



Базы данных

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

Дистрибутивы:

1. Oracle Database Standard Edition: это редакция Oracle Database начального уровня, предназначенная для малого и среднего бизнеса. Он включает в себя большинство функций Enterprise Edition, но ограничен одним сервером с максимум двумя процессорами.
2. Oracle Database Enterprise Edition: это флагманская версия Oracle Database, разработанная для крупных предприятий и предприятий с высокими требованиями к рабочим нагрузкам. Он включает в себя все функции Standard Edition, а также расширенные возможности для обеспечения высокой производительности, масштабируемости и безопасности.
3. Oracle Database Express Edition: это бесплатная версия Oracle Database начального уровня, предназначенная для малого бизнеса, разработчиков и студентов. Он включает в себя подмножество функций, доступных в выпусках Standard и Enterprise, и ограничен одним ЦП и 1 ГБ ОЗУ.
4. Oracle Database Cloud Service: это облачная версия Oracle Database, предлагаемая по подписке через Oracle Cloud. Он доступен как в стандартной, так и в корпоративной версиях, и его можно легко увеличивать или уменьшать в соответствии с меняющимися требованиями к рабочей нагрузке.
5. Oracle Database Appliance: это предварительно настроенная, полностью интегрированная аппаратно-программная система, включающая Oracle Database, серверное оборудование и хранилище. Он предназначен для предприятий малого и среднего бизнеса, которым требуется простое в использовании готовое решение для работы с Oracle Database.

Установка:

1. почта на которую будут приходить сообщения о безопасности или обновлениях;
2. автоматически устанавливать обновления или нет;
3. создать и настроить бд / установить бд / обновить бд;
4. desktop / server class;
5. single instance / real application clusters / RAC one node;
6. typical / advanced;
7. создание пользователя Windows;
8. путь установки.

Системный идентификатор (SID) — это уникальное имя, идентифицирующее конкретный экземпляр Oracle.

Глобальное имя базы данных — это уникальное имя, которое идентифицирует базу данных в сети. Он состоит из имени службы базы данных и доменного имени. Например, если имя службы базы данных — «sales», а имя домена — «example.com», то глобальное имя базы данных будет «sales.example.com».

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

Основные пользователи:

1. SYS: предопределённый привилегированный пользователь ранга администратора базы данных, который является владельцем ключевых ресурсов БД. Этот пользователь создается автоматически при установке Oracle Database.
2. SYSTEM: предопределённый привилегированный пользователь, которому принадлежат ключевые ресурсы БД.

В Oracle Database системная привилегия SYSDBA позволяет пользователю выполнять определенные административные задачи, такие как запуск и остановка базы данных, создание и удаление баз данных, а также резервное копирование и восстановление базы данных. Другие системные привилегии:

1. SYSOPER: Эта привилегия позволяет пользователю выполнять определенные рабочие задачи, такие как запуск и остановка базы данных, но не позволяет ему выполнять такие задачи, как создание и удаление баз данных.
2. SYSDG: эта привилегия позволяет пользователю выполнять задачи, связанные с защитой данных, например создавать и поддерживать конфигурацию защиты данных.
3. SYSBACKUP: Эта привилегия позволяет пользователю выполнять задачи резервного копирования и восстановления, такие как создание резервных копий базы данных и управление ими.
4. SYSKM: Эта привилегия позволяет пользователю выполнять задачи, связанные с шифрованием базы данных, такие как создание ключей шифрования и управление ими.

DBA — предопределённая роль, которая автоматически создаётся для каждой базы данных Oracle, содержит все системные привилегии, кроме SYSDBA и SYSOPER.

3. Понятия базы данных и экземпляра базы данных.

В Oracle Database база данных представляет собой набор данных, организованных в логические структуры (например, таблицы, представления, процедуры), доступ к которым и управление которыми могут получать пользователи и приложения.

Экземпляр базы данных — это работающий экземпляр программного обеспечения Oracle Database, который обращается к данным в базе данных и управляет ими. Когда вы запускаете программное обеспечение Oracle Database, оно создает экземпляр базы данных и назначает ему уникальный системный идентификатор (SID). Экземпляр обрабатывает запросы пользователей и приложений на доступ и изменение данных в базе данных, а также управляет данными в файлах базы данных на диске.

База данных может иметь несколько экземпляров базы данных, каждый со своим собственным SID, работающих одновременно. Это может быть полезно для управления рабочей нагрузкой, аварийного переключения и других целей. Однако каждый экземпляр может одновременно обращаться только к одной базе данных.

В Oracle Database база данных и экземпляр базы данных — это два отдельных понятия. База данных — это логическая структура, в которой хранятся данные, а экземпляр базы данных — это работающее программное обеспечение, которое обращается к данным в базе данных и манипулирует ими.

4. Запуск и останов экземпляра базы данных Oracle.

Для запуска или остановки экземпляра должно использоваться подключение с разрешением SYSDBA или SYSOPER.

Запуск:

1. NOMOUNT: этот режим запускает экземпляр базы данных, но не монтирует базу данных. Это полезно для таких задач, как создание новой базы данных или изменение параметров инициализации базы данных.
2. MOUNT: этот режим запускает экземпляр базы данных и монтирует базу данных, но не открывает ее для использования. Это полезно для выполнения таких задач, как резервное копирование и восстановление, или для запуска сценариев, не требующих открытия базы данных.
3. OPEN: это режим запуска по умолчанию, при котором экземпляр базы данных запускается и монтируется, а сама база данных открывается для использования.
4. RECOVER: этот режим используется для запуска экземпляра базы данных в режиме восстановления, для восстановления базы данных из резервной копии или для применения файлов журнала повторного выполнения.
5. RESTRICTED: этот режим похож на OPEN, но позволяет входить в систему только пользователям с привилегией RESTRICTED SESSION. Это полезно для выполнения задач обслуживания, в то же время запрещая другим пользователям доступ к базе данных.

Пример: STARTUP NOMOUNT/MOUNT/OPEN/RECOVER/RESTRICTED. (перевести из одного состояния в другое: ALTER DATABASE OPEN/...)

*Монтирование БД - это процесс, в ходе которого сервер базы данных Oracle сопоставляет файлы данных и индексов с экземпляром базы данных и подготавливает их к работе. При монтировании базы данных сервер базы данных

проверяет целостность файлов данных и индексов, но не открывает их для доступа.)

Останов:

- SHUTDOWN NORMAL Запрещено создавать новые сессии. Ожидается завершение работы всех пользователей. Самый безопасный и долгий способ останова. Никаких восстановительных работ при следующем старте не проводится.
- SHUTDOWN TRANSACTIONAL Запрещено создавать новые сессии. Запрещено запускать новые транзакции. Сервер дожидается завершения уже начатых транзакций и отключает пользователей, не имеющих активных транзакций. Применяется в случаях, когда нет возможности применить NORMAL. Никаких восстановительных работ при следующем старте не проводится.
- SHUTDOWN IMMEDIATE Запрещено создавать новые сессии. Запрещено запускать новые транзакции. Все незафиксированные транзакции откатываются. Применяется в случаях, когда нет возможности ждать. Никаких восстановительных работ при следующем старте не проводится.
- SHUTDOWN ABORT Применяется в крайних случаях, когда остальные режимы останова не приводят к результату. Все действия прекращаются. Все транзакции не фиксируются и не откатываются. Пользователей отсоединяют от БД. При следующем старте будет выполнено возможное восстановление.

	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Новые соединения	-	-	-	-
Ждать конца сеанса	-	-	-	+
Ждать конца транзакций	-	-	+	+
Задачи по очистке	-	+	+	+

Пример: SHUTDOWN ABORT/IMMEDIATE/TRANSACTIONAL/NORMAL

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

Словарь базы данных представляет собой набор таблиц и представлений в базе данных Oracle, в которых хранятся метаданные о самой базе данных. Словарь базы данных используется ядром базы данных и другими инструментами для доступа и управления структурой и содержимым базы данных.

Цель словаря базы данных — предоставить согласованный и централизованный источник информации о базе данных, включая структуру объектов базы данных (например, таблицы, представления, индексы), привилегии и роли, предоставленные пользователям, а также производительность и так далее.

В Oracle Database представления, начинающиеся с «USER_», показывают объекты, принадлежащие текущему пользователю.

Представления, начинающиеся с «ALL_», показывают все объекты в базе данных, к которым имеет доступ текущий пользователь.

Представления, начинающиеся с «DBA_», отображают все объекты в базе данных, независимо от владельца. Эти представления могут быть доступны только пользователям с ролью DBA или другим пользователям с достаточными привилегиями.

Представления, начинающиеся с «V\$», представляют собой динамические представления, отображающие текущее состояние и производительность экземпляра базы данных.

6. Мультиарендная архитектура Oracle Multitenant.

Oracle Multitenant - это архитектура, позволяющая размещать несколько баз данных в одной физической инстанции базы данных Oracle. Каждая база данных в этой архитектуре называется контейнером, а общие ресурсы экземпляра базы данных, такие как буферный кэш и процессы, разделяются между всеми контейнерами. Эта архитектура позволяет уменьшить накладные расходы на управление несколькими базами данных, а также упрощает обновление и обслуживание баз данных.

CDB (Container Database) — это база данных Oracle, которая содержит несколько контейнеров (PDB, Pluggable Database). CDB содержит также общую SGA и общий набор серверных процессов, которые разделяются между всеми контейнерами.

CDB\$ROOT – главный контейнер, который содержит метаданные CDB. Этот контейнер копируется при создании PDB («юзается как изначальный шаблон короче» ©Какой-то чел из 6 группы).

PDB (Pluggable Database) — это отдельная база данных, которая может быть создана в CDB и храниться в одном физическом файле с CDB. Каждый PDB имеет свой набор табличных пространств и набор схем; они изолированные между друг другом, не конфликтуют. Однако имеют общую SGA и один набор серверных процессов. Словарь разбивается на две части: общую часть и локальную(локальный очевидно для отдельной PDB)

Можно создавать несколько CDB. Одна CDB вмещает 252 PDB. PDB можно елозить между CDB (Елозить = перемещаться)

Если не понятно зачем CDB – Oracle дурачки и у них нельзя было создавать несколько БД в рамках одного инстанса, и чтобы не переписывать херову тучу кода решили костылем создавать БД внутри БД (мб не совсем так, но примерно)

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

Архитектура внешней памяти состоит из ЭКЗЕМПЛЯРА (ФАЙЛ ПАРАМЕТРОВ) и БАЗЫ ДАННЫХ (ФАЙЛЫ данных, файлы журнала повторного выполнения, управляющие файлы, временные файлы, файлы паролей).

Параметры:

```
select * from v$parameter;
```

Изменение spfile:

```
alter system set open_cursors = 350 scope = spfile;
```

Создать текстовый файл из бинарного:

```
create pfile 'p1.ora' from spfile;
```

Сформировать бинарный файл SPFILE параметров из текстового файла PFILE:

```
create spfile = 'sp.ora' from pfile = 'p1.ora'
```

Порядок поиска:

1. spfileORACLE_SID.ora;
2. spfile.ora;
3. initORACLE_SID.ora.

Управляющие файлы (control files) — файлы, содержащие имена и местоположение основных физических файлов БД и некоторых параметров. Обычно в экземпляре базы данных Oracle используется несколько управляющих файлов, что позволяет избежать потери информации в случае отказа одного из файлов. Все управляющие файлы расположены в папке datafiles (View-connections)

Местоположение:

Platform	Default Location
UNIX and Linux	ORACLE_HOME/dbs
Windows	ORACLE_HOME\database

Изменение управляющих файлов:

- Остановить Oracle
- скопировать один из управляющих файлов
- изменить параметр CONTROL_FILES В ФАЙЛЕ ПАРАМЕТРОВ
- стартовать ORACLE

```
select * from v$controlfile;
```

```
alter database backup controlfile to trace;
```


Файлы паролей (password files) — это специальные файлы, содержащие хэши паролей для пользователей с системными привилегиями. Эти файлы используются для проверки аутентичности пользователей, которые пытаются подключиться к экземпляру базы данных с использованием системных привилегий. Если этот файл отсутствует, то вы можете выполнять администрирование своей базы данных только локально.

```
select * from v$pwfile_users;
```

Файлы трассировки (trace files) — это специальные файлы, содержащие информацию об ошибках и предупреждениях, которые были обнаружены во время работы экземпляра базы данных Oracle. Файлы трассировки могут быть использованы для диагностики и устранения проблем с экземпляром базы данных. Информация, содержащаяся в файлах трассировки, может включать в себя SQL-запросы, параметры памяти, информацию об ошибках и т. д.

```
select * from v$diag_info; -- Там найти Diag Trace или Diag Alert
```

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

Файлы данных (data files) — это файлы, содержащие данные базы данных. В экземпляре базы данных Oracle обычно используется несколько файлов данных, что позволяет разделить данные на несколько физических файлов. У любой БД как минимум один файл данных, определённый файл данных может относиться только к одной БД.

```
select * from dba_data_files;
```

Журналы повтора (redolog files) — это файлы, в которых фиксируются изменения, вносимые пользователями в базу данных. Журналы служат для резервного копирования и восстановления базы данных, а также для реализации транзакций. Каждая база данных должна иметь не менее двух журналов повтора.

Текущий файл постепенно заполняется, после его заполнения (или переключения некоторыми командами), база данных приступает к записи в следующий файл. Эта операция называется переключением журналов — она происходит циклически.

(ДЛЯ СПРАВКИ: Когда файлы журнала используются циклически, это означает, что они перезаписываются снова и снова по мере того, как накапливается новая информация. Обычно это означает, что есть несколько файлов журнала, и один из них используется для записи новых данных, в то время как другие файлы журнала используются для хранения старых данных. Когда один файл журнала заполняется, система начинает использовать следующий файл журнала и так далее.)

```
select * from v$logfile;
```

Мультиплексирование журналов повтора (redo logs) — поддержка нескольких копий каждого журнала.

Мультиплексирование журналов повтора (redo logs) — это процесс разделения журналов повтора на несколько физических файлов, которые могут храниться на разных дисках. Мультиплексирование журналов повтора используется для увеличения производительности и уменьшения риска потери данных в случае отказа диска. Когда один из файлов журнала повтора заполнен, процесс переключается на следующий файл журнала повтора. Таким образом, журналы повтора постоянно циклически перезаписываются, пока экземпляр базы данных работает. В случае отказа диска, система может использовать журналы повтора для восстановления данных, так как они содержат информацию о всех изменениях, внесенных в базу данных.

<<< Два стула

```
select * from v$log;

-- Переключение группы:
alter system switch logfile;

-- Удаление файла:
alter database drop logfile member 'name';

-- Удаление группы:
```

```

alter database drop logfile group 4;

-- При удалении текущая группа должна быть не current

-- Добавление группы:
alter database add logfile group 4 'name' size 50m blocksize 512;

-- Добавление файла:
alter database add logfile member 'name' to group 4;

```

Архивы (archive logs) — это файлы, содержащие сообщения об изменениях, внесенных в базу данных с момента последнего резервного копирования. Архивы служат для резервного копирования и восстановления базы данных, а также для обеспечения целостности.

Если необходимо сохранить историю изменений, нужно, чтобы после переключения журналов сохранялась их копия. Для этого достаточно перевести работу базы данных в режим работы ARCHIVELOG.

Архивные файлы журналов повтора жизненно важны при восстановлении. Если часть базы данных потеряна или повреждена, то для устранения повреждений обычно требуется несколько архивных журналов. Файлы журналов повтора должны применяться к базе данных последовательно. Если один из архивных файлов журналов повтора пропущен, то остальные архивные файлы журналов не могут использоваться.

```

-- Посмотреть архивирование
select name, log_mode from v$database;
select instance_name, archiver from v$instance;

-- Включение
shutdown immediate
startup mount
alter database archivelog
alter database open

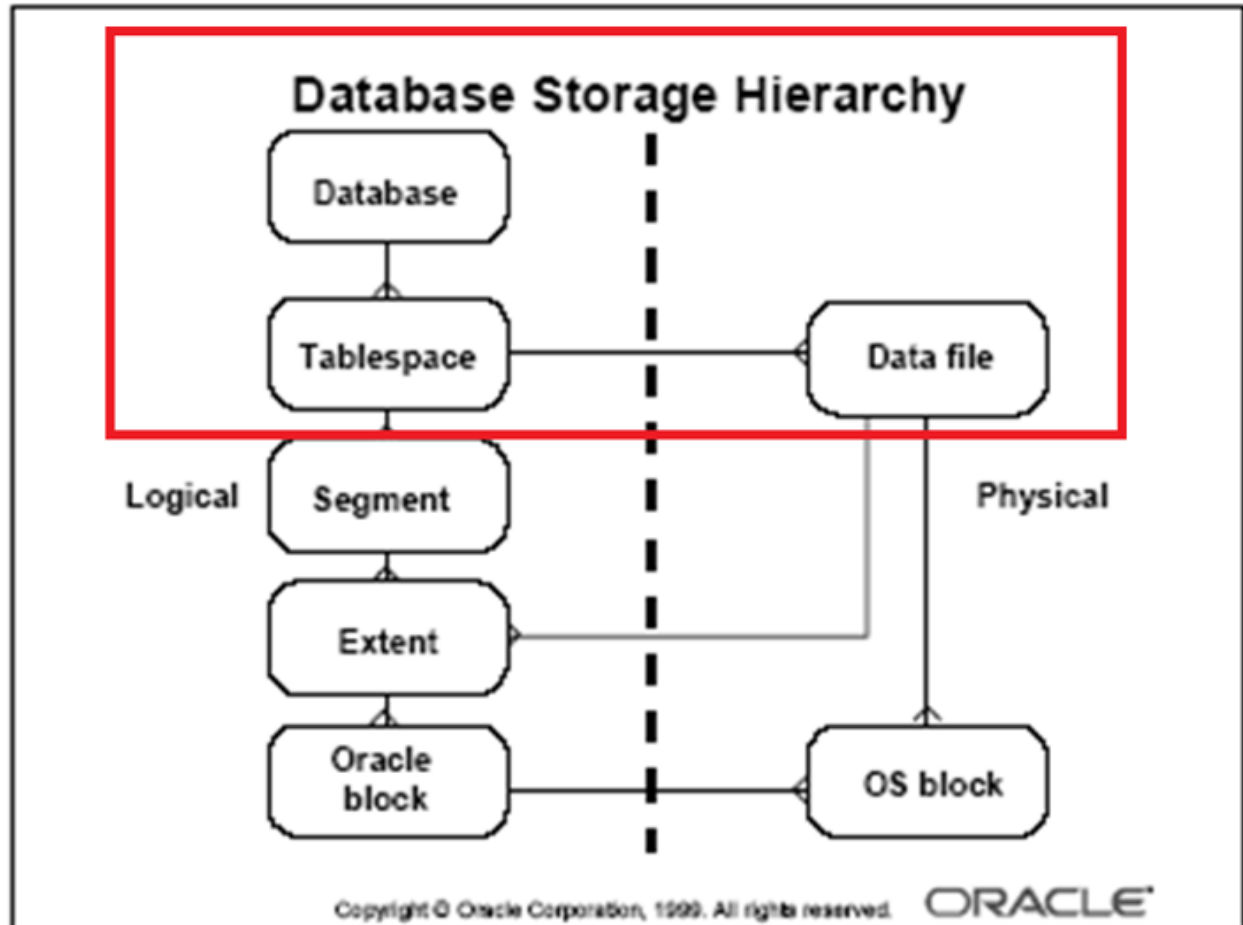
-- Выключение такое же, но noarchivelog

```

Архивы создаются после переключения журналов повтора.

9. Абстрактная модель Oracle. Логическая структура внешней памяти.

В Oracle существует **двухуровневая организация базы данных**, в которой физическая структура базы данных отделена от логической структуры. Физическая структура базы данных состоит из файлов данных, журналов и других файлов, которые хранятся на диске. Логическая структура базы данных состоит из таблиц, индексов и других объектов, которые определяют, как данные будут храниться и использоваться.



Логическая структура состоит из :

- **Табличное пространство (tablespace)** — это логическая структура, которая служит для хранения объектов базы данных, таких как таблицы, индексы и процедуры. Каждое табличное пространство состоит из одного или нескольких сегментов, которые физически хранят данные. С одним табличным пространством связаны один или несколько файлов, с каждым файлом связано только одно табличное пространство. Данные, временные данные, данные отката – организованы в виде табличных пространств.

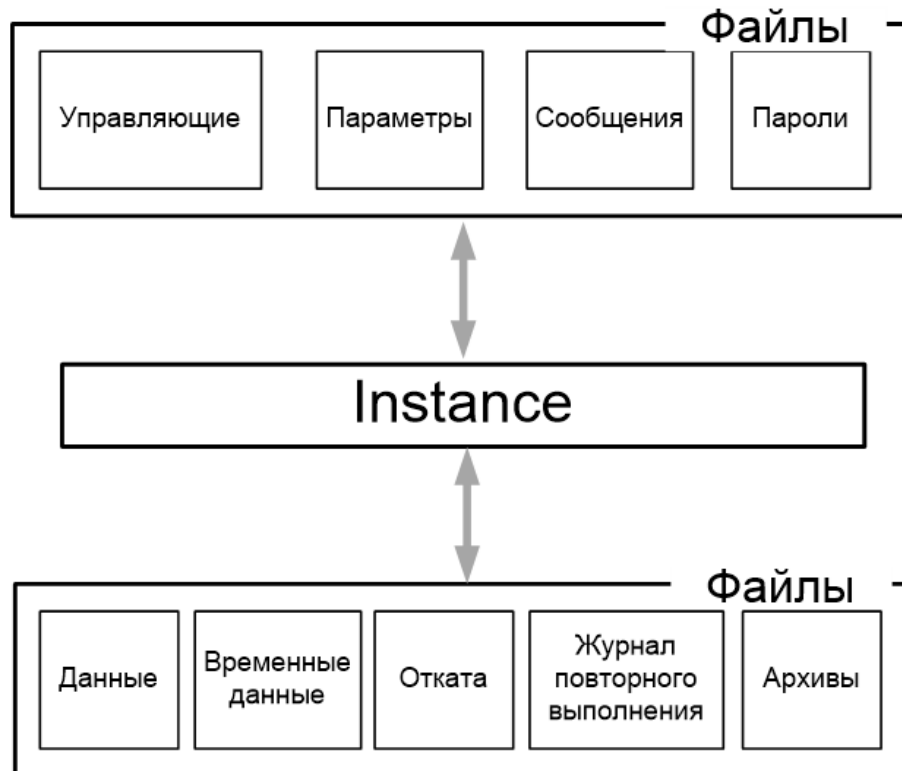
- **Сегмент** (segment) — это логическая структура, которая содержит таблицы, индексы или другие объекты базы данных. Каждый сегмент состоит из экстентов, которые физически хранят данные. Сегменты типизируются в зависимости от типа данных, хранящихся в них – сегменты таблиц, сегменты индексов, сегменты кластеров и т.д. (всего 10 типов).
- **Экстенты** — это небольшие группы размещенных непрерывно физических блоков внешней памяти, которые используются для хранения таблиц и индексов. Экстент состоит из блоков данных. Является единицей выделения вторичной памяти (выделяется целым числом экстентов). Когда экстент заполняется выделяется следующий. Размер экстента варьируется от одного блока до 2 Гб.
- **Блоки данных** — являются наименьшими размерными единицами хранения данных в БД Oracle. Блоки данных содержат реальные данные и управляющую информацию, такую как указатели на следующий блок и т. д. Размер кратен 2К, и должен быть кратен величине блока операционной системы (2К, 4К, 8К, допустимы 16К, 32К). Устанавливается в файле параметров экземпляра при создании БД.

В табличном пространстве все блоки одного размера. Сегмент состоит из одного и более экстентов. Экстент состоит из идущих подряд блоков.

10. Абстрактная модель Oracle. Физическая структура внешней памяти.

Основные компоненты: файлы, образующие базу данных и поддерживающие экземпляр — файлы параметров, сообщений, данных, временных данных и журналов повторов;

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.



Существуют две группы компонентов физического уровня :

1. Системные объекты — Spfile, Controlfile, файлы трассировки, архивы
2. Объекты юзера — файлы данных

11. Абстрактная модель Oracle. Структура SGA.

SGA (Глобальная область системы) — это область памяти в базе данных Oracle, которая используется для хранения данных и управляющей информации для экземпляра базы данных. Это область общей памяти, которая создается при запуске экземпляра Oracle и используется для хранения различных структур данных, включая следующие:

- Буферный пул — область SGA, которая содержит образы блоков, считанные из файлов данные или созданные динамически, чтобы реализовать модель согласованного чтения. Может содержать грязные или чистые блоки данных. Грязные блоки записываются на диск в следующих случаях:
 - истечение тайм-аута (3 сек);
 - контрольная точка;

- превышение длины грязных блоков заданного лимита;
- процесс не может обнаружить свободный блок.

Также он содержит следующие пулы:

- default (db_cache_size);
- keep (db_keep_cache_size);
- recycle (db_recycle_cache_size).

Просмотр размеров пулов: `select * from v$sga_dynamic_components where component like '%cache%';`

Помещение таблицы в определённый пул: `create table xxx (k int) storage (buffer_pool [имя пула]) tablespace users;`

Если после столбцов указать cache — то таблица будет помещена в конец lru списка.

- Буферы журналов повторов — предназначен для временного циклического хранения данных журналов повтора, содержимое сбрасывается на диск:
 - через каждые 3 секунды;
 - при фиксации транзакции;
 - при заполнении буфера на 1/3;
 - если в буфере более 1 мб данных.
- Фиксированная область SGA: хранит переменные, указывающие на другие области памяти, значения параметров.
- Разделяемый пул: используется для хранения кэша словаря данных, библиотечного кэша, скомпилированных запросов.
- Java pool: предназначен для работы JVM, в нем разворачиваются все необходимые объекты.
- Большой пул: это дополнительная область памяти, которая может использоваться для хранения больших объемов памяти, например, используемых для операций резервного копирования и восстановления.
- Пул потоков: Это дополнительная область памяти, которая может использоваться для хранения информации, связанной с потоками Oracle

LRU (Least Recently Used) - это алгоритм, который используется для управления кэшем в системах с ограниченным объемом памяти. Он основывается на идее, что чаще всего используемые данные будут повторно использоваться в будущем. В системах Oracle, алгоритм LRU используется для управления кэшем буфера (Buffer Cache). Когда база данных Oracle требует данные, она сначала проверяет, есть ли они в кэше буфера. Если да, то Oracle использует данные из кэша, что увеличивает скорость доступа к данным. Если же данные отсутствуют в кэше, то Oracle их загружает с диска и помещает в кэш. При этом алгоритм LRU удаляет самые старые данные из кэша, чтобы освободить место для новых.

Размер SGA и отдельных областей памяти в нем можно настроить с помощью параметров инициализации. SGA является общим для всех серверных и фоновых процессов, и доступ к нему осуществляется ими с помощью указателей.

Файлик для конфигурации – init.ora

Память различным пулам в SGA выдел. блоками – **гранулами** (наименьшая единица выделения памяти)

12. Абстрактная модель Oracle. Серверные процессы Oracle.

Серверные процессы — процессы, выполняющиеся на основании клиентского запроса. База данных Oracle создает серверные процессы, чтобы обрабатывать запросы пользовательских процессов, соединенных с экземпляром. В некоторых ситуациях, когда приложение и БД Oracle работают на том же самом компьютере, возможно объединить пользовательский процесс и соответствующий серверный процесс в единственный процесс, чтобы уменьшить системные издержки. Однако, когда приложение и БД Oracle работают на различных компьютерах, пользовательский процесс всегда связывается с БД Oracle через отдельный серверный процесс.

При отправке sql-запроса серверный процесс:

1. производит синтаксический разбор;
2. помещает запрос в разделяемый пул;

3. создает план запроса и выполняет его;
4. при необходимости обращается в буферный пул за данными.

Для получения списка процессов:

```
select * from v$process;
```

13. Абстрактная модель Oracle. Фоновые процессы Oracle.

Фоновые процессы — специальная группа процессов для обеспечения производительности и поддержки работы большого числа пользователей.

В Oracle существует несколько фоновых процессов, включая следующие:

1. LREG (Listener Registration Process) — периодическая регистрация пользователей в процессе Listener;
2. DBWn (Database Writer) — фоновый процесс, записывающих по LRU изменённые блоки в файлы базы данных, проверяет с периодичностью 3 секунды и освобождает место в буферном кэше;
3. CKPT (Checkpoint) — выполняется при shutdown, alter system checkpoint, переключении REDO. Даёт команду DBW и LGWR на сброс буферов, а также изменяет SCN в управляющих файлах;
4. LGWR (Log Writer) — записывает блоки буфера журналов повтора в группы журналов;
5. ARCn (Archiver) — копирует файлы журнала повтора после переключения группы журналов повтора;
6. PMON (Process Monitor) — отвечает за очистку после ненормального закрытия подключений:
 - a. инициирует откат незафиксированных транзакций;
 - b. освобождает ресурсы SGA;
 - c. следит за работой других фоновых процессов, отвечает за их перезапуск;
 - d. восстанавливает работу dispatcher и shadow процессов.
7. SMON (System Monitor) — восстанавливает экземпляр для узла:

- a. восстановление незавершенных транзакций;
 - b. очистка временных сегментов данных;
 - c. очистка временных табличных пространств;
 - d. очистка таблицы obj;
 - e. сжатие сегментов отката.
8. RECO (Recovery) — разрешение проблем, связанных с распределёнными транзакциями. Когда распределенная транзакция включает несколько баз данных, процесс RECO отвечает за координацию транзакции и обеспечение согласованности изменений, внесенных в данные, во всех базах данных. Это включает в себя отправку и получение сообщений в другие базы данных и из них для подтверждения или отката транзакции, а также разрешение любых конфликтов, которые могут возникнуть.

Для получения списка процессов:

```
select * from v$bgprocess;
```

14. Процесс-слушатель Oracle и его основные параметры.

Oracle Net Listener – процесс на стороне сервера, прослушивающий входящие запросы клиента на соединение с экземпляром.

Listener – это программа-сервер, прослушивающая TCP-порт, принимающая запросы на соединение с Oracle экземпляром от программ-клиентов.

В результате успешной работы Listener устанавливается соединение между программой-клиентом и обработчиком запросов экземпляра.

Процесс прослушивания может быть настроен с использованием различных параметров, которые указаны в файле конфигурации listener.ora. Некоторые из основных параметров, которые можно настроить для процесса прослушивания, следующие:

- LISTENER_NAME: имя слушателя.
- PROTOCOL_ADDRESS: протокол и адрес прослушивателя. Например, TCP:1521 указывает прослушиватель, который прослушивает подключения на

TCP-порту 1521.

- **SERVICE_NAME**: имя службы, которую обрабатывает слушатель. Это используется клиентами для подключения к экземпляру базы данных.
- **SID_NAME**: системный идентификатор (SID) экземпляра базы данных, который обрабатывает прослушиватель. Это используется клиентами для подключения к экземпляру базы данных.
- **TIMEOUT**: значение времени ожидания для слушателя в секундах. Это указывает количество времени, в течение которого прослушиватель будет ожидать ответа от экземпляра базы данных, прежде чем закрыть соединение.

15. Сетевые настройки Oracle. Установление соединения по сети.

В Oracle сетевые настройки определяют, как база данных взаимодействует с клиентами по сети. Сетевые настройки могут быть настроены с использованием различных параметров, которые указаны в файле конфигурации `sqlnet.ora`.

Oracle Net Services – набор служб, которые устанавливают подключение между сервером БД и пользователями БД

Oracle Net – программный компонент, который инициализирует, устанавливает и поддерживает подключения между клиентом и сервером. Должен быть установлен и на клиенте, и на сервере.

Состоит из 2х компонентов:

- **Oracle Network Foundation layer** – отвечает за установку и поддержание подключений между клиентским приложением и сервером.
- **Oracle Protocol Support** – отвечает за отображение функциональности TNS (Transparent Network Substrate) на стандартные протоколы, используемые при подключении.

Дескриптор подключения - объединенная спецификация двух обязательных компонентов подключения к базе данных: 1) Имени службы базы данных 2)

Местоположения адреса базы данных. **Идентификатор подключения** – именованный дескриптор.

Некоторые из основных сетевых параметров, которые можно настроить в Oracle, следующие:

- NAMES.DEFAULT_DOMAIN : доменное имя по умолчанию, которое используется для разрешения невалифицированных имен хостов.
- NAMES.DIRECTORY_PATH : путь к каталогу, который используется для разрешения глобальных имен баз данных.
- TCP.VALIDNODE_CHECKING : параметр, который определяет, должен ли клиент проверять имя узла сервера перед подключением.
- SQLNET.EXPIRE_TIME: количество секунд, в течение которых соединение может бездействовать до его завершения.
-

Oracle поддерживает 2 режима соединений **выделенный**(dedicated) и **общий**(shared):

1. В режиме выделенного сервера каждое клиентское подключение к базе данных обслуживается выделенным серверным процессом (также известным как "теневой процесс"). Это означает, что каждое клиентское соединение имеет свой собственный серверный процесс, который отвечает за обслуживание этого соединения и выполнение инструкций SQL от имени клиента.

Режим выделенного сервера хорошо подходит для сред с большим количеством клиентских подключений и высоким уровнем использования ЦП и памяти, поскольку он позволяет базе данных масштабироваться и эффективно использовать системные ресурсы.

2. В режиме общего сервера несколько клиентских подключений обслуживаются пулом общих серверных процессов. Когда клиент подключается к базе данных, он назначается общему серверному процессу из пула, который затем отвечает за обслуживание этого соединения и выполнение инструкций SQL от имени клиента.

Режим общего сервера хорошо подходит для сред с меньшим количеством

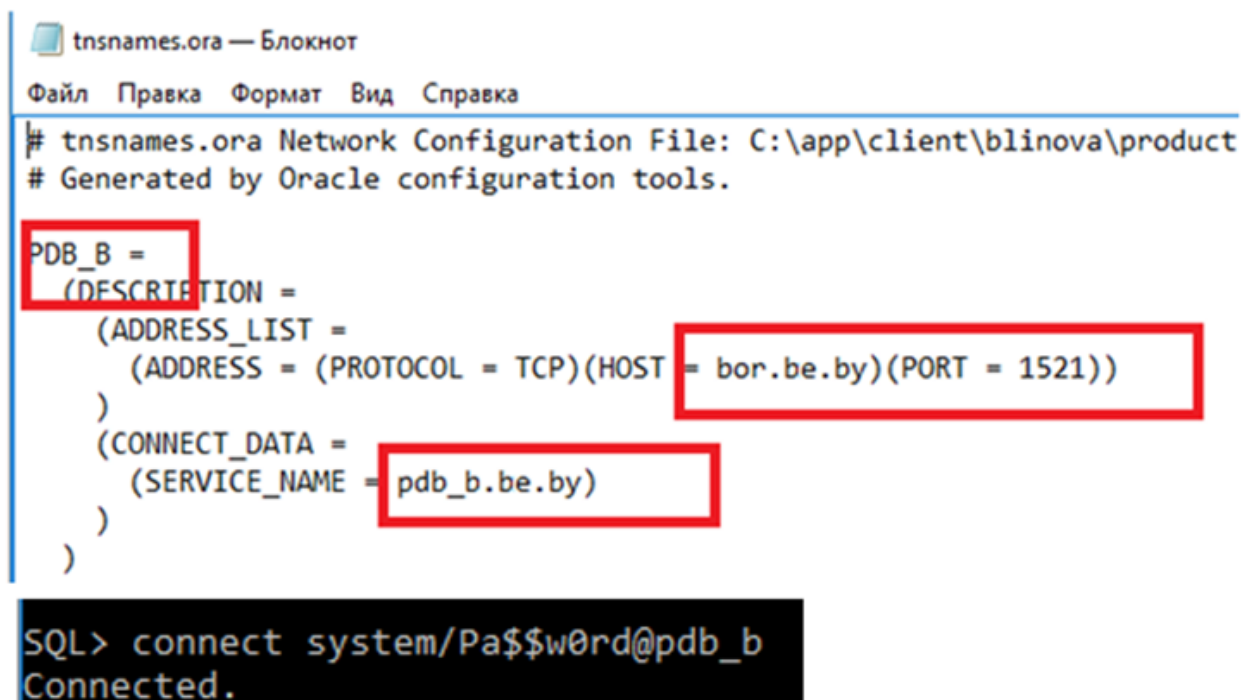
клиентских подключений и более низким уровнем использования процессора и памяти

Также имеются различные **типы** соединений:

- **Basic** : Базовое соединение - это прямое подключение к базе данных с использованием дескриптора подключения. Дескриптор подключения - это набор сведений о соединении, который определяет протокол, используемый для связи, номер порта для подключения и сетевой адрес сервера.

CONNECT имя/пароль@[//]хост[:порт][/имя_службы]

- **TNS** . Это подключение к базе данных с использованием сетевых сервисов Oracle, которые представляют собой набор сетевых инструментов и протоколов, позволяющих клиентам подключаться к базе данных по сети. Чтобы подключиться к базе данных с помощью TNS, клиентское приложение должно иметь доступ к файлу TNSnames, который содержит сведения о подключении к базе данных.



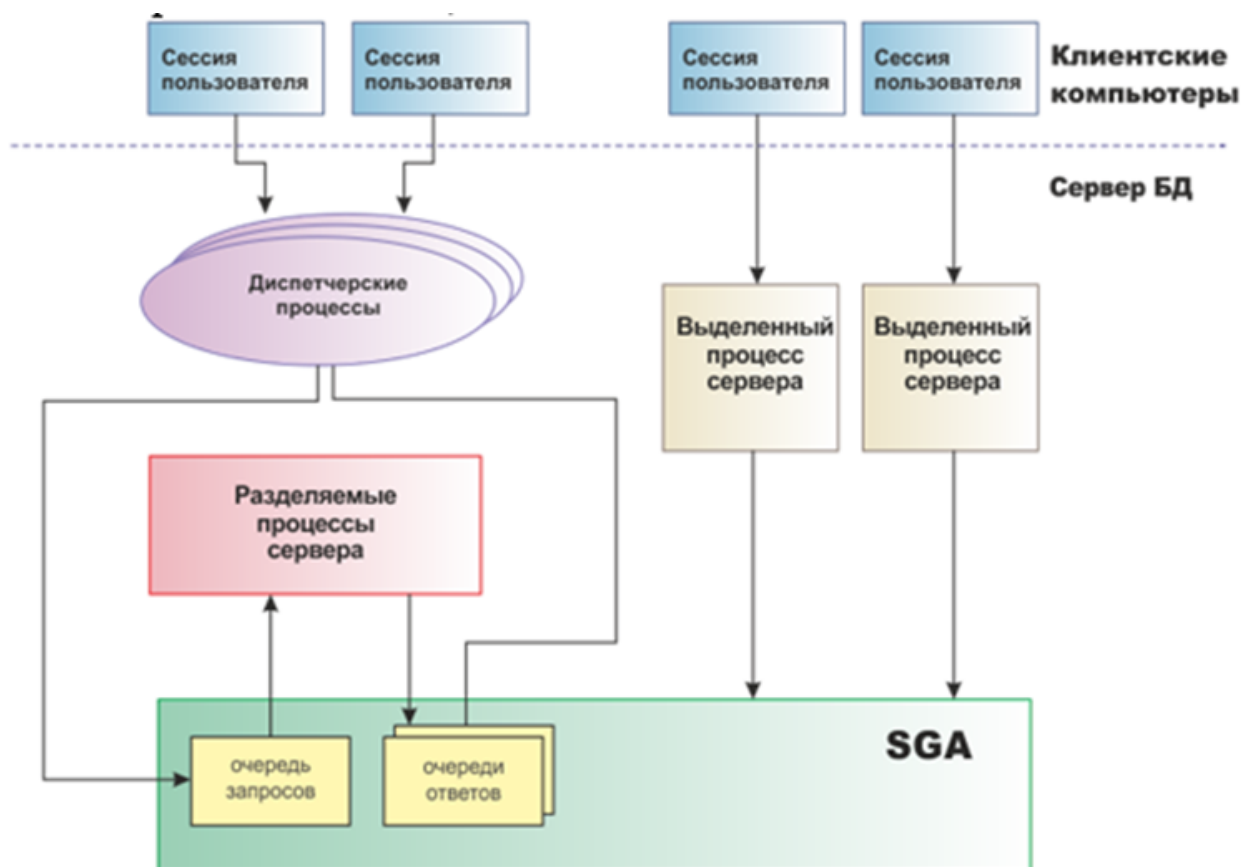
```
tnsnames.ora — Блокнот
Файл  Правка  Формат  Вид  Справка

# tnsnames.ora Network Configuration File: C:\app\client\blinova\product
# Generated by Oracle configuration tools.

PDB_B =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = bor.be.by)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = pdb_b.be.by)
    )
  )

SQL> connect system/Pa$$w0rd@pdb_b
Connected.
```

- **LDAP** - это подключение к базе данных с использованием сервера LDAP (Lightweight Directory Access Protocol) в качестве посредника. Сервер LDAP - это служба каталогов, которая хранит информацию о сетевых ресурсах, включая сведения о подключении к базам данных. Чтобы подключиться к базе данных с помощью LDAP, клиентское приложение должно иметь доступ к серверу LDAP и соответствующие учетные данные.
- **Local/bequeath** - это подключение к базе данных из клиентского приложения, которое выполняется на том же компьютере, что и сервер базы данных. Чтобы подключиться к базе данных с помощью локального/наследуемого соединения, клиентскому приложению не нужно использовать прослушиватель или Oracle Net Services. Вместо этого он может напрямую подключаться к базе данных, используя протокол bequeath, который является проприетарным протоколом для обмена данными между процессами на одном компьютере.



16. Табличные пространства СУБД Oracle и их основные параметры.

В Oracle табличное пространство — это логическая единица хранения, которая используется для управления объектами данных (такими как таблицы и индексы) в базе данных. Табличные пространства могут использоваться для организации данных в базе данных и выделения места для данных на диске.

В Oracle существует несколько типов табличных пространств, включая :

- Permanent — для хранения постоянных объектов БД, может быть несколько, может быть по умолчанию;
- Temporary — для хранения временных объектов БД, может быть несколько;
- Undo — хранение сегментов отката, может быть несколько, но используется всегда одно, которое указано в файле параметров.

Главные параметры которые можно указать при создании табличных пространств :

- TABLESPACE_NAME: это имя создаваемого табличного пространства.
- DATAFILE: это имя файла данных, который будет использоваться для хранения данных для табличного пространства. Вы можете указать несколько файлов данных, разделив их запятыми. Определённый файл может использоваться только для одного табличного пространства.
- SIZE: это начальный размер файла данных в байтах, килобайтах, мегабайтах или гигабайтах. Вы можете указать необязательный параметр MAXSIZE, чтобы указать максимальный размер, до которого может увеличиться файл данных.
- AUTOEXTEND ON/OFF: указывает, должен ли файл данных автоматически увеличивать свой размер, когда он заполняется. Если включено АВТОМАТИЧЕСКОЕ РАСШИРЕНИЕ, файл данных автоматически увеличится в размере на указанное значение ПРИРАЩЕНИЯ, когда он заполнится.
- EXTENT MANAGEMENT LOCAL/Dictionary: это указывает, следует ли управлять распределением экстендов для табличного пространства локально или с использованием словаря данных. Локальное управление экстендами может повысить производительность, но для этого требуется удалить файл данных и создать его заново, если необходимо изменить размер экстента.
- SEGMENT SPACE MANAGEMENT AUTO/MANUAL: Это указывает, должно ли распределение пространства для сегментов в табличном пространстве управляться автоматически или вручную..

- **DEFAULT STORAGE:** задает параметры хранилища по умолчанию для объектов, созданных в табличном пространстве, такие как начальный и следующий размеры экстенда, а также значения `pctfree` и `pctused`.

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

В Oracle **привилегия** — это право выполнять конкретный тип предложений SQL, или право доступа к объекту другого пользователя.

Чтобы предоставить привилегию или роль пользователю или роли в Oracle, вы можете использовать инструкцию `GRANT`. Например:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON my_table TO my_user;
```

Чтобы отозвать привилегию или роль у пользователя или роли в Oracle, вы можете использовать инструкцию `REVOKE`. Например:

```
REVOKE SELECT, INSERT, UPDATE, DELETE ON my_table FROM my_user;
```

В Oracle существует два типа привилегий:

- **Системные привилегии:** эти привилегии позволяют пользователю или роли выполнять определенные действия, влияющие на базу данных в целом, такие как создание табличных пространств или пользователей.

Группы системных привилегий - примеры

Группа	Примеры привилегий
PROCEDURE	CREATE
PROFILE	CREATE ANY
ROLE	ALTER
ROLLBACK SEGMENT	ALTER ANY
SESSION	DROP
SEQUENCE	
SYSTEM	
TABLE	
TABLESPACE	
TRIGGER	
USER	
VIEW	

- Привилегии объекта: эти привилегии позволяют пользователю или роли выполнять определенные действия с определенным объектом базы данных, таким как таблица или представление.

Объектные привилегии - примеры

Привилегия	TABLE	VIEW	SEQUENCE	PROCEDURE
ALTER	+		+	
DELETE	+	+		
EXECUTE				+
INDEX	+			
INSERT	+	+		
REFERENCES	+			
SELECT	+	+	+	
UPDATE	+	+		

В Oracle предложение **WITH ADMIN OPTION** используется для предоставления привилегии предоставлять определенные привилегии другим пользователям или ролям.

Предложение **WITH GRANT OPTION** аналогично предложению WITH ADMIN OPTION, но оно применяется к привилегиям объекта, а не к системным привилегиям.

В Oracle **роль** - это именованная группа привилегий, которые могут быть предоставлены пользователям или другим ролям. Роли могут использоваться для управления привилегиями пользователей в базе данных и упрощения процесса предоставления и отзыва привилегий.

Создание ролей:

```
CREATE ROLE NAME_ROLE;
```

Посмотреть все роли:

```
SELECT * FROM DBA_ROLES;
```

Посмотреть что доступно интересующей роли:

```
SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE = 'NAME_ROLE'
```

18. Пользователь СУБД Oracle и его основные параметры.

В Oracle **пользователь** - это учетная запись базы данных, которая используется для аутентификации и авторизации пользователя или приложения для доступа к базе данных. Пользователям могут быть предоставлены привилегии и роли для управления их доступом к базе данных и действиями, которые они могут выполнять.

Некоторые из основных параметров, которые могут быть настроены для пользователей в Oracle, следующие:

- **USERNAME**: это имя создаваемого пользователя.
- **IDENTIFIED BY**: здесь указывается пароль для пользователя. Пароль может быть указан напрямую или он может быть указан с помощью предложения EXTERNAL IDENTIFIED, которое позволяет хранить пароль во внешней службе аутентификации.
- **DEFAULT TABLESPACE** : задает табличное пространство по умолчанию для пользователя. Все объекты, созданные пользователем, будут храниться в этом табличном пространстве до тех пор, пока не будет указано другое табличное пространство.
- **TEMPORARY TABLESPACE** : указывает временное табличное пространство для пользователя. Временное табличное пространство используется для

хранения временных объектов, таких как буферы сортировки и глобальные временные таблицы.

- **QUOTA**: задает квоту для пользователя в указанном табличном пространстве. Квота ограничивает объем пространства, которое пользователь может использовать в табличном пространстве.
- **PROFILE** : здесь указывается профиль пользователя. Профиль - это набор ограничений ресурсов и политик паролей, которые могут быть назначены пользователю.
- **ROLE** : здесь указываются роли, которые должны быть предоставлены пользователю. Роли - это именованные группы привилегий, которые могут быть предоставлены пользователям.
- **ACCOUNT LOCK | UNLOCK**, (заблокированный или разблокированный пользователь, может использоваться для блокировки пользователя на время или навсегда)
- **PASSWORD EXPIRE**, (пароль пользователя должен быть изменен до того, как пользователь попытается войти в базу данных, то есть при первом входе, будет предложено сменить текущий пароль, старый пароль только для первого входа).

19. Профиль безопасности СУБД Oracle и его основные параметры.

В Oracle **профиль безопасности** - это набор параметров безопасности, которые могут быть назначены пользователю для управления его доступом к базе данных и действиями, которые он может выполнять. Профили безопасности можно использовать для обеспечения соблюдения политик безопасности и упрощения процесса управления привилегиями пользователей и ограничениями ресурсов.

Некоторые из основных параметров, которые можно настроить для профилей безопасности в Oracle, следующие:

- **PASSWORD_LIFE_TIME**: количество дней, в течение которых пароль пользователя действителен до истечения срока его действия.
- **FAILED_LOGIN_ATTEMPTS**: The number of failed login attempts that are allowed before the user's account is locked.

- **PASSWORD_REUSE_TIME**: количество дней, которые пользователь должен подождать, прежде чем повторно использовать пароль.
- **PASSWORD_LOCK_TIME**: количество дней, в течение которых учетная запись пользователя будет заблокирована после достижения максимального количества неудачных попыток входа в систему.
- **PASSWORD_GRACE_TIME**: количество дней, в течение которых пользователь должен сменить свой пароль после истечения срока его действия.
- **IDLE_TIME**: максимальное количество секунд, в течение которых сеанс пользователя может находиться в режиме ожидания до его завершения.
- **CONNECT_TIME**: максимальное количество секунд, в течение которых сеанс пользователя может быть подключен до его завершения.
- **SESSIONS_PER_USER** : параметр для ограничения количества одновременных сеансов, которые разрешено иметь пользователю.

Создание профиля безопасности

```
CREATE PROFILE PFEACORE LIMIT
```

```

PASSWORD_LIFE_TIME 180 -- количество дней жизни пароля
SESSIONS_PER_USER 3 -- количество сессий для пользователя
FAILED_LOGIN_ATTEMPTS 7 -- количество попыток входа
PASSWORD_LOCK_TIME 1 -- количество дней блокирования после ошибок
PASSWORD_REUSE_TIME 10 -- через сколько дней можно повторить пароль
PASSWORD_GRACE_TIME DEFAULT -- количество дней предупреждений о смене пароля
CONNECT_TIME 180 -- время соединения, минут
IDLE_TIME 30 -- количество минут простоя

```

Профиль с именем **DEFAULT**: профиль с неограниченными ресурсами. По умолчанию назначается всем пользователям и не имеет ограничений. Если пользователю не назначен профиль безопасности, либо в назначенном профиле отсутствуют некоторые параметры, то они берутся из профиля по умолчанию:

```

SELECT *
FROM DBA_PROFILES
WHERE PROFILE = 'DEFAULT';

```

	PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT	COMMON
1	DEFAULT	COMPOSITE_LIMIT	KERNEL	UNLIMITED	NO
2	DEFAULT	SESSIONS_PER_USER	KERNEL	UNLIMITED	NO
3	DEFAULT	CPU_PER_SESSION	KERNEL	UNLIMITED	NO
4	DEFAULT	CPU_PER_CALL	KERNEL	UNLIMITED	NO
5	DEFAULT	LOGICAL_READS_PER_SESSION	KERNEL	UNLIMITED	NO
6	DEFAULT	LOGICAL_READS_PER_CALL	KERNEL	UNLIMITED	NO
7	DEFAULT	IDLE_TIME	KERNEL	UNLIMITED	NO
8	DEFAULT	CONNECT_TIME	KERNEL	UNLIMITED	NO
9	DEFAULT	PRIVATE_SGA	KERNEL	UNLIMITED	NO
10	DEFAULT	FAILED_LOGIN_ATTEMPTS	PASSWORD	10	NO
11	DEFAULT	PASSWORD_LIFE_TIME	PASSWORD	180	NO
12	DEFAULT	PASSWORD_REUSE_TIME	PASSWORD	UNLIMITED	NO
13	DEFAULT	PASSWORD_REUSE_MAX	PASSWORD	UNLIMITED	NO

20. Язык SQL. Основные операторы и их назначение.

SQL (Structured Query Language) - это язык программирования, который используется для управления данными в системах управления реляционными базами данных (СУБД), таких как Oracle, MySQL и Microsoft SQL Server.

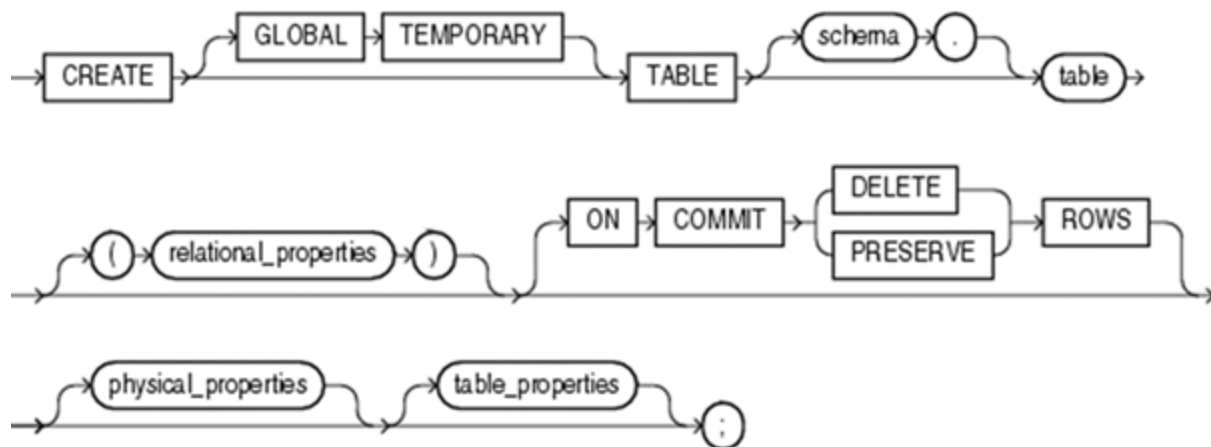
Существует 4 группы операторов sql :

1. **Data definition language** (DDL) operators: Эти операторы используются для определения структуры и организации базы данных. CREATE, ALTER , ,DROP.
2. **Data manipulation language** (DML) operators: Эти операторы используются для манипулирования и изменения данных в базе данных.SELECT, INSERT, UPDATE, DELETE.
3. **Data control language** (DCL) operators: Эти операторы используются для управления доступом к базе данных и для управления привилегиями. GRANT, REVOKE
4. **Операторы языка управления транзакциями** (TCL): Эти операторы используются для управления транзакциями в базе данных, которые

представляют собой группы инструкций SQL, которые рассматриваются как единое целое. COMMIT, ROLLBACK, SAVEPOINT

21. Таблица и ее основные параметры.

Таблицы — основная структура хранения информации в БД. Таблица — это логическая сущность, которая делает чтение и манипуляции данными интуитивно понятными для пользователя. Таблица состоит из столбцов и строк, причем строка соответствует одиночной записи, которая состоит из набора полей. Когда вы создаете таблицу, то присваиваете ей имя и определяете набор столбцов, относящихся к ней. Каждый столбец имеет имя и определенный тип данных. Для определенных столбцов можно специфицировать ширину или точность и масштаб (scale), а некоторым из них могут быть установлены значения по умолчанию.



Типы таблиц:

- Традиционные таблицы (heap organized table)
- Индекс-таблицы (index organized table)
- Кластеризованные индекс-таблицы (index clustered table)
- Кластеризованные хэш-таблицы (hash clustered table)
- Отсортированные кластеризованные хэш-таблицы (sorted hash clustered table)
- Вложенные таблицы (nested table)
- Временные таблицы (temporary table)
- Объектные таблицы

- Внешние таблицы

Может иметь до 1000 столбцов (<254)

Может иметь неограниченное число строк

Может иметь неограниченное число индексов

Нет ограничения на число таблиц

Create table Employee(

emp_id char(12),

emp_name nvarchar2(100),

hire_date date,

constraint pk_emp_id primary_key (emp_id)

)

organization index — определяет организацию в таблице, может повысить производительность

no monitoring — отключает мониторинг таблицы, не собирает статистику

logging — добавление dml операций в журнал

pctthreshold 20 — после выполнения операций должно остаться минимум 20% свободного места, предотвращает фрагментацию, повышает производительность

Параметры

Параметры PCTFREE и PCTUSED

Параметр PCTFREE – процент памяти блока, резервируемой для возможных обновлений строк, уже содержащихся в блоке

Параметр PCTUSED – процент занятой части памяти блока

Управление параметрами заполнения блока

Automatic segment space management - только PCTFREE

Manual segment space management - PCTUSED, PCTTHRESHOLD, FREELISTS и др.

22. Временные таблицы.

Временные таблицы – механизм хранения данных в БД. Состоит из столбцов и строк, как и обычная таблица. Временные таблицы – глобальны.

Привилегии для создания временной таблицы CREATE TABLE

Можно разместить временную таблицу в заданном табличном пространстве.

Временные таблицы – это шаблон, хранящийся в словаре базы данных, для нее выделяется временный сегмент в (по умолчанию) TEMPORARY-табличном пространстве и для каждого пользователя свой.

Каждый пользователь видит только свои данные (свой сегмент данных).

Статичны: временные таблицы создаются (CREATE) один раз и существуют, пока их не удалят (DROP). DROP не получится, если таблица в этот момент используется другим пользователем.

Временные таблицы бывают:

- ON COMMIT PRESERVE ROWS – на время сеанса, данные существуют только на время сеанса, возможны все DML-операторы, TCL-операторы
- ON COMMIT DELETE ROWS – на время транзакции, данные существуют только на время транзакции, возможны все DML-операторы, после выполнения COMMIT или ROLLBACK таблица становится пустой

В начале сеанса временная таблица всегда пуста!

Для временных таблиц можно создавать триггеры, указать констрейны (ограничения), создавать индексы

Не могут быть индексно-организованными, нельзя секционировать, размещать в кластере!

Данные повторного выполнения генерируются, но их количество пренебрежительно мало.

Отличие от обычных: нет flashback (в доке есть, но я не уверен в правдивости этого факта. В лекциях ничего нет. Есть инфо про приватные временные таблицы, но 18 оракла и выше, и там действительно дропается все сразу мимо корзины)

23. Ограничения целостности в таблицах.

Ограничения целостности в реляционных базах данных позволяют легко и автоматически применять важные бизнес-правила к таблицам базы данных.

datatype тип данных Предотвращает появление в столбце значений, не соответствующих типу данных

notnull запрет значений null Предотвращает появление в столбце значений null

default значение по умолчанию Устанавливает значение в столбце по умолчанию при выполнении операции INSERT

primary key первичный ключ Предотвращает появление в столбце (группе столбцов) повторяющихся значений (комбинации значений) и пустого значения (комбинации пустых значений)

foreign key внешний ключ. Устанавливает связь между таблицей со столбцом, имеющим свойство foreignkey (FK) и таблицей, имеющей столбец со свойством primarykey (PK); предотвращает несогласованные операции между PK и FK

unique уникальное значение Аналогично primarykey, но допускает пустые значения и не может быть использован для связи с foreignkey

check проверка значений Предотвращается появление в столбце значения, не удовлетворяющего логическому условию

Для ограничений целостности PRIMARYKEY, FOREIGNKEY, UNIQUE и CHECK может быть задано имя, которое при возникновении ошибки, связанной с этим ограничением, будет указано в сообщении сервера. В том случае, если это имя не задано, при создании таблицы сервер назначает ограничениям этих типов собственные имена.

пример

```
CREATE TABLE employees (  
  employee_id NUMBER(10) PRIMARY KEY,  
  first_name VARCHAR2(50) NOT NULL,  
  email VARCHAR2(255) UNIQUE ,  
  hire_date DATE DEFAULT SYSDATE,  
  department_id NUMBER(10) NOT NULL,  
  FOREIGN KEY (department_id) REFERENCES departments(department_id)  
);
```

24. Типы данных базы данных.

Тип данных связан с конкретным форматом хранения и ограничениями диапазона. В Oracle каждому значению или константе присваивается тип данных.

Типы данных ORACLE – символьные

CHAR	Символьное поле фиксированной длины до 2000 байт
NCHAR	Поле фиксированной длины для набора символов, состоящих из нескольких байт. Максимальный размер – 2000 символов или 2000 байт в зависимости от набора символов.
VARCHAR2	Символьное поле переменной длины до 4000 байт
NVARCHAR2	Поле переменной длины для набора символов, состоящих из нескольких байт. Максимальный размер – 4000 символов или 4000 байт в зависимости от набора символов.
LONG	Символьный, переменной длины, до 2GB, оставлен для совместимости
RAW(n)	Переменной длины, для бинарных данных n <= 2000 byte оставлен для совместимости
LONG RAW	Бинарные данные до 2GB
CLOB	Символьный тип большой объект до 4GB
NLOB	CLOB для многобайтных символов
BLOB	Большой двоичный объект до 4GB
BFILE	Указатель на двоичный файл операционной системы

Типы данных ORACLE – дата/время

DATE	7 байтовое поле фиксированной длины, используемое для хранения даты и времени
INTERVAL DAY TO SECOND	11 байтовое поле фиксированной длины для интервала времени: Дни, часы, минуты, секунды
INTERVAL YEAR TO MONTH TIMESTAMP	5 байтовое поле фиксированной длины для интервала времени: Годы и месяцы
TIMESTAMP WITH TIME ZONE	13 байтовое поле фиксированной длины Дата, время и настройки, связанные с часовым поясом.
TIMESTAMP WITH LOCAL TIME	7-11 байтовое поле переменной длины Дата и время, приведенные к часовому поясу базы данных

Типы данных ORACLE – числовые

--	--

NUMBER(n, s)	Числовой тип переменной длины Точность n <= 38, общее количество цифр Масштаб s = [-84,127], количество цифр после запятой
--------------	--

Неявные преобразования типов данных

VARCHAR2 CHAR	DATE
DATE	VARCHAR2
VARCHAR2 CHAR	ROWID
ROWID	VARCHAR2
VARCHAR2 CHAR	NUMBER
NUMBER	VARCHAR2

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

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

Виды индексов Oracle Database

Индексы Oracle могут относиться к нескольким видам, наиболее важные из которых перечислены ниже.

- **Уникальные и неуникальные индексы.** Уникальные индексы основаны на уникальном столбце. Уникальные индексы можно создавать явно, но 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.** Составные индексы — это 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.

Создание индекса

Индекс создается с помощью оператора *CREATE INDEX*:

```
CREATE INDEX employee_id ON employee(employee_id) TABLESPACE  
emp_index_01;
```

Типы индексов:

- Табличный (B*Tree) индекс
- Битовый индекс
- Функциональный индекс

Плотность запроса – количество возвращаемых строк запроса

Селективность таблицы — значение, представляющее долю строк таблицы, удовлетворяющих определённому условию выбора

1. **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

2. **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) индексированное значение

Предназначен для индексирования столбцов с низкой селективностью

Не подходит для таблиц с частым обновлением

Хорошо подходят для хранилищ данных

3. **Функциональный индекс** – предварительно вычисляют значения функции по заданному столбцу и сохраняют результат в индексе

LOWER(NAME) / UPPER (NAME)

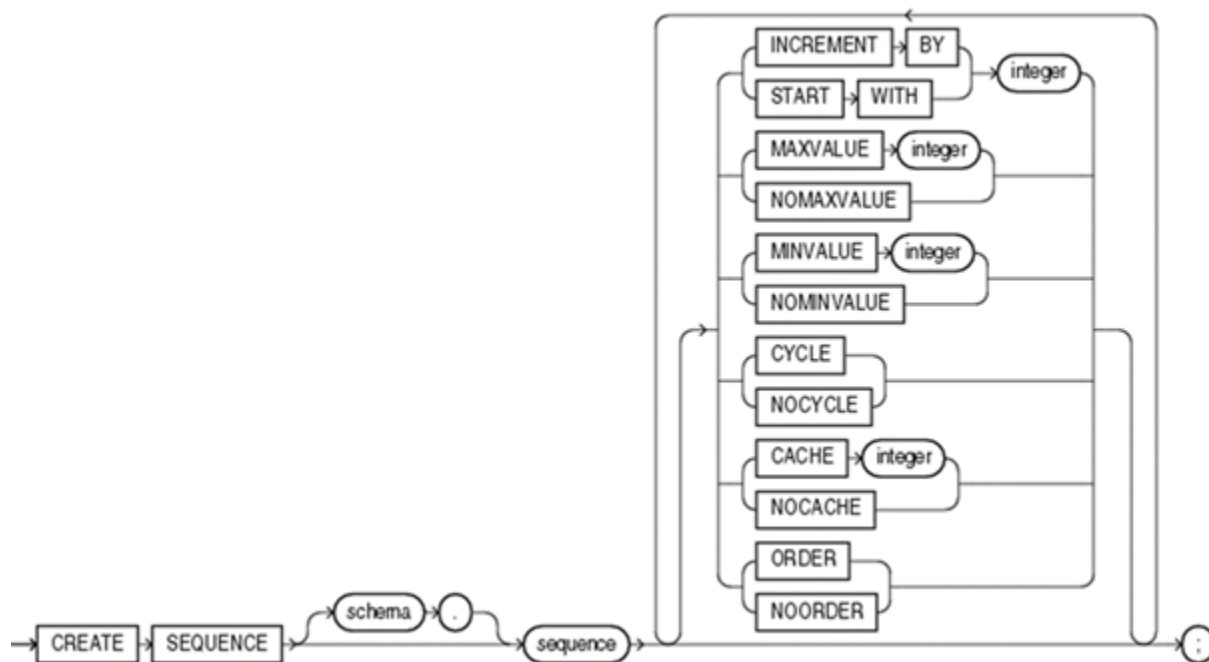
Руководство по созданию индексов:

- Индексация имеет смысл, если нужно обеспечить доступ одновременно не более чем к 4–5% данных таблицы. Альтернативой использованию индекса для доступа к данным строки является полное последовательное чтение таблицы от начала до конца, что называется полным сканированием таблицы. Полное сканирование таблицы больше подходит для запросов, которые требуют извлечения большего процента данных таблицы. Помните, что применение индексов для извлечения строк требует двух операций чтения: индекса и затем таблицы.
- Избегайте создания индексов для сравнительно небольших таблиц. Для таких таблиц больше подходит полное сканирование. В случае маленьких таблиц нет необходимости в хранении данных и таблиц, и индексов.
- Создавайте первичные ключи для всех таблиц. При назначении столбца в качестве первичного ключа Oracle автоматически создает индекс по этому столбцу.
- Индексируйте столбцы, участвующие в многотабличных операциях соединения.
- Индексируйте столбцы, которые часто используются в конструкциях *WHERE*.
- Индексируйте столбцы, участвующие в операциях *ORDER BY* и *GROUP BY* или других операциях, таких как *UNION* и *DISTINCT*, включающих сортировку. Поскольку индексы уже отсортированы, объем работы по выполнению необходимой сортировки данных для упомянутых операций будет существенно сокращен.
- Столбцы, состоящие из длинносимвольных строк, обычно плохие кандидаты на индексацию.
- Столбцы, которые часто обновляются, в идеале не должны быть индексированы из-за связанных с этим накладных расходов.
- Индексируйте таблицы только с высокой селективностью. То есть индексируйте таблицы, в которых мало строк имеют одинаковые значения.
- Сохраняйте количество индексов небольшим.

- Составные индексы могут понадобиться там, где одностолбцовые значения сами по себе не уникальны. В составных индексах первым столбцом ключа должен быть столбец с максимальной селективностью.

26. Последовательность СУБД Oracle и ее параметры.

Последовательность – объект базы данных, предназначенный для генерации числовой последовательности. **Oracle** использует **генератор последовательностей** для **автоматической генерации уникальной последовательности чисел**, которые пользователь может применять в своих операциях. Обычно последовательности используются для создания уникальных чисел для столбца первичного ключа.



При создании последовательности, что означают опции cache и nocache?

Что касается последовательности, опция cache определяет, сколько значений последовательности будут сохранены в памяти для быстрого доступа.

Недостатком создания последовательности с cache, что если происходит отказ системы, все кэшированные значения последовательности, которые не были использованы, будут утеряны. Это приведет к разрывам в значениях, назначенной последовательности. Когда в система восстановится, Oracle будет кэшировать

новые номера, с того места, где была прервана последовательность, игнорируя утерянные значения последовательности.

Как установить значение lastvalue в последовательность Oracle?

Например, если последнее значение используемой последовательности Oracle был 100, и вы хотите, чтобы следующее значение было 225. Вы должны выполнить следующие команды.

```
ALTER SEQUENCE seq_name  
INCREMENT BY 124;  
SELECT seq_name.nextval FROM dual;  
ALTER SEQUENCE seq_name  
INCREMENT BY 1;
```

Следующее значение последовательности, для использования, теперь будет 225.

Привилегия

create sequence

increment by 10

start with 100

nomaxvalue

nominvalue

nocycle

cache 20

order --если клиенту подкл в определенном порядке, сервер следит

Представления словаря: sys.dba_sequences, sys.all_sequences,
sys.user_sequences

27. Кластер и его параметры

Кластер – объект БД, который хранит значения общих столбцов нескольких таблиц

Создание CREATE CLUSTER

Привилегия CREATE CLUSTER

Таблицы, с которыми часто работают совместно, можно физически хранить совместно.

⇒ Для этого создается кластер, который будет их содержать

⇒ Строки из отдельных таблиц сохраняются в одних и тех же блоках, поэтому объединяющие запросы выполняются быстрее

Уменьшается количество операций ввода-вывода

Производительность операций вставки, обновления и удаления может быть ниже, чем для обычных таблиц

Связанные столбцы называются кластерным ключом

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

Наибольшие преимущества – в запросах, использующих операции равенства:

```
select Name from STUDENT where Id = 999;
```

Пример: `create cluster CJACORE.cl1 (x number(2,5) hash case 100)...`

Параметры:

1. HASHKEYS: количество хеш-ключей, которые кластер использует для хранения данных. Хэш-ключи определяют распределение данных между блоками данных в кластере.
2. SIZE: размер блоков данных в кластере в байтах. Это определяет количество данных, которые могут быть сохранены в каждом блоке данных.
3. HASH IS: функция, которую кластер использует для генерации хеш-ключей. Функция должна принимать значения столбцов в ключе кластера в качестве входных данных и возвращать хэш-ключ в качестве выходных данных.
4. PCTFREE: процент свободного места, который должен оставаться в блоках данных кластера. Это определяет объем пространства, доступного для вставки или обновления строк в кластере.
5. PCTUSED: Процент используемого пространства, при котором кластер должен освободить свободное пространство. Когда используемое пространство в блоках данных достигает этого порога, кластер уплотняет данные, чтобы освободить свободное пространство.

28. Представление и его параметры.

Представление – хранимый запрос

- Можно обращаться, как к обычной таблице
- Данные хранятся в таблице
- Добавляют уровень защиты данных
- Скрывают сложность данных
- Скрывают имена столбцов таблиц

Привилегия – CREATE VIEW

Создание – CREATE (OR REPLACE) VIEW

Параметры:

- FORCE – создает представление, независимо от того, существуют ли таблицы и есть ли права
- NOFORCE – по умолчанию
- WITH CHECK OPTION – запрещает вставку в представления данных, которые не будут отображаться в этом представлении
- WITH READ ONLY
- WITH NO DATA

29. Материализованное представление и его параметры.

Материализованное представление — физический объект базы данных, содержащий результат выполнения запроса

Привилегия – CREATE MATERIALIZED VIEW

Создание – CREATE MATERIALIZED VIEW

BUILD IMMEDIATE – создает представление в момент выполнения оператора

START WITH – показывает, когда выполнится в первый раз (если не был построен сразу)

NEXT – показывает, когда выполнится в следующий раз

Далее – в разницу времени между START WITH и NEXT

(start with – next = периодичность) **COMMIT !!!!** **Отличие:** Хранят не только запрос, но и какие-то данные, которые обновляются на некотором временном промежутке или их можно обновлять вручную

REFRESH COMPLETE — полное обновление данных из базовых таблиц

REFRESH FAST – используются журналы фиксации изменений базовых таблиц

```
CREATE MATERIALIZED VIEW LOG ON EMPLOYEE  
WITH ROWID, PRIMARY KEY  
INCLUDING NEW VALUES;
```

```
CREATE MATERIALIZED VIEW EMP_SALARY_LIST  
REFRESH FAST ON COMMIT  
ENABLE QUERY REWRITE  
AS SELECT EMP_ID, SALARY FROM EMPLOYEE;
```

REFRESH FORCE – попытка быстрого обновления; если быстрое обновление невозможно, то выполняется полное обновление

Обновление явное

Обновление неявное по расписанию

ENABLE QUERY REWRITE — возможность перезаписывать план запроса на получения данных из представления, а не из таблицы, повышает производительность, усложняет структуру

REFRESH:

- ON COMMIT – обновление при COMMIT
- ON DEMAND – обновление по требованию

Просмотр USER_MVIEWS, USER_MVIEW_LOGS

Удалить DROP MATERIALIZED VIEW

30. Частные и публичные синонимы СУБД Oracle.

Синоним – способ обращаться к объекту базы данных без указания обязательной полной идентификации объекта (хост – экземпляр – владелец – объект).

Частный синоним принадлежит пользователю, который его создал.

Публичный синоним используется совместно всеми пользователями базы данных.

Привилегия – CREATE (PUBLIC) SYNONYM

Создание – CREATE (PUBLIC) SYNONYM

Допустимость синонима не проверяется сервером при создании!

Представление словаря dba.synonyms

Может указывать на: таблицы, процедуры, функции, последовательности, представления, пакеты, объекты в локальной или удаленной базе данных

пример: *create synonym T1 for svvcore.teacher;*

31. Основные характеристики языка PL/SQL.

PL/SQL - Procedure Language extensions to SQL

Основной язык для программирования хранимых процедур (stored procedures);

Интегрирован с базой данных Oracle;

Производительность серверных модулей;

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

Отсутствие накладных расходов на приведение типов;

Может выполняться независимо от пользователя;

PL/SQL-функции можно вызывать из SELECT запросов

Взаимодействие с пользователем (user interaction);

Внутренний язык (proprietary for Oracle);

Содержит элементы объектно-ориентированного программирования;

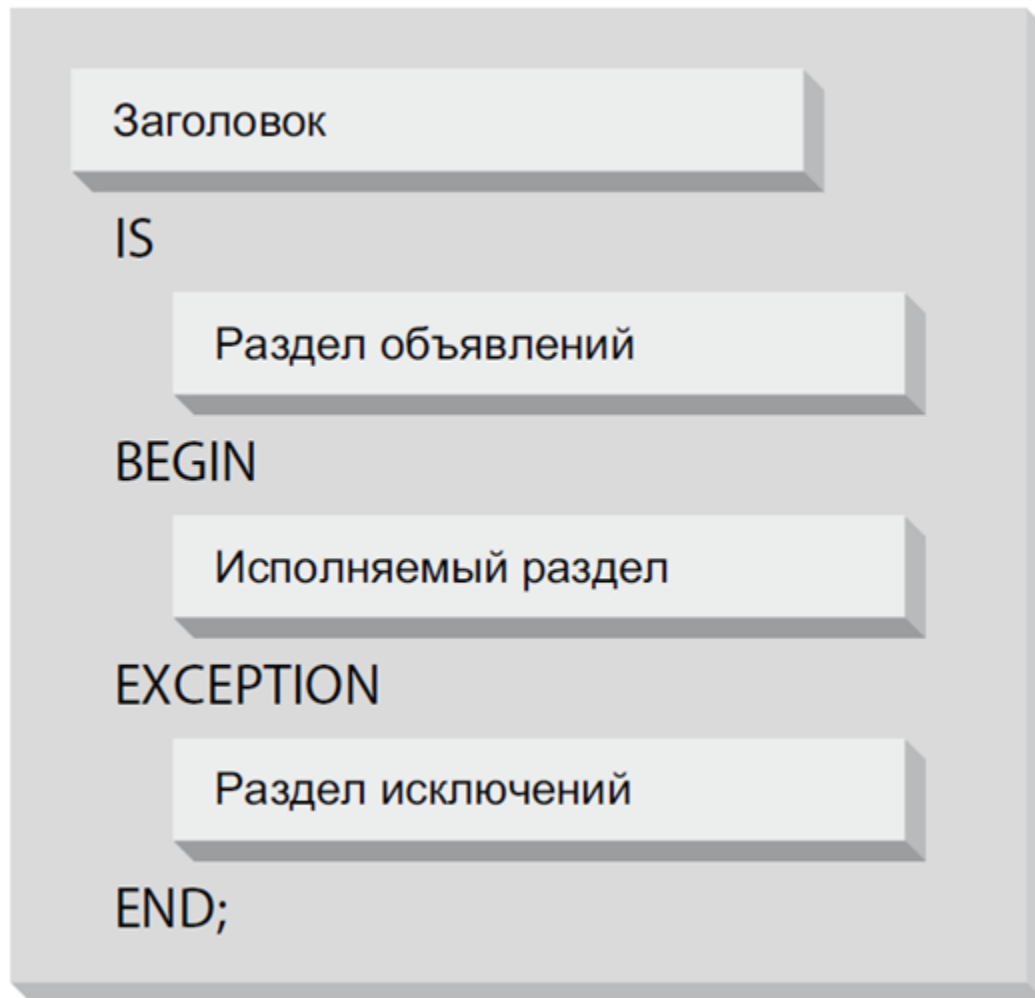
Позволяет использовать объектные типы;

Интерпретация (режим по умолчанию);

Компиляция (промежуточный код на C и конечный объектный код процессора);

Среда выполнения: SQL*Plus, SQL Developer, TOAD.

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



Нет имени (заголовка) – анонимный блок

DECLARE

Раздел объявлений – не обязательно – объявление переменных

BEGIN

Исполняемый раздел – обязательно, хоть один

EXCEPTION

Раздел исключений – не обязательно

END;

Анонимный блок – не может быть вызван из другого блока

- Используется как скрипт для выполнения PL/SQL выражений
- Начинается с DECLARE или BEGIN
- Простейший состоит из команды *null* (не делает ничего, используется когда специально хотим показать, что гасим всякую деятельность)
- не чувствителен к регистру (лучше маленьк, большими только sql-операторы)

Именованные блоки – процедуры и функции

when others – обработка любого исключения

sqlerrm – функция, которая возвращает сообщение об ошибке

sqlcode – функция, которая возвращает № ошибки (обычно 5-значный)

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

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

Символьные типы

CHAR	Символьное поле фиксированной длины до 2000 байт
NCHAR	Поле фиксированной длины для набора символов, состоящих из нескольких байт. Максимальный размер – 2000 символов или 2000 байт в зависимости от набора символов.
VARCHAR2	Символьное поле переменной длины до 4000 байт
NVARCHAR2	Поле переменной длины для набора символов, состоящих из нескольких байт. Максимальный размер – 4000 символов или 4000 байт в зависимости от набора символов.
LONG	Символьный, переменной длины, до 2GB, оставлен для совместимости
RAW(n)	Переменной длины, для бинарных данных n <= 2000 byte, оставлен для совместимости
LONG RAW	Бинарные данные до 2GB
CLOB	Символьный тип большой объект до 4GB
NLOB	CLOB для многобайтных символов

BLOB	Большой двоичный объект до 4GB
BFILE	Указатель на двоичный файл операционной системы

Дата / время

DATE	7 байтовое поле фиксированной длины, используемое для хранения даты и времени
INTERVAL DAY TO SECOND	11 байтовое поле фиксированной длины для интервала времени: Дни, часы, минуты, секунды
INTERVAL YEAR TO MONTH TIMESTAMP	5 байтовое поле фиксированной длины для интервала времени: Годы и месяцы
TIMESTAMP WITH TIME ZONE	13 байтовое поле фиксированной длины Дата, время и настройки, связанные с часовым поясом.
TIMESTAMP WITH LOCAL TIME	7-11 байтовое поле переменной длины Дата и время, приведенные к часовому поясу базы данных

Числовые

NUMBER(n,s)	Числовой тип переменной длины Точность $n \leq 38$, общее количество цифр Масштаб $s = [-84, 127]$, количество цифр после запятой
-------------	---

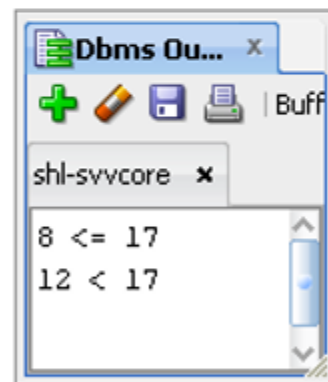
Основные операции:

- Оператор IF

```

declare
    x pls_integer := 17;
begin
    if 8 > x
    then
        dbms_output.put_line('8 > '||x);
    end if;
-----
    if 8 > x
    then
        dbms_output.put_line('8 > '||x);
    else
        dbms_output.put_line('8 <= '||x);
    end if;
-----
    if 8 > x
    then
        dbms_output.put_line('8 > '||x);
    elsif 8 = x
    then
        dbms_output.put_line('8 = '||x);
    elsif 12 > x
    then
        dbms_output.put_line('12 > '||x);
    elsif 12 = x
    then
        dbms_output.put_line('12 = '||x);
    else
        dbms_output.put_line('12 < '||x);
    end if;
end;

```

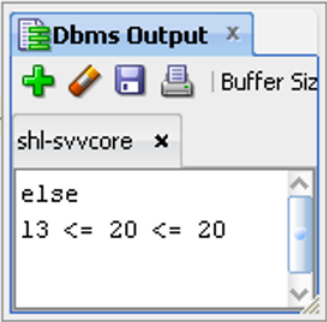


- Оператор CASE


```

declare
    x pls_integer := 17;
begin
    -----
    case x
        when 1 then dbms_output.put_line('1');
        when 2 then dbms_output.put_line('2');
        when 3 then dbms_output.put_line('3');
        else dbms_output.put_line('else');
    end case;
    -----
    case
        when 8 > x then dbms_output.put_line('8 > '||x);
        when 8 = x then dbms_output.put_line('8 = '||x);
        when 12 = x then dbms_output.put_line('12 = '||x);
        when x between 13 and 20 then dbms_output.put_line('13 <= '||x||' <= 20');
        else dbms_output.put_line('else');
    end case;
    -----
end;

```



- Циклы loop, for, while

```

declare
    x pls_integer := 0;
begin
    -----

    loop
        x := x + 1;
        dbms_output.put_line(x);
    exit when x > 5;
    end loop;

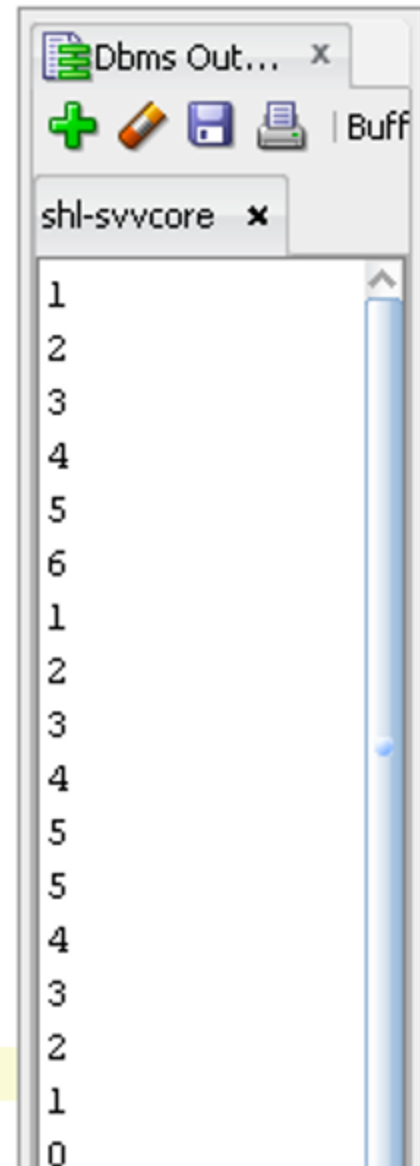
    -----

    for k in 1..5
    loop
        dbms_output.put_line(k);
    end loop;

    -----

    while (x > 0)
    loop
        x := x - 1;
        dbms_output.put_line(x);
    end loop;
    -----
end;

```



Константы: с помощью constant

Пример: n1 constant number (5) := 5;

При попытке изменить значение будет выдана ошибка.

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

NLS – National Language Support

- может 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 быть 2 набора символов: основной (database character set) и дополнительный (national character set)
- устанавливаются при создании БД
- изменяются: alter database (national) character set
- кроме символов алфавита в набор входит знаки препинания, числа, символы денежных единиц и пр.

основной набор символов для:

- хранения символьных типов char, varchar2, clob, long
- описание имен объектов, переменных
- ввод и хранения pl/sql модулей

дополнительный набор символов для:

- хранения символьных типов nchar, nvarchar2, nlob

переменная окружения **NLS_LANG = lang_territory.charset**

lang – имена месяцев, дней, направление текста : по умолч. american

territory – настройка календаря, формат даты, денег

charset – отображение символов, заглавных букв,

словари: nls_ [session | instance | database] _parameters

Семантика символов:

- байтовая семантика – рассматривает строки как последовательность байтов (По умолчанию - BYTE)
- символьная семантика – рассматривает строки как последовательность символов
- задается nls_length_semantics

- Можно задавать семантику для столбца: VARCHAR2(20 BYTE) VARCHAR2(10 CHAR)

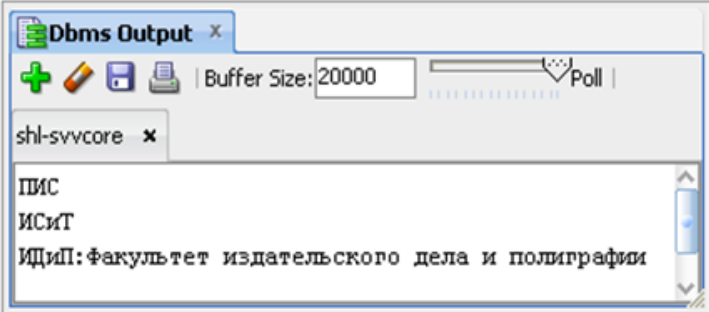
35. Связанные объявления переменных: инструкция %TYPE, инструкция %ROWTYPE.

Тип переменной основан на известной структуре 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 курсор

```
-- ll/ll.sql
declare
    subject      svvcore.subject.subject%type;
    pulpit       svvcore.pulpit.pulpit%type;
    faculty_rec  svvcore.faculty%rowtype;
begin
    subject := 'ПИС';
    pulpit := 'ИСИТ';
    faculty_rec.faculty := 'ИДИП';
    faculty_rec.faculty_name := 'Факультет издательского дела и полиграфии';
    dbms_output.put_line(subject);
    dbms_output.put_line(pulpit);
    dbms_output.put_line(rtrim(faculty_rec.faculty) || ':' || faculty_rec.faculty_name;);
exception
    when others
    then dbms_output.put_line('error = ' || sqlerrm);
end;
```



36. Локальные процедуры и функции языка PL/SQL.

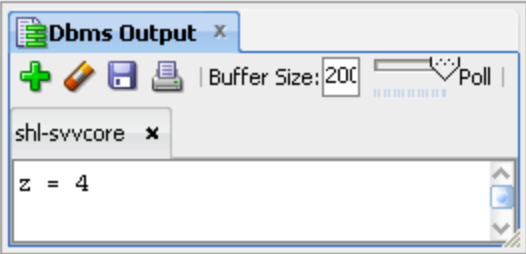
Локальный программный модуль – это процедура или функция, определенная в секции декларации PL/SQL блока

- Объявление локальных процедур и функций должно размещаться в конце секции декларации после всех типов, записей, курсоров, переменных и исключений
- могут быть использованы только в рамках блока, в котором они объявлены
- могут быть перегружены

Локальная процедура:

```
-- 13/08.sql
declare
  x number(3) := 4;
  y number(3) := 5;
  z number(3);
  procedure summod5 (x1 number, x2 number, x3 out number)
  is
    z number(3) := 5;
    begin
      x3 := mod(x1+ x2,z);
    end summod5;
  begin
    summod5(x,y,z);
    dbms_output.put_line('z = '||z);

  exception
    when others then dbms_output.put_line(sqlerrm);
end;
/
```



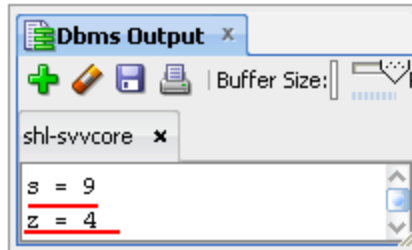
Локальная функция:

```

-- 13/09.sql
declare
  x number(3) := 4;
  y number(3) := 5;
  z number(3);
  s number(5);
  function summod5 (x1 number, x2 number, x3 out number)
    return number is
      z number(3) := 5;
    begin
      x3 := mod(x1+ x2,z);
      return (x1+x2);
    end summod5;
begin
  s := summod5(x,y,z);
  dbms_output.put_line('s = '||s);
  dbms_output.put_line('z = '||z);

exception
  when others then dbms_output.put_line(sqlerrm);
end;
/

```



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

Запись – 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

Объявление записей:

- на осн.таблицы (%rowtype)

declare

one_book books%rowtype;

- на основе курсора (cursor + %rowtype)

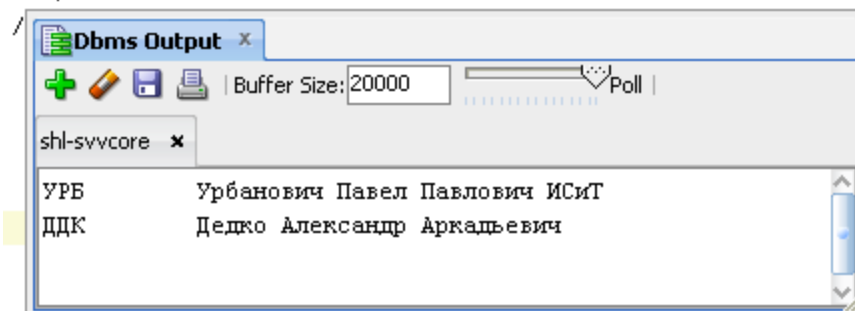
declatе cursor *myc* is

```
select * from books where author like '%Fadeev%';
```

```
one_curs mycc%rowtype;
```

- запись определяемая программистом:

```
-- 10/03.sql
declare
  rec1 teacher%rowtype;
  type person is record
  (
    code teacher.teacher%type,
    name teacher.teacher_name%type
  );
  rec2 person;
begin
  select * into rec1 from teacher where teacher = 'УРЕ';
  select teacher, teacher_name into rec2 from teacher where teacher = 'ДДК';
  dbms_output.put_line(rec1.teacher||' '||rec1.teacher_name||' '||rec1.pulpit);
  dbms_output.put_line(rec2.code||' '||rec2.name);
exception
  when others then dbms_output.put_line(sqlerrm);
end;
```



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

```

-- 13/07.sql
declare
  rec1 teacher%rowtype;
  type address is record
  (
    address1 varchar2(100),
    address2 varchar2(100),
    address3 varchar2(100)
  );
  type person is record
  (
    code teacher.teacher%type,
    name teacher.teacher_name%type,
    homeaddress address
  );
  rec2 person;
begin
  rec2.code := 'ПРХК';
  rec2.name := 'Переходько Олеся';
  rec2.homeaddress.address1 := 'Беларусь';
  rec2.homeaddress.address2 := 'Пинск, Бресткая обл.';
  rec2.homeaddress.address3 := 'Набережная 7, кв.77';
exception
  when others then dbms_output.put_line(sqlerrm);
end;
/

```

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

Оператор IF

- if ... then ... end if;
- if ... then ... else ... end if;
- if ... then ... elsif ...then ...(elseif ... then ...) ... else ... end if;

Оператор CASE

case x

when ... then ...;

when ... then ...;

when x between 13 and 20 then ...;

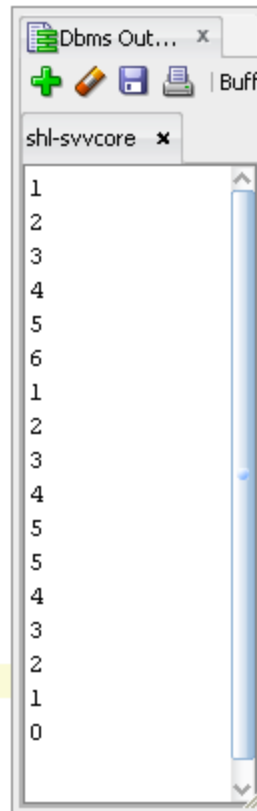
else ...;

end case;

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

Циклы loop, for, while

```
-- 11/14.sql
declare
  x pls_integer := 0;
begin
  -----
  loop
    x := x + 1;
    dbms_output.put_line(x);
  exit when x > 5;
  end loop;
  -----
  for k in 1..5
  loop
    dbms_output.put_line(k);
  end loop;
  -----
  while (x > 0)
  loop
    x := x - 1;
    dbms_output.put_line(x);
  end loop;
  -----
end;
/
```



- for – нельзя измен.знач.в цикле
- loop – обяз.указ.когда вых. из цикла
- while – вып. пока вып.усл.цикла

Выход из цикла:

- exit – безуслов.выход
- exit when – выход при вып.усл.
- goto – выход во внешн.контекст

39. Курсоры. Виды курсоров. Схемы обработки курсора.

Курсор – объект БД, позв работать с записями построчно

Курсор Oracle – указатель на область в PGA, в кот. хран: строки запроса, число строк, ук-ль на разобранный запрос в общем пуле

Виды:

- явный (объяв разработчиком), неявный (нет объяв, open fetch close)
- стат (выражение опр при компил), динам (выраж опр при выполнении)
- может возвр 1/0/неск строк
- для повторного созд. рез набора для др. значений параметров, курсор надо закрыть и открыть

Операторы управления курсором:

- 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.

Ошибки неявного курсора:

- no_data_found – не возвр строк вообще
- too_many_rows – более 1 строки

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

40. Курсоры. Атрибуты курсора. Курсоры с параметрами.

Атрибуты:

%ISOPEN – открыт ли (у неявного всегда false)

%FOUND – true, если строки были встав/уд/выбраны

%NOTFOUND – наоборот

%ROWCOUNT - № тек строки

явные курсоры с парам:

```

cursor cur (capacity int) is
  select * from aud where cap > capacity
begin
  for aum in cur(80)
  loop ...output... end loop;

```

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

Курсорные переменные – структуры д-х, указ. на курсорный объект

- для передачи курсора в качестве параметра
- чтобы отложить связь курсора с SELECT-запросом до выполнения OPEN

Если курсор переменной объявлен с помощью REF CURSOR без **RETURN** – она мб связана с любым запросом, иначе – только с запросом, возвращающим результат точно соотв. числу и ТД в записи после фразы return

```

declare
  type tlesson type is ref cursor return tlesson%rowtype;
  xcurs tlesson_type;
  rec_tlesson tlesson%rowtype;
begin
  open xcurs for select * from svvcore.tlesson;
  fetch xcurs into rec_tlesson;

```

sys_refcursor - является краткой записью создание курсорной переменной (невозможно накладывать ограничения RETURN)

Параметры Oracle, связанные с курсорами

cursor_space_for_time = {TRUE|FALSE} – большой объем памяти для курсоров и никогда не освобождается. Применяется для увеличения скорости работы курсоров при наличии памяти для разделяемого пула.

cursor_sharing = {EXACT|SIMILAR|FORCE}

open_cursors - максимальное количество открытых курсоров.

session_cached_cursors – максимальное количество кэшируемых курсоров для сессии.

42. Курсоры. Курсорные подзапросы.

```

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;
  aut auditorium_type.auditorium_type%type;
  txt varchar2(1000);
  aum auditorium.auditorium%type;
begin
  open curs_aut;
  fetch curs_aut into aut, curs_aum;
  while (curs_aut%found)
  loop
    txt := rtrim(aut)||':';
    loop
      fetch curs_aum into aum;
      exit when curs_aum%notfound;
      txt := txt||','||rtrim(aum);
    end loop;
    dbms_output.put_line(txt);
    fetch curs_aut into aut, curs_aum;
  end loop;
  close curs_aut;
exception
  when others
  then dbms_output.put_line(sqlerrm);
end;
```

```

ЛК: ,236-1,313-1,324-1,408-2,103-4,105-4,107-4,110-4,111-4,132-4,02Б-4,229-4,314-4,
ЛБ-К: ,206-1,301-1,413-1,423-1,304-4
ЛК-К: ,114-4
ЛБ-Х:
ЛБ-СК:
```

43. Курсоры. Использование конструкции CURRENT OF в курсорах.

Если планируете обновить/удалить записи, на которые ссылается SELECT FOR UPDATE:

UPDATE имя_таблицы

SET set_clause

WHERE CURRENT OF имя_курсора;

или

DELETE FROM имя_таблицы

WHERE CURRENT OF имя_курсора;

это позволяет обновить/удалить запись, которая была в курсоре последней

44. Курсоры. Динамические курсоры.

execute immediate – одностроч запросы и ddl-команды

open for, fetch, close – динам многострочные запросы

* для улучш производительности выполнения sql-выражений можно использовать динамические курсоры со связанными переменными

* позволяет повторно использовать разобранные SQL-выражения из разделяем. пула

```
EXECUTE IMMEDIATE 'INSERT INTO dept (deptno, dname, loc) VALUES (:deptno, :dname, :loc)' USING deptno_in, dname_in, loc_in;
```

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

ROWID – псевдостолбец, являющий уникальный идентификатор строки.

*уникальный не только в рамках таблицы, но и в рамках БД.

*упрощает работу с БД, т.к. позволяет однозначно идентифицировать любую строку таблицы, что позволяет удалять/измен. строки таблицы без первичного ключа.

*поиск по rowid – самый быстрый

!нельзя применять при разработке приложений, расчит. на работу с БД других типов.

ROWNUM – псевдостолбец, кот. умеет нумеровать строки в возвращ.рез-тах.

!нельзя напрямую использовать в запросе (вернет ошибку)

*позволяет вводить ограничение на количество выводимых записей

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

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

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

Два типа исключительных ситуаций:

- программно-определяемые исключения
- предопределенные (стандартные) исключения

Генерация исключений:

- ошибка, сгенерированная сервером
- ошибка в результате действий пользователя

- ошибка, сгенерированная приложением пользователю

sqlerrm – функция, возвращает сообщ об ошибке

sqlcode – функция, возвращает № ошибки

ошибка, сгенерированная сервером / приложением / в результате действия юзера

Обработка исключений – перехват ошибки в секции исключений

47. Принцип распространения исключений в PL/SQL.

Инструкция RAISE_APPLICATION_ERROR.

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

raise_application_error – команда, перехватывает выполнение текста блока

- определена в пакете dbms_standard
- может присвоить сообщ об ошибке

При выполнении процедуры:

- выполнение блока прерывается
- изменения в аргументах in out / out откатываются
- изменения в глобальной структуре не откатываются – надо явно rollback

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

Встроенные функции:

- Числовые функции: abs, trunc, round, floor, asin, acos, sign, greatest...
- Символьные функции: ascii, asciistr, chr, unistr, concat, initcap, instr...
- Функции по работе с датами: current_date, current_timestamp, sysdate, localtimestamp, next_day, last_day, months_between, new_time, tz_offset...
- Конвертирование: convert, to_number, to_char, to_date
- Функции обработки ошибок: sqlerrm, sqlcode

Внизу явно не все их очень много

Для строк:

- `rpad (str1, x, str2)` – возвращ.стр1 дополненную справа строкой 2 до размера x
- `rtrim (str1, str2)` – возвр.стр1, в кот.удалены правые крайние иденичные стр2.
- `upper (str)` – все символы прописные

Для даты:

- `trunc (d, [формат])` – возвращ.дату d, усеч.до ед.указ.в формате
- `sysdate` – тек.дата и время

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

Встроенные функции:

- Числовые функции: `abs`, `trunc`, `round`, `floor`, `asin`, `acos`, `sign`, `greatest...`
- Символьные функции: `ascii`, `asciistr`, `chr`, `unistr`, `concat`, `initcap`, `instr...`
- Функции по работе с датами: `current_date`, `current_timestamp`, `sysdate`, `localtimestamp`, `next_day`, `last_day`, `months_between`, `new_time`, `tz_offset...`
- Конвертирование: `convert`, `to_number`, `to_char`, `to_date`
- Функции обработки ошибок: `sqlerrm`, `sqlcode`

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_INSTR` определяет местоположение вхождения шаблона в строку
- `REGEXP_REPLACE` заменяет шаблон выражения на заданный
- `REGEXP_SUBSTR` выделяет из строки шаблон
- `REGEXP_COUNT` определяет количество вхождений

Метасимволы привязки:

метасимвол	описание	пример	Результат

^	К началу строки	REGEXP_LIKE(str,'^t')	test11 => true 11123345 => f
\$	К концу строки	REGEXP_LIKE(str,'\$5')	test11 => false 11123345 => true
Квантификатора + операторы повтора			
*	Встреч.0+	REGEXP_REPLACE(str,'11*', '1')	test11 => test1
?	0/1		
+	1+	REGEXP_LIKE(str,'5+')	test11 => false
{m}	m раз	REGEXP_LIKE(str,'3{2}')	11123345 => true
{m,}	по крайней мере m		
{m,n}	От m до n		
.....			

50. 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

Виды коллекций: Массив переменной длины VARRAY, Вложенная таблица (nested tables), Ассоциативный массив (associative array).

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

Коллекции типа VARRAY всегда ограничены

Вложенные таблицы и ассоциативные массивы не ограничены.

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

Массивы VARRAY всегда являются плотными

Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными

Ассоциативные массивы могут быть как разреженными, так и плотными в зависимости от способа их заполнения

Работа с коллекциями:

1. Объявление коллекций

2. Инициализация коллекций

- 2.1. Явно с помощью конструктора

- 2.2. Неявно при выборке из базы данных

- 2.3. Прямым присвоением переменной с другой коллекции такого же типа

1. Добавление и удаление элементов

- 3.1. Вложенные таблицы и массивы переменной длины – сначала увеличить размер при помощи функции EXTEND, а затем присвоить значения новым элементам

- 3.2. Ассоциативный массив – присвоение значения новому элементу

Массивы переменной длины – одномерные, связанные коллекции однотипных -элементов

Доступны в рамках PL/SQL и в БД

Являются плотными

```

declare
    type myarraytype is varray(3) of number;
    va myarraytype;

begin
    for i in 1..3 loop
        va(i) := 1;
    end loop;

exception
    when others then dbms_output.put_line(sqlerrm);
end;

```

51. Коллекции. Вложенные таблицы.

- Коллекция – структура данных, содержащая элементы одного типа. Элементом коллекции может быть как скалярная величина, так и композитные данные
- 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 плотной, если все ее элементы, определены и каждому из них присвоено некоторое значение
- Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными

Вложенные таблицы (Nested tables)– одномерные, несвязанные коллекции однотипных элементов. Вложенную таблицу можно рассматривать как одномерный массив, в котором индексами служат значения целочисленного типа.

Свойства:

Могут использоваться как в SQL (тип столбца в таблице), так и в PL\SQL коде

- Содержат однородные данные, т.е. все строки имеют одинаковую структуру данных

- Требуется инициализация, при использовании в PL\SQL
- Порядок элементов не зафиксирован
- Используется память PGA (для всех 3 типов коллекций)
- При попытке чтения элемента с несуществующим индексом -> исключение NO_DATA_FOUND

Использование вложенных таблиц в SQL

CREATE TYPE XX_TYPE_CHAIR IS TABLE OF VARCHAR2(100);

Методы для вложенных таблиц:

Exists(n)- Возвращает TRUE, если элемент существует; FALSE — если элемент не существует

Count - Возвращает текущее количество элементов

First/Last - Возвращает индекс первого и последнего элемента

Prior(n) - Возвращает индекс предыдущего элемента

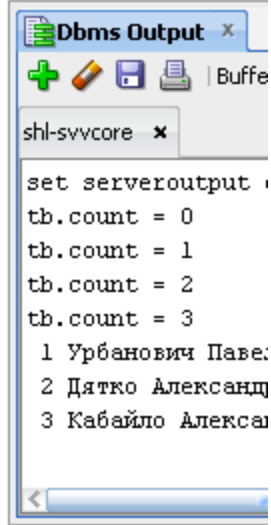
Next(n) - Возвращает индекс следующего элемента

Extend - Добавляет один пустой элемент в коллекцию

Delete - Удаляет все элементы в коллекции, не вызывает исключение при несуществующем индексе

```
-- 14/01.sql
declare
  type tperson is record
  (
    name teacher.teacher_name%type,
    pulpit teacher.pulpit%type
  );
  type mytable is table of tperson;
  tb mytable := mytable();
  rec tperson; -- := tperson('xxx','xxxxxx');
begin
  dbms_output.put_line('tb.count = '||tb.count);

  tb.extend;
  dbms_output.put_line('tb.count = '||tb.count);
  tb(1).name := 'Урбанович Павел Павлович';
  tb(1).pulpit := 'ИСиТ';
  tb.extend;
  dbms_output.put_line('tb.count = '||tb.count);
  tb(2).name := 'Дятко Александр Аркадьевич';
  tb(2).pulpit := 'ИСиТ';
  tb.extend;
  dbms_output.put_line('tb.count = '||tb.count);
  tb(3).name := 'Кабайло Александр Серафимович';
  tb(3).pulpit := 'ИСиТ';
  for i in 1..tb.count
  loop
    dbms_output.put_line(' '||i||' '||tb(i).name||' '||tb(i).pulpit);
  end loop;
exception
  when others then dbms_output.put_line(sqlerrm);
end;
/
```



52. 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

Виды коллекций: Массив переменной длины VARRAY, Вложенная таблица (nested tables), Ассоциативный массив (associative array).

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

Коллекции типа VARRAY всегда ограничены

Вложенные таблицы и ассоциативные массивы не ограничены.

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

Массивы VARRAY всегда являются плотными

Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными

Ассоциативные массивы могут быть как разреженными, так и плотными в зависимости от способа их заполнения

Работа с коллекциями:

1. Объявление коллекций

2. Инициализация коллекций

- 2.1. Явно с помощью конструктора

- 2.2. Неявно при выборке из базы данных

- 2.3. Прямым присвоением переменной с другой коллекции такого же типа

1. Добавление и удаление элементов

- 3.1. Вложенные таблицы и массивы переменной длины – сначала увеличить размер при помощи функции EXTEND, а затем присвоить значения новым элементам

- 3.2. Ассоциативный массив – присвоение значения новому элементу

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

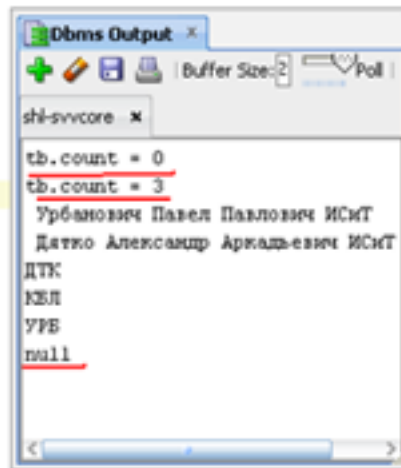
Доступны только в рамках PL/SQL

Изначально являются разреженными, индекс может принимать непоследовательные значения

```

declare
    type tperson is record (name teacher.teacher_name%type, pulpit teacher.pulpit%type);
    type mytable is table of tperson index by teacher.teacher_name%type;
    tb mytable;
    ntx teacher.teacher_name%type;
    rec tperson;
begin
    dbms_output.put_line('tb.count = '||tb.count);
    tb('УФБ').name := 'Урбанович Павел Павлович';
    tb('УФБ').pulpit := 'ИСИТ';
    tb('ДТК').name := 'Дятко Александр Аркадьевич';
    tb('ДТК').pulpit := 'ИСИТ';
    tb('КЭЛ').name := 'Кабайко Александр Серафимович';
    tb('КЭЛ').pulpit := 'ИСИТ';
    dbms_output.put_line('tb.count = '||tb.count);
    dbms_output.put_line(' '||tb('УФБ').name||' '||tb('УФБ').pulpit);
    rec := tb('ДТК');
    dbms_output.put_line(' '||rec.name||' '||rec.pulpit);
    ntx := tb.first;
    if ntx is not null then dbms_output.put_line(ntx);
    else dbms_output.put_line('null');
    end if;
    ntx := tb.next(ntx);
    if ntx is not null then dbms_output.put_line(ntx);
    else dbms_output.put_line('null');
    end if;
    ntx := tb.next(ntx);
    if ntx is not null then dbms_output.put_line(ntx);
    else dbms_output.put_line('null');
    end if;
    ntx := tb.next(ntx);
    if ntx is not null then dbms_output.put_line(ntx);
    else dbms_output.put_line('null');
    end if;
    exception
    when others then dbms_output.put_line(sqlerrm);
end;

```



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

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

Для создания процедур необходима привилегия create procedure

Типы данных параметров: PL/SQL или программно-определенный, Не может быть ограничен по размеру, Размер определяется через вызывающую программу или через связанное объявление переменной

Типы параметров: IN, OUT, IN OUT.

При выполнении: Значения OUT устанавливаются в NULL ; Значения IN OUT остаются неизменными; При ошибке присвоения для параметров откатываются, кроме NOCOPY.

IN, IN OUT можно не задавать при вызове.

```
create or replace procedure svvcore.xsum(  
    min_cy in svvcore.auditorium.auditorium_capacity%type, -- минимальная вместимость  
    max_cy in out svvcore.auditorium.auditorium_capacity%type, -- максимальная вместимость  
    n_aud out number -- количество  
)  
  
is  
    m_max_cy svvcore.auditorium.auditorium_capacity%type;  
    m_n_aud number := 0;  
begin  
    select count(*), max(auditorium_capacity) into m_n_aud, m_max_cy from svvcore.auditorium  
    where auditorium_capacity >= min_cy and auditorium_capacity <= max_cy;  
exception  
    when others then dbas_output.put_line(sqlerrm);  
end xsum;
```

Передача параметров:

Позиционный – каждое значение в списке аргументов вызова ставится в соответствие формальному параметру по порядку. *Empid_to_name(23, name, surname);*

Именованный – явно связывает аргументы при вызове с параметрами по именам. *Empid_to_name(in_id =>23, out_name=> name, out_surname =>surname);*

Можно комбинировать оба метода, пока позиционные аргументы стоят слева.

Empid_to_name(23, name, out_surname =>surname);

Вызов процедур:

```
SVUCORE@sh1> exec :max_cy:=10;
```

Процедура PL/SQL успешно завершена.

```
SVUCORE@sh1> exec xsum(20,:max_cy,:n_aud);
```

Процедура PL/SQL успешно завершена.

```
SVUCORE@sh1> select :max_cy, :n_aud from dual;
```

:MAX_CY	:N_AUD
-----	-----
	0

```
SVUGUEST@sh1>
```

```
SVUGUEST@sh1>
```

```
SVUGUEST@sh1> CALL svvcore.xsum(20,:max_cy,:n_aud);
```

Вызов завершен.

```
SVUGUEST@sh1> select :max cy, :n aud from dual;
```

:MAX_CY	:N_AUD
-----	-----
60	5

```
-- 16/03.sql
```

```
declare
```

```
max_cy number(3) := 80;
```

```
n_aud number(3):= 0;
```

```
begin
```

```
svvcore.xsum(20, max_cy, n_aud);
```

```
dbms_output.put_line(max_cy || ', ' || n_aud);
```

```
xsum(n_aud=>n_aud, min_cy =>20, max_cy=>max_cy);
```

```
dbms_output.put_line(max_cy || ', ' || n_aud);
```

```
xsum(20, n_aud=>n_aud, max_cy=>max_cy);
```

```
dbms_output.put_line(max_cy || ', ' || n_aud);
```

```
end;
```


Значения по умолчанию - DEFAULT

```
-- 16/04.sql
create or replace procedure svvcore.xsum(
    min_cy in svvcore.auditorium.auditorium_capacity%type default 20, -- минимальная из
    max_cy in out svvcore.auditorium.auditorium_capacity%type, -- максимальная вместимость
    n_aud out number -- количество
)
is
    no_max_cy exception;
begin
    select count(*), max(auditorium_capacity) into n_aud, max_cy from svvcore.auditorium
    where auditorium_capacity >= min_cy and auditorium_capacity <= max_cy;
    if n_aud is null
    then raise no_max_cy;
    end if;
exception
    when no_max_cy then return;
    when others then dbms_output.put_line(sqlerrm);
end xsum;
```

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

- PL / SQL — полностью переносимый, высокопроизводительный язык обработки транзакций.
- PL / SQL предоставляет встроенную, интерпретируемую и независимую от ОС среду программирования.

Функции — эти подпрограммы возвращают одно значение; в основном используется для вычисления и возврата значения.

1) Автономная функция создается с помощью оператора **CREATE FUNCTION**

2) Список необязательных параметров содержит имя, режим и типы параметров. IN представляет значение, которое будет передано извне, а OUT представляет параметр, который будет использоваться для возврата значения вне процедуры

3) Функция должна содержать инструкцию **возврата**

4) Предложение **RETURN** указывает тип данных, который вы собираетесь вернуть из функции

5) Функция *body* содержит исполняемую часть.

6) Ключевое слово AS используется вместо ключевого слова IS для создания отдельной функции.

7) ПРИМЕР

```
CREATE OR REPLACE FUNCTION totalCustomers
```

```
RETURN number IS
```

```
total number(2) := 0;
```

```
BEGIN
```

```
SELECT count(*) into total
```

```
FROM customers;
```

```
RETURN total;
```

```
END;
```

Функция называется детерминированной, если при одном наборе параметров IN и IN OUT она всегда возвращает одно и то же значение. Функция должна быть объявлена с ключевым словом DETERMINISTIC.

Суть в том, что у нас например есть параметры *param_1* и *param_2* и каждый раз когда мы будем их передавать мы всегда будем получать одно и то же, поэтому с ключевым словом *deterministic* мы получим детерменированную функцию, которая гораздо оптимизированнее !

- Pipeline function- Конвейерная обработка позволяет табличной функции быстрее возвращать строки и может уменьшить объем памяти, необходимый для кэширования результатов табличной функции. Конвейерные табличные функции включают фразу **PIPELINED** и используют вызов **PIPE ROW**, чтобы вытолкнуть строки из функции, как только они создадутся, вместо построения табличной коллекции. Заметим, что вызов **RETURN** пустой, поскольку нет никакой коллекции, возвращаемой из функции.
- - Построение конвейерной табличной функции.

```
CREATE OR REPLACE FUNCTION get_tab_ptf (p_rows IN NUMBER) RETURN  
t_tf_tab PIPELINED AS
```

```
BEGIN
```

```

FOR i IN 1 .. p_rows LOOP
PIPE ROW(t_tf_row(i, 'Description for ' || i));
END LOOP;
RETURN;
END;

```

- - Тестирование

```

SELECT *
FROM TABLE(get_tab_ptf(10))
ORDER BY id DESC;

```

- Параметры по умолчанию – ключевое слово default – задаем какое-то дефолтное значение для переменной, которую передаем в функцию

Параметры бывают 1) IN параметр Этот параметр используется для ввода данных в подпрограммы В вызывающем операторе эти параметры могут быть переменной, литеральным значением или выражением, например, это может быть арифметическое выражение типа «5 * 8» или «a / b», где «a» и «b» являются переменными По умолчанию параметры имеют тип IN. 2) Выходной параметр Этот параметр используется для получения выходных данных из подпрограмм. В операторе вызова эти параметры всегда должны быть переменной для хранения значения из текущих подпрограмм. 3) Параметр IN OUT Этот параметр используется как для ввода, так и для получения выходных данных из подпрограмм. Это переменная чтения-записи внутри подпрограмм. Их значения могут быть изменены внутри подпрограмм.

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

Пакеты – коллекция pl/sql объектов, сгруппированных вместе.

Нахера Что делает: 1) скрытие инфы, 2) объектно-ориентированный дизайн, 3) постоянство объектов в транзакциях, 4) улучшенная производительность.

Можно включать: процедуры, функции, константы, исключения, курсоры, переменные, TYPE выражения, записи, REF курсоры

Спецификация пакета (package) – обязательна, содержит список объектов для общего доступа из других модулей или приложения

```
3 CREATE OR REPLACE PACKAGE time_pkg IS
    FUNCTION GetTimestamp RETURN DATE;
    PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE);
END time_pkg;
```

Реализация пакета (package body) – содержит реализацию процедур и функций из спецификации; приватные объекты; секцию инициализации.

```
3 CREATE OR REPLACE PACKAGE BODY time_pkg IS
    StartTimeStamp DATE := SYSDATE;
    -- StartTimeStamp is package data.
    -- Function
3    FUNCTION GetTimestamp RETURN DATE IS
    BEGIN
        RETURN StartTimeStamp;
    END GetTimestamp;
    -- Procedures
3    PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE)
    IS
    BEGIN
        StartTimeStamp := new_time;
    END ResetTimestamp;
    -- Initialization section
3    BEGIN
        null;
    EXCEPTION
        WHEN NO_DATA_FOUND
            THEN dbms_output.put_line('not initialized');
END time_pkg;
```

Вызов пакета: *Package_name.package_element*;

Пакетные данные – структуры данных в пакете.

Пакетные переменные являются глобальными данными (сохраняют своё состояние от одной транзакции к другой)

56. Процедурные объекты. Триггеры. Виды триггеров. Классификация, порядок выполнения и предикаты триггеров.

Триггеры замещения. Привилегии для создания триггеров. Включение/отключение триггеров. Псевдозаписи old и new.

Триггер – особый вид процедур, кот. срабатывают по запускающему их событию

Применение триггеров:

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

Виды триггеров:

- *по привязанному объекту:* на таблицу, на представлении (instead of trigger)
- *по событиям запуска:* insert, update, delete
- *по области действия:* уровень оператора, ур. записи, составные
- *по времени срабатыванию:* before (до записи в журнал), after

Уровни триггеров:

- for each row – сраб для каждой измененной строки
- по умолч (операторный ур) – сраб 1 раз на триггерное событие

Порядок вып:

- операторные BEFORE
- для каждой строки BEFORE
- выполняется оператор
- для каждой строки AFTER
- операторный AFTER

Предикаты триггеров:

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

create or replace trigger ..

after insert or update or delete on ..

begin

if inserting then ...

elsif updating then ...

elsif deleting then ...

Триггеры замещения: INSTEAD OF – только для предст! только уровня строки!

Привилегии:

- create (any) trigger
- alter any trigger
- drop any trigger
- administer database trigger – созд/изм/уд системные триггеры

Вкл / откл триггеров:

- alter trigger { disable | enable }
- всех для таблиц : alter table .. { enable | disable } all triggers;
- компиляция триггера : alter trigger .. compile
- переименование триггера

Псевдозаписи new, old:

операция срабатывания	OLD.column	NEW.column
insert	null	новое знач
update	старое знач	новое знач
delete	старое знач	null

57. Секционирование таблиц. Виды секционирования.

Секционирование – метод хранения сегмента данных (напр. таблица) физически в виде нескольких сегментов (логически – монолитная структура).

Секция – отдельный сегмент. Могут находиться в разных ТП (сл-но на разных дисках). Отдельные ТП могут находиться в OFFLINE, не нарушая работоспособности всей таблицы.

Секции имеют: 1) общие логические атрибуты – имена, типы, ограничения столбцов, ROW_MOVEMENT; 2) собственные физические атрибуты – например атрибуты ТП.

Операции над секциями:

1. **RENAME, DROP, MOVE, ADD**
2. **SPLIT** – разбиваем секцию.

```
ALTER TABLE DEACore.sales SPLIT PARTITION sales_11q4
AT ('01-JUL-2012') INTO
(PARTITION sales_sp1 TABLESPACE data01,
PARTITION sales_sp2 TABLESPACE data02)
```

3. **MERGE** – объединяем секции.

```
ALTER TABLE DEACore.sales MERGE PARTITIONS
sales_11q1, sales_11q2 INTO sales_m1;
```

4. **EXCHANGE** – пишем данные секции в какую-то таблицу (вроде бы).

```
ALTER TABLE DEACore.sales EXCHANGE PARTITION sales_11q2
WITH TABLE DEACore.my_sales WITHOUT VALIDATION;
```

Мы можем читать данные прямо из секции:

```
SELECT * FROM DEACore.sales PARTITION (sales_11q1);
```

Если читаем так, то называется ассоциативная ссылка (хз почему):

```
SELECT * FROM DEACore.sales PARTITION FOR ('02-FEB-2011');
```

Виды секционирования:

1. **Диапазонное (RANGE)** – определяем секции ручками. Обязательно указываем MAXVALUE секцию. Проблема: При загрузке новых данных в

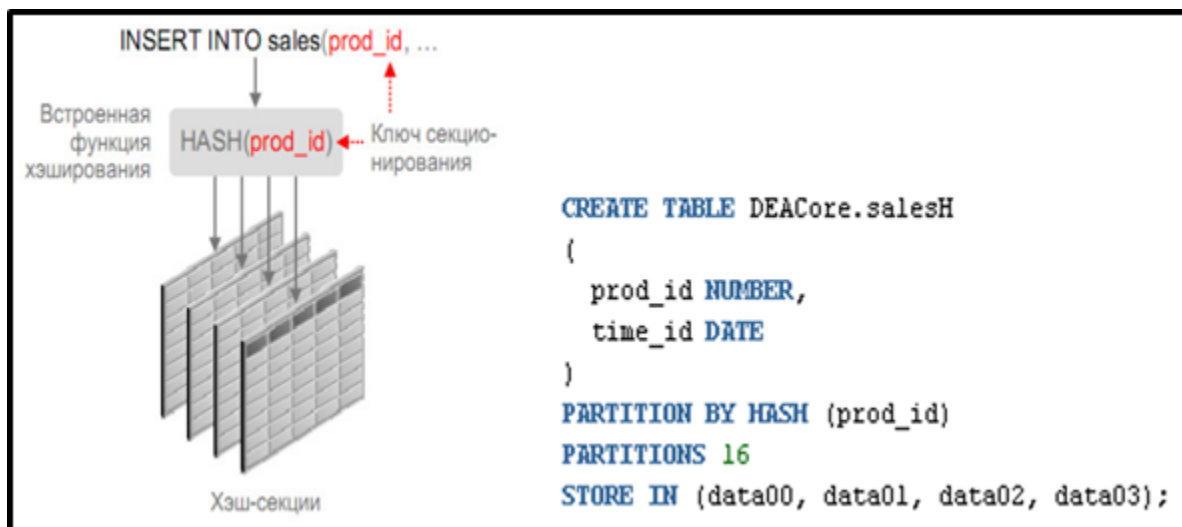
таблицу нужно постоянно расщеплять секцию MAXVALUE

```
CREATE TABLE DEACore.sales
(
  prod_id NUMBER,
  time_id DATE
)
PARTITION BY RANGE (time_id)
(
  PARTITION sales_llq1 VALUES LESS THAN ('01-APR-2011') TABLESPACE data00,
  PARTITION sales_llq2 VALUES LESS THAN ('01-JUL-2011') TABLESPACE data01,
  PARTITION sales_llq3 VALUES LESS THAN ('01-OCT-2011') TABLESPACE data02,
  PARTITION sales_llq4 VALUES LESS THAN ('01-JAN-2012') TABLESPACE data03,
  PARTITION sales_max VALUES LESS THAN (maxvalue) TABLESPACE data04
);
```

2. **Интервальное** – Создается единственная диапазонная секция без MAXVALUE. Новые секции будут создаваться автоматически (по 1ой операции INSERT, не попадающей в диапазоны существующих секций).
Предустановленный интервал можно изменить через alter table set interval

```
create table t3 (x1 nvarchar2(50), x2 number(3))
partition by range (x2)
interval (100) store in (users)
(
  partition t11 values less than (101)
);
```

3. **Хэш-секционирование** – на основе заполняемых данных (например, primary key) вычисляется хэш, который потом юзается для быстрого доступа.



4. **Списочное** – разбиваем по списку дискретных значений

```
CREATE TABLE DEACore.salesL
(
  prod_id NUMBER,
  time_id DATE,
  group_id NUMBER
)
PARTITION BY LIST (group_id)
(
  PARTITION sales_l1 VALUES (3, 9),
  PARTITION sales_l2 VALUES (5, 4),
  PARTITION sales_other VALUES (DEFAULT)
);
```

5. **Композитное** – Если данных много настолько, что сами секции большие, то их тоже можно секционировать. Такие секции могут секционироваться по другому критерию секционирования. Допускается 2 уровня секционирования (секции и подсекции).

```

CREATE TABLE DEACore.salesC                                Hash2
(
    prod_id NUMBER,                                         Hash3
    time_id DATE,
    cust_id NUMBER                                           Hash4
)
PARTITION BY RANGE (time_id)
SUBPARTITION BY HASH(cust_id)
SUBPARTITIONS 4 STORE IN (data00, data01, data02, data03)
(
    PARTITION sales_llq1 VALUES LESS THAN ('01-APR-2011'),
    PARTITION sales_llq2 VALUES LESS THAN ('01-JUL-2011')
);

```

Зачем секционирование:

1. **Отказоустойчивость** – данные могут располагаться на разных дисках
2. **Производительность** – тут по двум причинам. 1) Читаем из разных секций параллельно. 2) Например, у нас два клиента – Microsoft и Google. Мы храним их инфу в одной таблице. Следовательно, когда нам потребуется какая-то выборка данных по Google'у, мы будем ещё читать и данные Microsoft. При секционировании мы можем разделить эти данные на разные секции и сразу читать данные только Google'а, хотя логически всё будет в одной куче.

58. 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 оператора SQL и завершается, когда вы фиксируете или отказываете транзакцию. Фиксация (committing) транзакции закрепляет проведенные вами изменения, а откат (roll back) – отменяет их.

- Как только вы зафиксировали транзакцию, все прочие транзакции других пользователей, которые начались после нее, смогут видеть изменения, проведенные вашими транзакциями
- Когда транзакция вообще не может выполняться (скажем, из-за отключения электропитания), то она вся целиком должна быть отменена. Oracle

откатывает все изменения, проведенные предшествующими операторами SQL, возвращая данные в исходное состояние

ACID properties- основные правила которых подчиняются транзакции:

1. Atomicity- Выполняются все задачи транзакции или ни одна из них
2. Consistency - Транзакция переводит базу данных из одного согласованного состояния в другое согласованное состояние(последовательность)
3. Isolation - Эффект транзакции не виден другим транзакциям, пока транзакция не будет зафиксирована
4. Durability - Изменения, внесенные совершенными транзакциями, являются постоянными(долговечность).

Автономная транзакция является независимой сделкой , которая может быть вызвана из другой транзакции, которая является основной транзакцией.

Автономные транзакции полезны для действий, которые должны выполняться независимо, независимо от того, фиксирует или откатывает вызывающая транзакция. Например, в транзакции покупки акций вы хотите зафиксировать данные о клиентах независимо от того, проходит ли покупка акций в целом. Кроме того, вы хотите регистрировать сообщения об ошибках в таблице отладки, даже если вся транзакция откатывается.

Автономные транзакции имеют следующие характеристики:

1. Автономная транзакция не видит незафиксированных изменений
2. Изменения в автономной транзакции видны другим транзакциям после фиксации автономных транзакций
3. Автономные транзакции могут запускать другие автономные транзакции.

РАСПРЕДЕЛЕННЫЕ транзакции - В отличие от транзакции в локальной базе данных, распределенная транзакция изменяет данные в нескольких базах данных.

59. Обработка заданий. Системные пакеты обработки заданий в Oracle.

DBMS_JOB – поддержка упр-ния заданиями

Задание – процедура, PL-SQL блок, внешняя процедура

*вып-ся в фон.реж., надо задать кол-во одноврем.вып-мых процессов

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=9;
```

SUBMIT – создание задания

```
declare job_number user_jobs.job%type;
begin
dbms_job.submit(job_number, 'begin up_students_grades_JOB; end;',
    SYSDATE, 'SYSDATE + 60/86400'); -- 24 * 60 * 60 = 86 400
COMMIT;
SYS.dbms_output.put_line(job_number);
end;
```

ISUBMIT – создание задания с номером

```
--isubmit
declare job_number user_jobs.job%type;
begin
dbms_job.isubmit(43, 'begin up_students_grades_JOB; end;',
    SYSDATE, 'SYSDATE + 60/86400'); -- 24 * 60 * 60 = 86 400
COMMIT;
select job into job_number from user_jobs;
SYS.dbms_output.put_line(job_number); --43
end;
```

REMOVE – удаление задания из очереди

RUN – немедленное выполнение задания в пользовательском сеансе

```
-- RUN
declare job_number user_jobs.job%type;
begin
dbms_job.run(43);
COMMIT;
select job into job_number from user_jobs;
SYS.dbms_output.put_line(job_number); --43
end;
```

BROKEN – разрушение задания (16)

```
begin
dbms_job.broken(43,true, null);
end;
```

INSTANCE – указание экземпляра

NEXT_DATE – изменение времени выполнения

INTERVAL – изменение интервала выполнения

```
-- next_date -- interval
begin
dbms_job.next_date(43,SYSDATE + 2/24); -- через 2 часа
dbms_job.interval(43,SYSDATE + 3); -- через 31 дня
COMMIT;
end;
```

CHANGE – изменение параметров задания

```
-- change
declare job_number user_jobs.job%type;
begin
dbms_job.change(43, -- номер задания
               null, --what
               null, -- следующее исполнение
               'SYSDATE + 360/86400'); --интервал
COMMIT;
end;
```

WHAT – изменение задания

DBMS_SCHEDULER

Schedule – расписание

Расписание — это спецификация того, когда и как часто база данных должна выполнять задание.

```
---scheduler
begin
dbms_scheduler.create_schedule(
  schedule_name => 'Sch_1',
  start_date => '01/01/2017 12:00:00',
  repeat_interval => 'FREQ=DAILY',
  comments => 'Sch_1 DAILY at 12:00');
end;
```

Program – программа

Программа содержит метаданные о задании Scheduler. Программа включает имя, тип (например, код PL/SQL или сценарий оболочки UNIX) и действие программы, которое представляет собой действительное имя процедуры или исполняемого сценария.

```
---program
begin
dbms_scheduler.create_program(
  program_name => 'Pr_1',
  program_type => 'STORED_PROCEDURE',
  program_action => 'up_students_grades_JOB',
  number_of_arguments => 0,
  enabled => false,
  comments => 'up_students_grades_JOB');
end;
```

Job – плановая программа

Job— это задача, которая планируется для однократного или многократного автоматического запуска. Задание содержит спецификацию того, что и когда должно быть выполнено.

Одним из ограничений пакета *DBMS_JOB* является то, что он может выполнять только задания на базе PL/SQL, и его нельзя использовать для планирования запуска системных сценариев либо исполняемых файлов. Scheduler позволяет использовать сценарии PL/SQL, сценарии оболочки операционной системы,

программы Java, и родные двоичные исполняемые файлы для выполнения запланированных заданий.

```
---job
begin
dbms_scheduler.create_job(
  job_name => 'J_1',
  program_name => 'Pr_1',
  schedule_name => 'Sch_1',
  enabled => true);
end;
```

Представления словаря: DBA_JOBS, USER_JOBS,
USER_SCHEDULER_SCHEDULES,
USER_SCHEDULER_PROGRAMS, USER_SCHEDULER_JOBS,
USER_SCHEDULER_JOB_LOG.

► Права:

- Роль SCHEDULERADMIN
- Права на объекты

60. Системные пакеты Oracle.

Системные пакеты Oracle:

- устанавливается во время установки Oracle
- используется для расширения функц. возможностей Oracle
- владелец пакетов – SYS
- написаны на C или plsql

```
SELECT Object_Type, OWNER, Object_Name FROM DBA_Objects_AE
WHERE Object_Type = 'PACKAGE'
ORDER BY Owner, Object_Name;
```

Пакеты APEX:

- apex – oracle application express – среда разработки веб-прил

- apex_custom_auth – проверка подлинности управл сеансом
- apex_application – использ глоб пер-ных
- apex_item – созд эл-тов форм на основе sql-запроса
- apex_util – различ утилиты состояния сеанса, файлов, авториз ...

Пакеты DBMS:

- dbms_advanced_rewrite – перехват и замена sql-запросов
- dbms_advisor – часть набора экспертной с-мы для реш проблем производительности, связ. с компонентами сервера БД
- dbms_sqltune – сбор статистики, используется при анализе производительности sql-операторов
- dbms_appl_info – присвоение имени процессу для удобства мониторинга и отладки

DBMS_AQ – пакет для Advanced Queuing – система обмена очередями сообщений

DBMS_TRANSFORM – пакет для Oracle Advanced Queuing – преобразования данных

DBMS_DEFER – вызов удаленных процедур

DBMS_OFFLINE_OG, DBMS_REPCAT – поддержка репликации

DBMS_REFRESH – создание группы материализованных представлений для обновления как единого целого

DBMS_CRYPTO – пакет шифрования данных. Поддерживаются алгоритмы шифрования - DES, AES, RC4, 3DES, 3DES-2KEY

DBMS_CUBE - поддерживает развертывание кубических материализованных представлений из существующих реляционных материализованных представлений

DBMS_DIMENSION - отображение информации об измерениях – иерархических связях между данными

Data Warehouse - предметно-ориентированная база данных, предназначенная для отчетов и/или OLAP

DBMS_DEBUG – пакет для отладки PL/SQL-кода на сервере

и так далее...

```
SELECT Object_Type, OWNER, Object_Name FROM DBA_Objects_AE  
WHERE Object_Type = 'PACKAGE'  
AND Object_Name LIKE 'DBMS_%'  
ORDER BY Owner, Object_Name;
```

- устанавливается во время установки Oracle
- используется для расширения функц. возможностей Oracle
- владелец пакетов – SYS
- написаны на C или plsql

```
SELECT Object_Type, OWNER, Object_Name FROM DBA_Objects_AE  
WHERE Object_Type = 'PACKAGE'  
ORDER BY Owner, Object_Name;
```

Пакеты APEX:

- apex – oracle application express – среда разработки веб-прил
- apex_custom_auth – проверка подлинности управл сеансом
- apex_application – использ глоб пер-ных
- apex_item – созд эл-тов форм на основе sql-запроса
- apex_util – различ утилиты состояния сеанса, файлов, авториз ...

Пакеты DBMS:

- dbms_advanced_rewrite – перехват и замена sql-запросов
- dbms_advisor – часть набора экспертной с-мы для реш проблем производительности, связ. с компонентами сервера БД
- dbms_sqltune – сбор статистики, используется при анализе производительности sql-операторов
- dbms_appl_info – присвоение имени процессу для удобства мониторинга и отладки

DBMS_AQ – пакет для Advanced Queuing – система обмена очередями сообщений

DBMS_TRANSFORM – пакет для Oracle Advanced Queuing – преобразования данных

DBMS_DEFER – вызов удаленных процедур

DBMS_OFFLINE_OG, DBMS_REPCAT – поддержка репликации

DBMS_REFRESH – создание группы материализованных представлений для обновления как единого целого

DBMS_CRYPTO – пакет шифрования данных. Поддерживаются алгоритмы шифрования - DES, AES, RC4, 3DES, 3DES-2KEY

DBMS_CUBE - поддерживает развертывание кубических материализованных представлений из существующих реляционных материализованных представлений

DBMS_DIMENSION - отображение информации об измерениях – иерархических связях между данными

Data Warehouse - предметно-ориентированная база данных, предназначенная для отчетов и/или OLAP

DBMS_DEBUG – пакет для отладки PL/SQL-кода на сервере

и так далее...

```
SELECT Object_Type, OWNER, Object_Name FROM DBA_Objects_AE
```

```
WHERE Object_Type = 'PACKAGE'
```

```
AND Object_Name LIKE 'DBMS_%'
```

```
ORDER BY Owner, Object_Name;
```