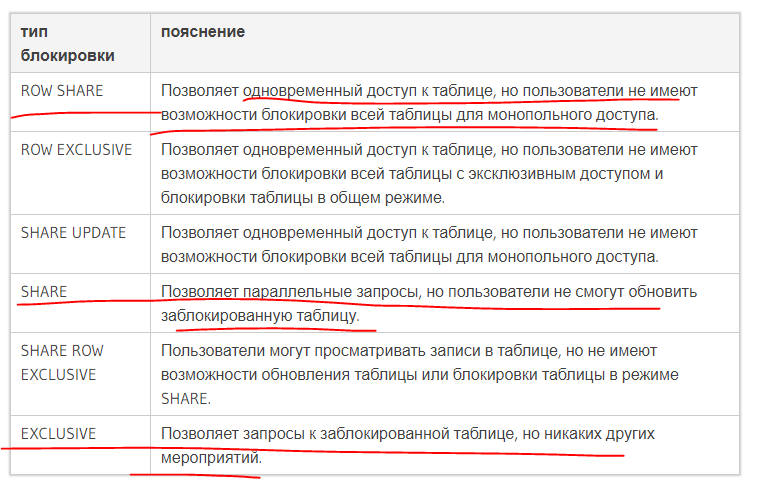
**Оператор *ROLLBACK* отменяет, или откатывает, изменения, внесенные операторами SQL внутри транзакции, которая еще не была зафиксирована. После выдачи оператора *ROLLBACK* ни одно из изменений, проведенных в таблицах операторами SQL с момента начала транзакции, в базу данных не заносится.**

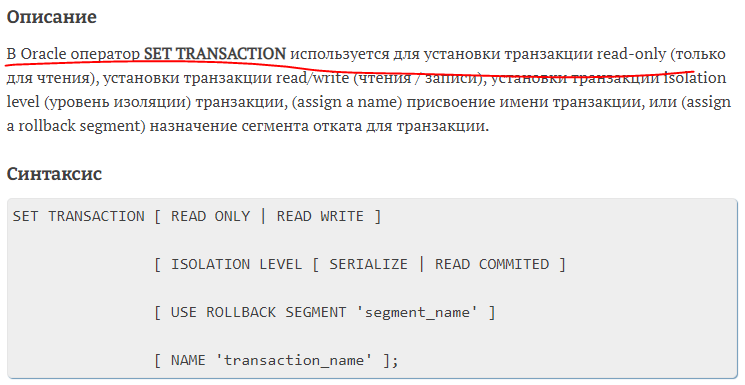
* **Атомарность. Вся транзакция должна либо выполниться целиком, либо не выполниться вообще**.
* **Согласованность. База данных всегда должна пребывать в согласованном состоянии.** Например, в банковской транзакции, включающей снятие денег с депозитного счета и перенос на текущий счет, база не может просто добавить деньги на текущий счет и остановиться. Это привело бы к несогласованным данным, и свойство согласованности транзакций гарантирует, что база никогда не оставит данные в таком несогласованном состоянии. Все транзакции должны предохранять согласованность базы данных. Например, если требуется удалить идентификатор отдела из таблицы Department, то база данных не должна допустить такого действия, если в таблице Employees имеются сотрудники, относящиеся к удаляемому отделу.
* **Изоляция. Изоляция означает, что несмотря на параллельный доступ к базе данных множества транзакций, каждая из них должна проходить изолированно от остальных. Свойство изоляции транзакций гарантирует, что транзакция не увидит изменений, внесенных другими транзакциями, прежде чем первая транзакция будет зафиксирована**.
* **Вечность. Последнее из свойств ACID — вечность — гарантирует, что база данных сохранит результаты зафиксированной транзакции на постоянной основе. Как только транзакция завершена, база данных должна гарантировать, что ее результаты не будут потеряны.** Это свойство обеспечивается механизмами восстановления базы данных, гарантируя сохранность всех зафиксированных транзакций. Oracle использует протокол опережающей записи, который обеспечивает **первоначальную**[**запись всех изменений в журналы**](https://oracle-patches.com/oracle/prof/2959-%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F-%D0%B1%D0%B0%D0%B7%D1%8B-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-oracle)**повторного выполнения на диск до их переноса в файлы на диске.**

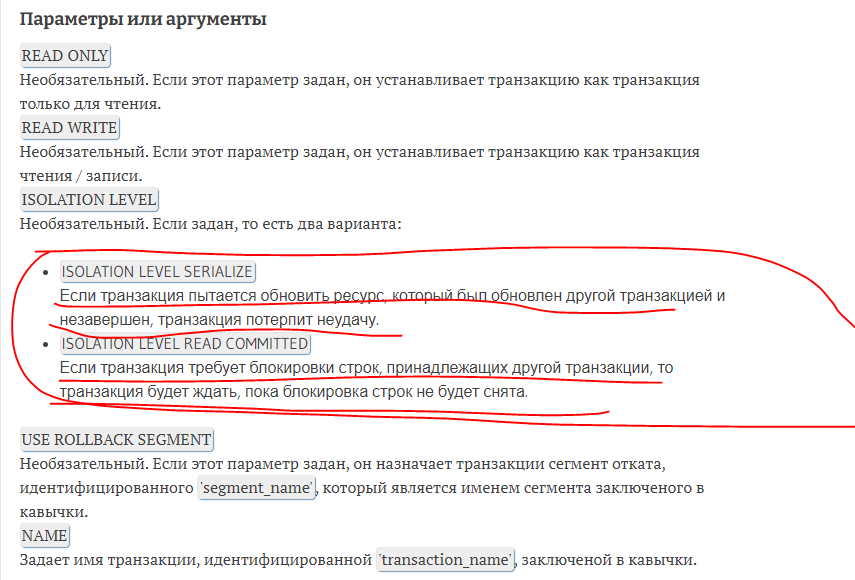


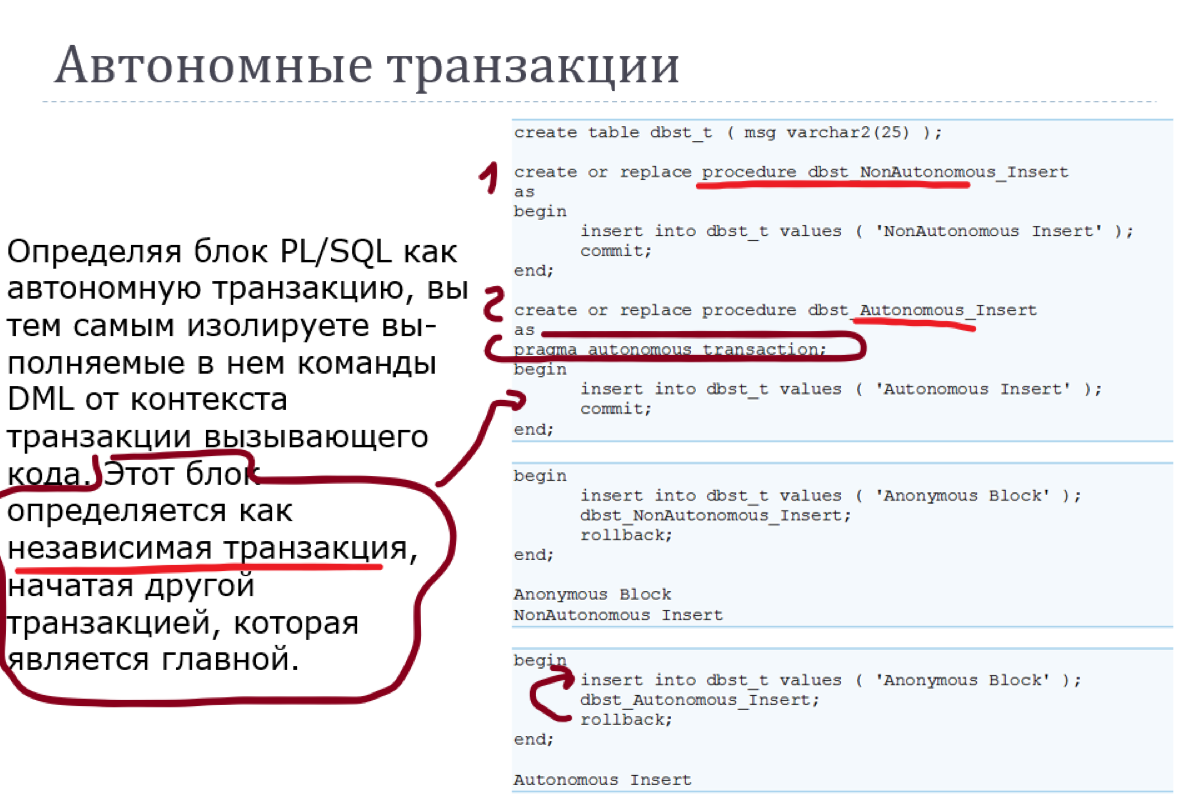
LOCK TABLE suppliers IN *SHARE(см таблицу)* MODE NOWAIT;

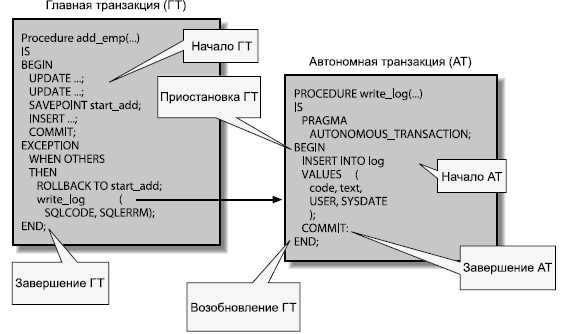
* WAIT  
  Он указывает на то, что база данных будет ждать (до определенного количества секунд, как это определено целым числом), чтобы получить блокировку DML.
* NOWAIT  
  Он указывает, что база данных не должна ждать, пока блокировка будет снята.











* В блоке автономной транзакции главная транзакция приостанавливается. Вы выполня­ете [SQL-операции](https://oracle-patches.com/db/sql/3542-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B-ddl-%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0-pl-sql-%D0%BD%D0%B0-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85), затем производите их фиксацию или откат и возобновляете главную [транзакцию](https://oracle-patches.com/db/sql/3543-%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F%D0%BC%D0%B8-%D0%B2-pl-sql)

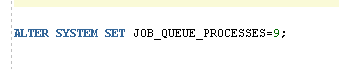
1. Обработка заданий. Системные пакеты обработки заданий в Oracle.

**DBMS\_JOB – поддержка управления заданиями. ПАКЕТ ПЛАНИРОВЩИКА ЗАДАНИЙ!!!!!!**

Задание – процедура, Pl-SQL блок, внешняя процедура

Выполняется в фоновом режиме, надо задать количество одновременно выполняемых процессов

**Пакет DBMS JOB позволяет запланировать однократное или регулярное выполнение заданий в базе данных. Задание представляет собой хранимую процедуру, анонимный блок PL/SQL или внешнюю процедуру на языке С или Java. Эти задания выполняются серверными процессами в фоновом режиме**

****

* SUBMIT – создание задания – АВТОНУМЕРАЦИЯ
* ISUBMIT – создание задания с номером

**dbms\_job.isubmit(номер, действие, когда запустить первый раз, интервал);**

* REMOVE – удаление задания из очереди
* RUN – немедленное выполнение задания в пользовательском сеансе

**Run (job, force – true/false)**

* BROKEN – разрушение задания (16 раз не смогло выполниться - разрушение)

**dbms\_job.broken(jobnumber, true, null);**

**задача – принудительно – следующий запуск**

* INSTANCE – указание экземпляра выполнения
* NEXT\_DATE – изменение времени выполнения – указать следующую дату выполнения

dbms\_job.next\_date(jobnumber, SYSDATE + interval '5' minute);

* INTERVAL – изменение интервала выполнения
* CHANGE – изменение параметров задания

dbms\_job.change(jobnumber(КАКАЯ ЗАДАЧА), null(ЧТО ДЕЛАЕТ), null(NEXT\_DATE), 'SYSDATE + interval ''2'' minute' (ИНТЕРВАЛ), instance, force);

* WHAT – изменение задания

select \* from dba\_jobs;

select \* from dba\_jobs\_running;

dbms\_job.isubmit(jobnumber, - номер

action, - что

SYSDATE, - запустить сразу

'SYSDATE + interval ''1'' minute' – следующий раз запуска

);

**DBMS\_SCHEDULER – комбинирует расписание + программа + job задача**

**Job = SCHEDULER + PROGRAM**

Schedule – расписание

Program - программа

Job – плановая программа

**dbms\_scheduler.create\_schedule(**

schedule\_name => 'Sch01',

start\_date => SYSTIMESTAMP,

repeat\_interval => 'FREQ=MINUTELY;INTERVAL=1;',

comments => 'Sch01 Minutely');

**dbms\_scheduler.create\_program(**

program\_name => 'Pr01',

program\_type => 'STORED\_PROCEDURE',

program\_action => 'SCHEDULPROC',

number\_of\_arguments => 2,

enabled => false,

comments => 'SCHEDULPROC');

**dbms\_scheduler.define\_program\_argument(**

program\_name => 'Pr01',

argument\_name => 'startdate',

argument\_position => 1,

argument\_type => 'DATE',

default\_value => null);

**dbms\_scheduler.define\_program\_argument(**

program\_name => 'Pr01',

argument\_name => 'enddate',

argument\_position => 2,

argument\_type => 'DATE',

default\_value => null);

**dbms\_scheduler.enable (name => 'Pr01');**

**dbms\_scheduler.create\_job(**

job\_name => jobname,

program\_name => 'Pr01',

schedule\_name => 'Sch01',

enabled => true);

dbms\_scheduler.set\_job\_argument\_value(job\_name => jobname, argument\_position => 1, argument\_value =>to\_date('01.12.2022','DD.MM.YYYY'));

dbms\_scheduler.set\_job\_argument\_value(job\_name => jobname, argument\_position => 2, argument\_value =>to\_date('18.12.2022','DD.MM.YYYY'));

dbms\_scheduler.set\_attribute(jobname, 'logging\_level', DBMS\_SCHEDULER.LOGGING\_FULL);

Изменение расписания:

**dbms\_scheduler.set\_attribute**(name => 'SCH01', attribute => 'repeat\_interval', value => 'FREQ=MINUTELY;INTERVAL=2;');

dbms\_scheduler.**disable**(jobname);

dbms\_scheduler.**enable**(jobname);

dbms\_scheduler.**stop\_job**(job\_name => jobname);

select \* from user\_scheduler\_schedules;

select \* from user\_scheduler\_programs;

select \* from user\_scheduler\_jobs;

select \* from user\_scheduler\_job\_log;

select \* from user\_scheduler\_job\_run\_details;

select \* from user\_scheduler\_running\_jobs;

begin

dbms\_scheduler.**drop\_job**(job\_name => 'ARCH\_WORKERS\_JOB');

end;

begin

dbms\_scheduler.**drop\_program**(program\_name => 'PR01');

end;

begin

dbms\_scheduler.**drop\_schedule**(schedule\_name => 'SCH01');

end;

1. Системные пакеты Oracle.

Встроенные системные пакеты:

* Устанавливаются во время установки Oracle
* Используются для расширения основных функциональных возможностей Oracle
* Владелец пакетов - SYS (обычно, есть и другие)
* Написаны на C или на PL/SQL

Пакеты APEX:

**- apex – oracle application express – среда разработки веб-приложений**

APEX\_CUSTOM\_AUTH –проверка подлинности и управления сеансом

APEX\_APPLICATION – использование глобальных переменных

APEX\_ITEM – создание элементов форм на основе SQL-запроса

APEX\_UTIL – различные утилиты состояния сеанса, файлов, авторизации и пр.

**Пакеты DBMS:**

- dbms\_advanced\_rewrite – перехват и замена sql-запросов

- dbms\_advisor – часть набора экспертной системы для решения проблем производительности, связанных с компонентами сервера БД

- dbms\_sqltune – сбор статистики, используется при анализе производительности операторов

- dbms\_appl\_info – присвоение имени процессу для удобства мониторинга и отладки

- DBMS\_AQ – система обмена очередями сообщений

- DBMS\_DEFER – вызов удаленных процедур

- DBMS\_BACKUP\_RESTORE

- DBMS\_CRYPTO

- DBMS\_DEBUG

- DBMS\_DESCRIBE

- DBMS\_FILE\_TRANSFER – пересылка файлов между файловыми системами

- DBMS\_JAVA

- DBMS\_JOB

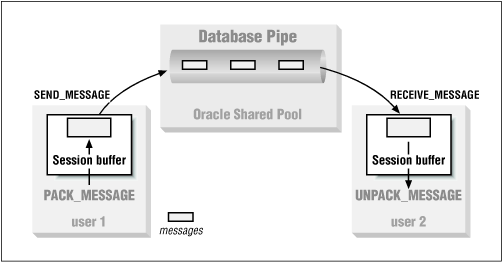
- DBMS\_SCHEDULER

- DBMS\_LOCK - блокировки

- DBMS\_LOB

- DMBS\_OUTPUT серверный вывод

- DBMS\_PIPE



- DBMS\_RANDOM

- DBMS\_ROWID

- DBMS\_SHARED\_POOL

DBMS\_TRANSACTION – пакет управления транзакциями

* Read only
* Read write
* \*Set isolation level read committed
* \*Set isolation level serializable
* \*Lock table

