1. Основные понятия теории баз данных: база данных, система управления базами данных, основные требования к информации в БД, модели данных, логическая схема БД, основная терминология реляционных баз данных.
2. База данных (БД): Организованная коллекция связанных данных, которая хранится и обрабатывается с помощью системы управления базами данных (СУБД). БД предоставляет удобный способ хранения и доступа к данным.
3. Система управления базами данных (СУБД): Программное обеспечение, которое управляет созданием, хранением, обновлением и извлечением данных в БД. MS SQL и Oracle являются примерами СУБД.
4. Основные требования к информации в БД: Информация в БД должна быть точной, полной, актуальной и безопасной. Она должна быть доступна только авторизованным пользователям и подвержена контролю целостности.
5. Модели данных: Существует несколько моделей данных, но реляционная модель является наиболее широко используемой в MS SQL и Oracle. Она представляет данные в виде таблиц, состоящих из строк (кортежей) и столбцов (атрибутов).
6. Логическая схема БД: Это описание структуры данных в БД без учета физического размещения. Логическая схема определяет таблицы, их атрибуты и связи между таблицами.
7. Основная терминология реляционных баз данных: В реляционных БД используются такие понятия, как таблицы (отношения), строки (кортежи), столбцы (атрибуты), первичные ключи, внешние ключи, индексы и запросы на языке SQL для манипуляции и извлечения данных.
8. Язык SQL.

Доступ к д-м осущ в виде запросов на языке SQL – декларативный (непроцедурный) язык: с его пом. можно сформул, что надо получить, но нельзя указать КАК.

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

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

\* DCL (control) – grant, revoke

\* TCL – commit, rollback, savepoint

1. Системные базы данных: master, msdb, model, tempdb.

**master**: информация о сервере; информация о базах данных, логины, настройки сервера и другие метаданные. Является ключевой базой данных, необходимой для запуска SQL Server.

**msdb**: Она содержит информацию о резервном копировании, восстановлении, планировщике задач, истории задач и уведомлениях оператора.

**model:** База данных model является шаблоном для создания новых баз данных. Когда в SQL Server создается новая база данных, она использует структуру и настройки из базы данных model.

**tempdb**:tempdb: База данных tempdb используется для временного хранения временных таблиц, переменных, индексов и промежуточных результатов запросов.

1. Структура файла базы данных в SQL Server. Файловые группы.

* **Файлы данных (data files):** Это файлы, в которых хранятся фактические данные таблиц и индексов. Файлы данных имеют расширение .mdf (primary data file) или .ndf (secondary data file).
* **Журнальные файлы (log files):** Это файлы, в которых хранится информация о транзакциях, выполняемых на базе данных. Журнальные файлы помогают обеспечить целостность данных и возможность восстановления. Журнальные файлы имеют расширение .ldf.

**Структура файла:**

- File Header - содержит метаданные и информацию о файле базы данных, такие как версия SQL Server, размер файла, идентификатор файла и другие параметры.

- Структура страниц: Файл базы данных разбивается на фиксированные по размеру страницы. Каждая страница имеет размер 8 килобайт и используется для хранения данных и метаданных. Структура страницы включает заголовок страницы и различные типы данных, включая данные таблицы, индексы, метаданные объектов и другую информацию.

- Аллокационные единицы (Allocation Units): Аллокационная единица представляет собой логическую группу страниц, которая выделяется для хранения определенного типа данных. В SQL Server используются следующие аллокационные единицы:

* + Data Pages: Это страницы, на которых хранятся данные таблиц и индексов.
  + Index Pages: Это страницы, на которых хранятся структура и данные индексов.
  + IAM Pages (Index Allocation Map): Это страницы, которые содержат информацию об использовании страниц индекса.
  + GAM Pages (Global Allocation Map): Это страницы, которые содержат информацию об использовании экстента данных в файле базы данных.
  + SGAM Pages (Shared Global Allocation Map): Это страницы, которые содержат информацию об использовании экстента индексов в файле базы данных.
  + PFS Pages (Page Free Space): Это страницы, которые содержат информацию о доступном пространстве на страницах файла базы данных.

**Файловые группы (file groups):** Файловые группы представляют собой логические контейнеры для организации файлов базы данных.

* Первичная файловая группа (primary file group): При создании базы данных автоматически создается первичная файловая группа, которая содержит основные файлы данных (.mdf) и журнальные файлы (.ldf).
* Дополнительные файловые группы (secondary file groups): Вы можете создать дополнительные файловые группы, чтобы разместить дополнительные файлы данных (.ndf) на отдельных физических носителях или разделах. Это позволяет распределять данные по различным дискам для улучшения производительности или обеспечения отказоустойчивости.

1. Нормализация таблиц базы данных. Нормальные формы таблиц.

1NF (Первая нормальная форма): Каждая ячейка таблицы содержит только одно значение, и столбцы имеют уникальные имена.

2NF (Вторая нормальная форма): Кроме того, что таблица находится в 1NF, каждый неключевой столбец должен зависеть от полного составного первичного ключа.

3NF (Третья нормальная форма): Кроме того, что таблица находится в 2NF, неключевые столбцы не должны зависеть друг от друга.

4NF (Четвертая нормальная форма): Кроме того, что таблица находится в 3NF, должны быть удалены зависимости, основанные на многозначных зависимостях.

5NF (Пятая нормальная форма): Кроме того, что таблица находится в 4NF, должны быть удалены зависимости, основанные на зависимостях от множества значений.

**Нормализация** помогает устранить избыточность данных и обеспечивает структурированное представление данных в базе данных, повышая эффективность и поддерживаемость базы данных.

1. Таблицы и их типы данных Microsoft SQL Server.
2. **Целочисленные типы данных:**
   * INT: 32-битное целое число.
   * BIGINT: 64-битное целое число.
   * SMALLINT: 16-битное целое число.
3. **Десятичные и числовые типы данных:**
   * DECIMAL(p, s): Фиксированное точное число с заданным количеством цифр до и после десятичной точки.
   * NUMERIC(p, s): То же, что и DECIMAL.
   * FLOAT: Плавающее число с плавающей точкой.
4. **Символьные и строковые типы данных:**
   * CHAR(n): Фиксированная строка заданной длины n.
   * VARCHAR(n): Строка переменной длины с максимальной длиной n.
   * TEXT: Длинный текстовый блок переменной длины.
5. **Даты и времена:**
   * DATE: Дата без времени.
   * TIME: Время без даты.
   * DATETIME: Дата и время.
6. **Булев тип данных:**
   * BIT: Логическое значение true/false или 1/0. (так же может быть NULL)
7. **Другие типы данных:**
   * BINARY: Фиксированный двоичный блок данных.
   * VARBINARY: Двоичный блок данных переменной длины.
   * XML: XML-данные.
8. Подзапросы. Конструкции in, exists, all, any, some.

Подзапросы являются мощным инструментом в SQL для создания более сложных запросов. Они позволяют включать один запрос (подзапрос) внутри другого запроса.

**IN:** проверка наличия значения в результате подзапроса

Select \* from xxx where col1 in (1, 2, 3)

**EXISTS:** проверка на существование рез-та подзапроса

**ALL:** возвращает true если все значения в подзапросе удовлетворяю условию

SELECT \* FROM table WHERE column > ALL (SELECT column FROM another\_table).

**ANY или SOME:** возвращают true если хотя бы одно значение удовлетворяет условию. SELECT \* FROM table WHERE column > ANY (SELECT column FROM another\_table).

1. Группировка данных с использованием cube, rollup

**CUBE:** генерирует все возможные комбинации итогов, включая итоги по каждому отдельному столбцу, комбинации по парам столбцов, тройкам столбцов и т.д

**ROLLUP:** создания итоговых данных с иерархической структурой. Он генерирует итоги на разных уровнях иерархии данных. Начиная с самого детального уровня, он создает итоги по всем комбинациям столбцов, а затем постепенно "сворачивает" данные до более высоких уровней

1. Операторы union (all), intersect, except.

**UNION –** объединение рез-ов двух или более запросов в один, удаляет дубликаты

**UNION ALL** – также объединяет рез-ты но не удаляет дубликаты

**INTERSECT –** находит общие строки двух запросов

**EXCEPT –** находит строки, которые есть в первом запросе, но нет во втором

1. Операторы соединения таблиц.

**INNER JOIN –** возвращает только строки в которых есть соответствие из обоих таблиц

SELECT table1.column1, table2.column2

FROM table1 INNER JOIN table2 ON table1.column = table2.column;

**LEFT (OUTER) JOIN -**  возвращает все строки из левой таблицы и те строки из правой, которые удовлетворили условию

**RIGHT (OUTER) JOIN –** возвращает все строки из правой таблицы и те строки из левой что удовлетворили условию

**FULL (OUTER) JOIN –** возвращает все строки из обеих таблиц независимо от совпадающих значений, однако если нет совпадающих значений, для соответствующих столбцов будет возвращено NULL

1. Язык T-SQL. Пакеты. Объявление переменных. Операторы присвоения Оператор цикла while.

**Пакеты:** В T-SQL пакеты являются логическими контейнерами для хранения и организации объектов базы данных, таких как хранимые процедуры, функции, представления и другие объекты. Пакеты упрощают управление и поддержку кода, позволяя группировать связанные объекты в одном месте.

**Переменные:** DECLARE @VariableName DataType;

**Операторы присвоения:** SET @VariableName = Value;

WHILE Condition

BEGIN -- Блок кода, который будет выполняться

END;

1. Язык T-SQL. Операторы print, if-else, case. Операторы begin-end, waitfor и return.

**PRINT**

PRINT 'Значение переменной: ' + CAST(@Variable AS VARCHAR);

**IF ELSE**

IF Condition

BEGIN

-- Блок кода, выполняемый при выполнении условия

END

ELSE

BEGIN

-- Блок кода, выполняемый при невыполнении условия

END

**WAITFOR**

WAITFOR DELAY '00:00:05'; -- Приостановить выполнение на 5 секунд

**RETURN**

CREATE FUNCTION dbo.MyFunction()

RETURNS INT

BEGIN

DECLARE @Result INT;

-- Логика функции

RETURN @Result;

END;

1. Язык T-SQL. Обработка ошибок в конструкциях try-catch. Функция RAISERROR.

BEGIN TRY

-- Блок кода, который может вызвать ошибку

END TRY

BEGIN CATCH

-- Обработка ошибки

END CATCH

Функция **RAISERROR:** Функция RAISERROR используется для генерации пользовательских сообщений об ошибках или предупреждений.

BEGIN TRY

-- Блок кода, который может вызвать ошибку

END TRY

BEGIN CATCH

DECLARE @ErrorMessage VARCHAR(255);

SET @ErrorMessage = 'Произошла ошибка: ' + ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

1. Локальные и глобальные временные таблицы в SQL Server.

**Локальные временные таблицы**: Локальные временные таблицы видны только внутри текущей сессии или соединения пользователя и существуют только во время выполнения запроса или сеанса. Они создаются с префиксом "#" перед их именем.

CREATE TABLE #TempTable

(

ID INT,

Name VARCHAR(50)

);

**Глобальные временные таблицы**: Глобальные временные таблицы видны для всех сеансов или соединений пользователей в пределах одной базы данных. Они создаются с префиксом "##" перед их именем.

CREATE TABLE ##GlobalTempTable

(

ID INT,

Name VARCHAR(50)

);

Важно отметить, что как локальные, так и глобальные временные таблицы автоматически удаляются при завершении сеанса, выполняющего их создание, или при явном удалении таблицы.

1. Курсоры в SQL Server. Объявление курсора. Общая схема работы с курсором: declare, open, fetch, close, deallocate. Типы курсоров.

**Курсоры** в SQL Server используются для обработки результатов запроса построчно.

Схема работы с курсором:

1) обьявление курсора

DECLARE CursorName CURSOR FOR

SELECT Column1, Column2

FROM TableName;

2) Открытие курсора

OPEN CursorName;

3) получение данных из курсора

FETCH NEXT FROM CursorName INTO @Variable1, @Variable2;

4) закрытие курсора

CLOSE CursorName;

5) освобождение ресурсов курсора

DEALLOCATE CursorName;

Типы курсоров:

* **Динамический курсор –** изменения данных отображаются в динамике
* **Статический курсор –** данные выбраны один раз и произошедшие изменения не видны

- **локальные курсоры** – создаются и используются в пределах текущей сессии или соединения, а также ограничены областью видимости соединения.

Declare xxx cursor global

For select \* from t1

Read\_only <- read\_only значит что ре-ты доступны для чтения

- **глобальные курсоры** – видны для всех сеансов и соединений в пределах базы данных, могут одновременно использоваться из различных сеансов

* @@FETCH\_STATUS
  + 0 – успешная выборка,
  + -1 – вышли за диапазон таблицы,
  + -2 – запись удалена после открытия курсора

1. Хранимые процедуры и функции T-SQL. Создание хранимых процедур. Передача параметров. Входные и выходные параметры.

Хранимые процедуры позволяют создавать повторно-используемый код

CREATE PROCEDURE ProcedureName

@InputParameter DataType,

@OutputParameter DataType OUTPUT

AS

BEGIN

-- Код хранимой процедуры

END;

* Входные параметры используются для передачи значений в хранимую процедуру или функцию.
* Выходные параметры используются для возврата значений из хранимой процедуры.
* Оба типа параметров могут быть переданы при вызове хранимой процедуры или функции.

**Вызов процедуры** – EXEC ProcedureName @Parameter1, @Parameter2;

1. Хранимые процедуры и функции T-SQL. Виды функций. Создание функций. Передача параметров.

Функции позволяют создать повторно используемый код, возвращающий нек-ые значения.

**Типы функций:**

* Встроенные
  + Математические
  + Строковые
  + Работа с датами
* Пользовательские

-**Не допускается применение:**

* + DDL-операторов
  + DML-операторов изменяющих данные (INSERT, DELETE, UPDATE)
  + конструкций TRY/CATCH
  + транзакций

**вызов функций** – fn() или select \* from fn()

1. Индексы. Назначение и применение индексов. Виды индексов. Применение различных видов индексов. Оптимизация запросов.

**Индекс** представляет собой отдельную физическую структуру данных, которая позволяет получать быстрый доступ к одной или нескольким строкам данных.

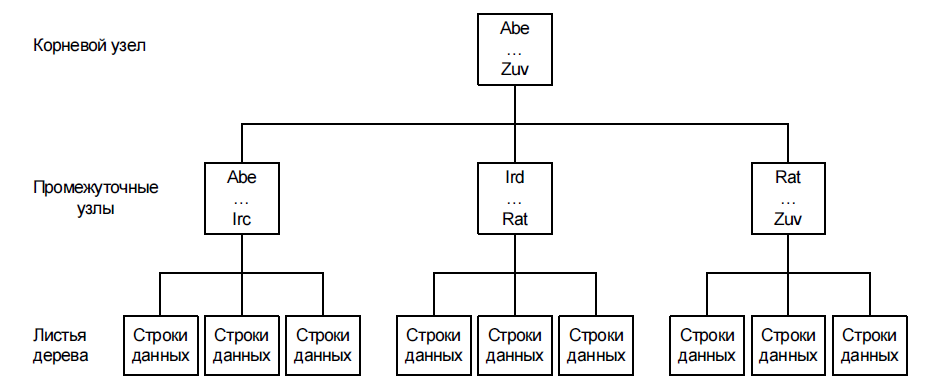
* Индексы сохраняются в страницах индексов
* Для каждой индексируемой строки имеется элемент индекса, который сохраняется на странице индексов
* Каждый элемент индекса состоит из ключа индекса и указателя
* Индексы создаются по сбалансированному дереву B+
* B+-дерево имеет древовидную структуру, в которой все листья находятся на расстоянии одинакового количества уровней от вершины дерева
* Это свойство поддерживается при добавлении или удалении данных в индексированном столбце

**Типы индексов :**

* Кластеризованные индексы
* Некластеризованные индексы

**Кластеризованный:**

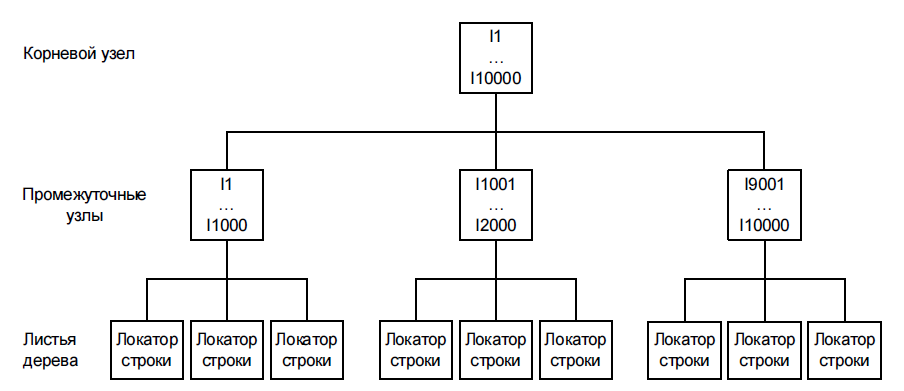
* Определяет физический порядок данных в таблице
* Может только один для одной таблицы
* Таблица перестраивается в порядке индекса
* Листья дерева индекса содержат страницы данных



* Создается по умолчанию для каждой таблицы, для которой определен первичный ключ
* Уникальный – в столбце, для которого определен кластеризованный индекс, каждое значение данных может встречаться только один раз
* Если кластеризованный индекс создается для столбца, содержащего повторяющиеся значения, СУБД принудительно добавляет четырехбайтовый идентификатор к строкам, содержащим дубликаты значений

**Некластеризованный**

* физически находится отдельно от таблицы
* страницы листьев состоят из ключей индекса и закладок
* может быть несколько для одной таблицы
* не изменяет физическое упорядочивание строк таблицы



* Если есть кластеризованный индекс, то закладка некластеризованного индекса показывает B+-дерево кластеризованного индекса таблицы
* Если нет кластеризованного индекса, закладка идентична RID — Row Identifier, состоящего из:
  + Адреса файла, в котором хранится таблица,
  + Адреса физического блока (страницы), в котором хранится строка,
  + Смещения строки в странице.
* **SP\_HELPINDEX -** получить перечень индексов, связанных с заданной таблицей

Create clustered index #indexname on #tablename (colname asc)

1. План запроса. Этапы обработки select запроса. Понятие стоимости запроса. Понятия селективности и плотности. Индексы. Реорганизация, перестроение, включение и отключение индексов.

**План запроса** - это план, составленный оптимизатором запросов для выполнения SELECT запроса в базе данных. Он определяет оптимальный способ получения данных и обработки запроса.

**Этапы обработки** SELECT запроса:

* Парсинг: Запрос разбирается и проверяется на синтаксическую и семантическую корректность.
* Оптимизация: Оптимизатор запросов создает план выполнения запроса, который оптимально использует ресурсы базы данных.
* Выполнение: Запрос выполняется на основе созданного плана выполнения, данные извлекаются и возвращаются в результате.

**Стоимость запроса** - это оценка ресурсов, необходимых для выполнения запроса, таких как время выполнения и использование памяти.

**Селективность** - это мера количества выбранных строк относительно общего количества строк в таблице. Она указывает, насколько "селективным" является условие в запросе.

**Плотность** - это мера количества уникальных значений в столбце таблицы. Чем ниже плотность, тем более уникальные значения содержит столбец.

**Индексы** - это структуры данных, создаваемые на таблицах для ускорения операций поиска и сортировки данных.

**Операции над индексами :**

Реорганизация индекса - Процесс физической перестройки индекса, который помогает устранить фрагментацию и повысить эффективность использования индекса.

Перестроение индекса

1. Триггеры. Типы триггеров. Создание after-триггера.

**DML триггеры**

**DDL триггеры**

**Before, after, instead of** триггеры.

1. Триггеры. Создание и назначение instead of-триггеров. Таблицы inserted, deleted.

Inserted и deleted хранят вставленные и удаленные строки во время dml операции, с ними может в дальнейшем работать триггер.

1. Транзакции. Свойства ACID. Транзакции. Уровни изолированности транзакций. Функция trancount.

**ACID:**

* Atomicity - Атомарность
* Consistency - Согласованность
* Isolation - Изолированность
* Durability - Долговечность

**@@trancount –** уровень вложенности транзакции.

**Уровни изоляции:**

* SET TRANSACTION ISOLATION LEVEL:
  + READ UNCOMMITTED
  + READ COMMITTED
  + REPEATABLE READ
  + SERIALIZABLE

1. Операторы TCL. Привилегии. Роли. Назначение привилегий.

Create role role\_name;

Grant select, insert, update to role\_name;

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

Дистрибутивы:

1. **Oracle Database Standard Edition** (SE): Предоставляет основные функции базы данных для небольших и средних предприятий.
2. **Oracle Database Enterprise Edition (**EE): Полнофункциональный дистрибутив, включающий дополнительные возможности, такие как расширенные инструменты управления и высокую доступность.
3. **Oracle Database Express Edition** (XE): Бесплатный дистрибутив, предназначенный для разработки, тестирования и небольших продакшн сред.

на oracle.com выбрать Database 12c > Download > 2 файла для вашей ОС

Инсталляция:

запуск Oracle Database 12c Installer

\* указ. email (необяз)

\* Skip software updates

\* Create and configure database

\* Server class / Desktop class

\* Single instance database installation

\* Typical / Advanced install

\* Create New Windows User : name, password

\* Oracle database name : orcl.be.by

Administrative password : ….

SID : orcl

SID (System ID) – уник. имя, кот. однозначно идентиф. экземпляр/БД

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

**Осн. системные пользователи:**

\* **SYS** – предопред. привилегир. юзер ранга админа БД, кот. явл. владельцем ключевых ресурсов БД Oracle

\* **SYSTEM** – предопред. привилег. юзер, кот-му принадлежат ключевые ресурсы БД Oracle

**Специальные сист. привилегии:**

\* **SYSDBA, SYSOPER** – спец. привилегии админа, позвол. выполнять базовые задачи администрирования: запуск и остановка экземпляра БД; создание, удаление, открытие, монтирование бд…

**DBA** – предопред. роль, кот. авто- создается для каждой БД oracle и содержит все сист. привилегии, кроме SYSDBA и SYSOPER

1. Понятия базы данных и экземпляра базы данных.

*БД* – набор физ. файлов в ОС (мб доступна в нескольких экземплярах)

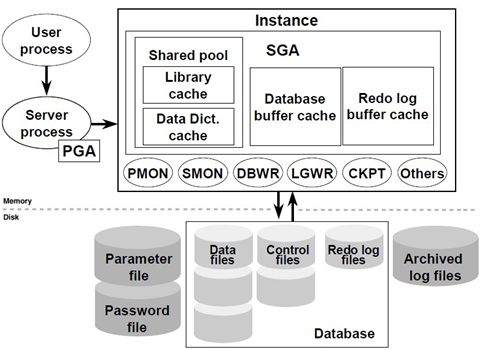
*Экземпляр* – набор процессов и область SGA (обесп доступ только к 1 БД)

**Экземпляр (инстанс):**

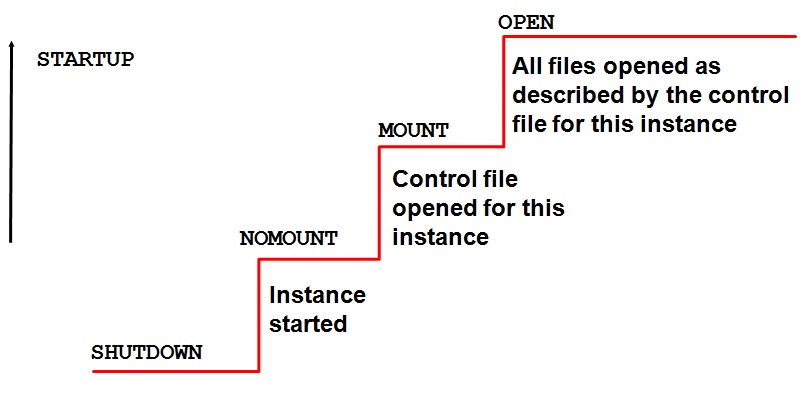
\* запущенный сервер (программа) СУБД

\* общая (глоб) область памяти : SGA и другие системные области памяти

\* фоновые процессы, предназн. для управления файлами БД



1. Запуск и останов экземпляра базы данных Oracle.



startup nomount | mount | open

shutdown normal | transactional | immediate | abort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| режим остановки | Abort | Immediate | Transactional | Normal |
| Разреш. новые соед | - | - | - | - |
| Ждать конца тек. сеанса | - | - | - | + |
| Ждать конца тек. транзакции | - | - | + | + |
| Контр. точка, закрытие файлов | - | + | + | + |

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

***Таблицы юзера*** – набор таблиц, созданных и обслуживаемых юзером

+ сод. инфу юзера

***Словарь д-х*** – набор таблиц, созданных и обслуживаемых сервером Oracle

+ сод. инфу о БД

***Словарь Oracle*** – набор таблиц и связ. с ними представлений, позвол. отследить внутр. структуру БД и деятельность СУБД oracle

\* созд при генерации БД

\* обнов сервером в фоновом режиме после вып. DDL-опер.

\* расп в системном ТС – **SYSTEM**

\* владелец – юзер **SYS**, нек. представления – **SYSTEM**

\* привил – *grant select any dictionary*

**Запросы к словарю д-х:**

- USER – объекты юзера

- ALL – объекты, к кот. юзер имеет доступ

- DBA – все объекты БД (для админа)

- V$ - производительность сервера

**Основные представления:**

dba\_users, \_profiles, \_roles

dba\_ (role | sys | tab | col) \_privs,

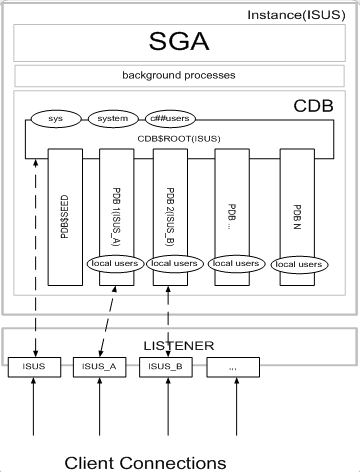
role\_ (role | sys | tab | col) \_privs,

user\_ (role | sys | tab) \_privs

1. Мультиарендная архитектура Oracle Multitenant.

***Oracle Multitenant*** – технология, позвол. запустить неск. независимых БД в рамках одного экземпляра

\* Каждая БД имеет свой набор ТС и набор схем, но при этом у них общая SGA и один набор серверных процессов

****\* БД изолированы, друг о друге ничего не знают, не конфликтуют

\* Словарь разбивается на 2 части: общую и локальную

**CDB** – Container DB

**PDB** – Pluggable DB

**\*** Можно создать неск. CDB – для разных версий ПО СУБД

**\*** Одну PDB можно переносить между CDB

**\*** В CDB создается главный контейнер Root (содержит метаданные)

**\*** В одной CDB м. создать до 252 PDB

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

***Файлы параметров*** – для хранения параметров экземпляра – SPFILE (бин, pfile – текст). Изменения примен при след. старте Oracle

\*select name, description from v$parameter;

***Упр. файлы*** – сод. имена осн. физ. файлов БД и нек. параметров – использ для поиска др. файлов ОС

\*для надежности созд.2 упр.ф. (на разн диск носит)

\*show parameter control; *--полная инфа*

\*select name from v$controlfile; *--расположение*

для измен: shutdown immediate -> скопир упр. файл -> измен парам CONTROL\_FILES в файле параметров -> startup open

***Файл паролей*** – для аутентиф. админов БД. Можно созд, пересозд, измен.

\*просмот пользователей

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

***Файл трассировки*** – фиксир все события, происх. на сервере БД, в т.ч. критических. По их содержимому м. выяснить причину сбоя

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

***Файл д-х*** – файл, в кот.хран.все д-е в БД.

\*в кажд.БД хотя бы 1. (с 1 БД мб связан только 1 файл д-х)

***Журнал повт.вып-ния*** – файл ОС, в кот.фикс.измен.вносимые юзерами в БД

2+ файла ОС, примен. при восст. БД.

использ. циклически (сначала запись в 1й файл, 2, 3… 1)

select \* from v$logfile;

**Мультиплексирование журналов повтора** – поддержка неск. копий каждого журнала

\*select \* from v$log;

**SCN** (system change number) – сист. № измен.в БД (пред.селект (first\_change))

**Параметры ЖП** (указ.в CONTROLFILE)**:**

\*maxlogfiles – макс.кол-во групп ЖП

\*maxlogmembers – макс.кол-во файлов в группе

\*maxinstances – макс.кол-во инстансов

\*maxdatafiles – макс.кол-во файлов д-х

Добавить новой группы:

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

Добавление файла в группу:

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

Переключение журналов:

alter system switch logfile;

**Архивы журналов повтора:**

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

- archivelog;

- noarchivelg;

\*жизненно важны при восстановл.

\*примен послед-но (если 1 пропущен – ост.не м.исп-ся)

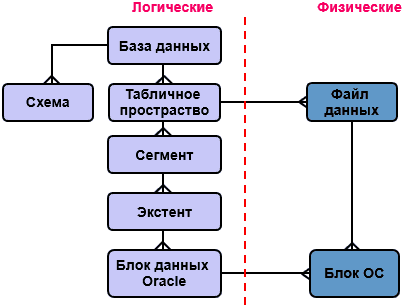
Архивы:

\*включить процесс архив-ния (shutdown, mount, .., open)

alter database archivelog;

\*архивный журнал появл.после переключ.оперативн.журнала

1. Абстрактная модель Oracle. Логическая структура внешней памяти.



в Oracle исп.2уровневая организ.БД.

\*объекты верх.ур-ня – **лог.стр-ры** БД.

\*объекты нижн.ур-ня – **физ.стр-ры** БД.

Лог.стр-ра сост из польз.компонет: табл.пр-во, сегмент, экстент и блок д-х.

**Табл.пр-во -** логич. структура хранения д-х, контейнер сегментов.

**Сегмент** – область на диске, выделяемая под объекты. Расп. в ts

**Экстент** – непрерывный фрагмент дисковой памяти

**Блок** – мин единица объема памяти, примен. при записи и чтении д-х

*Сегмент сост. из экстентов, экстент – из блоков.В одном ts мб неск сегментов. Сегмент если не секционирован, расп в одном ts.*

**Сегмент:**

хранит только д-е, поэтому он создается только при добавлении данных. При удалении строк из таблицы, сегмент не удал. При удалении таблицы измен. имя сегмента, и инфа об удалении запис. в словарь БД. Мб восстановлен с пом. механизма RECYCLEBIN. Для удал recyclebin-сегмента примен. к-да PURGE.

\***сегмент д-х** (хран.табл + их строки (1 табл – 1 сегмент))

\***сегмент индексов** (хранят индексы (ключ.столбец + rowid))

\***сегмент отката** (строятся с-мой и исп.при вып-нии транз.)

\***временный сегмент** (врем. раб. обл. для промеж. стадий обраб.запроса)

**Экстент:**

длину выделяемого экстента вычисляет СУБД. Если при создании ts задана опция UNIFORM, то все экстенты имеют один. длину.

БД сост из ТС. ТС сост из файлов данных. ТС содержит сегменты.

- *Сегменты* сост. из экстентов. Сегмент привязан к ТС, но его данные могут находиться в разных файлах д-х, образующих это ТС.

- *Экстент* – набор расположенных рядом на диске блоков. Экстент целиком находится в одном ТС и в одном файле данных этого ТС.

- *Блок* – наим. единица управления пр-вом в БД. Блок – наим единица ввода-вывода, использ. сервером.

1. Абстрактная модель Oracle. Физическая структура внешней памяти.

Физ.БД организ.как совок.файлов, созд-мых обычными ср-ми ОС. Т.о. основой физ. ур-ня – *файл*.

2 группы компонентов физ.ур:

\*системные объекты

Spfile, Controlfile, трассировки, Журналы Повтора + архивы

\*объекты юзера (фал.д-х)

1. Абстрактная модель Oracle. Структура SGA.

**SGA (system global area):**

- группа областей разделяемой памяти

- содерж данные и упр-щую инфу для одного экземпляра Oracle

- совместно всеми сервер. и фон. процессами

|  |  |
| --- | --- |
| **Структура:** | **Осн пулы SGA:** |
| \* java pool (БД сод много java кода + кучи)  \* large pool (для || запросов)  \* shared pool (plsql и sql код + инфа словарей)  \* фикс. область SGA  \* буф. кэш (копии блоков, из файлов д-х)  \* буфер ЖП (запуск д-е повт вып-ния, кот записаны в этот буфер) | \* java pool  \* large pool  \* shared pool  \* *streams pool*  \* *“null” pool* |

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

select \* from v$sga;

1. Абстрактная модель Oracle. Серверные процессы Oracle.

***Процесс*** – механизм в ОС, вып. послед-сть шагов.

**Серверный процесс** – проц, вып-щийся на основании клиентского запроса. Клиент – любая прога, кот.пыт.подключится к БД. Каждый юзер, подключенный к БД им.свой *отд.серв.проц*, сущ-ший на протяж.сеанса. Первичное установл.соезд.происх.через спец.сервер.проц.**Oracle Net Listener.**

В нек.сит, когда прилож и БД раб.на 1 компе, м.объед.юзерский проц и серв.проц в один для уменьш.сист.издержек. Однако, когда прилож и БД раб.на различн.ПК , польз.проц.всегда связ.в БД через отдельный серверн.проц.

**Серверн.проц.** м.:

\* вып sql-операторы

\*чит.файлы д-х

\*возвр. результаты

\*осущ.поиск в кэше

**Пример:**

Отправка sql-запроса select \* from students;

\*произв.синтакс.разбор

\*помещ.в раздел.пул (находит в пуле)

\*созд.план запроса и вып.его

\*при необх., произв.поиск д-х в буф.кэше или запрашивает в кэш с диска

**Oracle Net Listener** – проц.на стор.С,кот.прослуш. вх.запр.К на соед.с экземплярм.

**Listener** – программа-сервер, кот.просл.TCP-порт, приним-щая запросы на соед.

В рез-те успешн.работы Listener устан.соед.между К и обработчиком запросов экземпляра. ПО умолч. TCP-порт – 1521.

1. Абстрактная модель Oracle. Фоновые процессы Oracle.

**Фоновой процесс** – спец.группа проц для обеспеч.произв-сти и поддержки работы большого числа юзеров.

Фон.проц запуск.вместе с БД и вып.различн.задачи обслуживания.

**LREG** – период.регистр.сервисов в проц.Listener;

**DBWn** (database writer) – запис. изменения в ФД (3, ck, -)

**LGWR** (log writer) – запис измен в ЖП (до фикс)

**CKPT** – вып-ет проц.checkpoint;

\* при shutdown, alter system checkpoint, switch logfile, backup

\* запис.инфу о контр.точке в управл.файл

**ARCn** (archiver) – копир.файлы ЖП после переключ.группы журналов

**PMON** (process monitor) – отвеч.за очистку после ненорм.закрытия подключ.

**RECO** (recovery) – разреш.проблемн.связанных с распред.транзакциями.

1. Процесс-слушатель Oracle и его основные параметры.

**Прослушиватель (Listener) –** компонент, кот.позвол.устан.соед.между К и БД. Один прослушивать м.обслуж.неогран.кол-во БД. Прослушиватель управляется файлом listener.ora. Мб сконфигурир.только 1 файл listener.ora, на С мб натсроено неск.прослушивателей, и этот файл м.обслуживать всех просл.

Если на С неск.прослушив. => с целью обеспеч.отказоуст/баланс обращений

По умолч, прослуш.им.имя LISTENER и созд.при установке oracle.

Если неск.просл – каждый д.им.уник.имя.

**Парам:**

Протокол, порт, имя хоста.

Для упр-ния прослуш.исп.утилита **lsnrctl**; Для прослуш.доступны след.команды:

Start – запуск проц.прослушивания

Stop – остановка

Status – тек.статус прослушивателя.

+oracle net listener (12)

1. Сетевые настройки Oracle. Установление соединения по сети.

Oracle поддерж.2 режма соед: dedicated(выделенный) и shared(разделяемый) server.

**Dedicated** (по умолч) – для кажд.К – отд.серверн.проц.

**Shared** – в кач-ве обработчика выступает прога dispatcher, кот:

1. Получ запрос от К
2. Помещ.его во вх.очередь к раздел.С
3. Незанятый раздел.С извлек и обраб.запрос
4. После обраб.раздел.С помещ.рез-т обработки в вых.очередь
5. Из очереди рез-т извлек диспетчер
6. Диспетчер пересыл рез-т клиенту

За подключ отвеч 2 конфиг.файла:

\*listener.ora – для управления прослушивателем и связи К с СУБД(подробно выше)

\*tnsnames.ora – подробн.опис.подлкюч.к БД

**Виды подключений**:

\*простое подключ (basic) – явно указ.все пар-ры соед.

CONNECT имя/пароль@[//]хост[:порт][/имя\_службы]

\*локальное именование (tns)

В tncnames (host, port, service\_name) -> oracle net manager

\*LDAP-соед – с помощью службы каталогов.

\*local/bequeath-соед

\*только на С

\*м.соед.с пом.sqlplus/sqldeveloper

\*без указ.пар-ров соед

\*только с выдел.С

\*listener не задействован

1. Табличные пространства СУБД Oracle и их основные параметры.

***TS*** – логич. структура хранения д-х, контейнер сегментов

граничный объект БД (файлы – лог)

*С одним TS связаны 1/неск файлов ОС, с каждым файлом ОС только 1 TS*

\* permanent – хранение пост. объектов

– мб неск, 1 мб прописано в кач-ве ТС по умолч

– можно создать свои объекты в любом PTS, если не запрет

\* temporary – хранение врем. д-х

– мб неск

– приписано 1/неск юзерам, кот. могут там размещ. свои д-е

\* undo – хранение сегментов отката, исп. всегда один (указ. в SPFILE.ORA)

! м. указ для таблиц по умолч

! удаление табл.с опред.ТП: drop table xxx1 purge;

! вывести список ТП: select \* from dba\_tablespaces;

**Параметры:**

create [temp] tablespace ..

datafile (tempfile) путь…

size

auto\_extent on next

maxsize

\* smallfile / bigfile (128 ТБ)

\* logging / nologging / force logging (журналир измен)

\* online / offline (раб/нераб сост)

\* reuse

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

***Привилегия*** – право выполнять конкр. тип предложений SQL, или право доступа к объекту другого юзера

\* grant / revoke

\* нельзя выдавать в одном предложении

|  |  |
| --- | --- |
| **системные** | **объектные** |
| на именение с-мы: create table / trigger | на изм объекта: select |
| *WITH ADMIN OPTION* – дают право юзеру также назнач/отбир привил:  alter, analyze, audit, backup, create, drop, select (стараться редко исп)  any – для любого объекта;  all – для всех объектов | *WITH GRANT OPTION* – дают право юзеру также назначать/отбир привил (но объектные): alter, delete, execute, insert, update, select, references  снимает привил тот, кто их назначал |
| **объекты грантов:**  database, user, profile, tablespace, role, table, index, trigger, procedure, sequence, view | **объекты грантов:**  table, view, sequence, procedure |

***Роль*** – именованный набор привилегий

role, password\_required, authen\_type, common, oracle\_maintained

1. Пользователь СУБД Oracle и его основные параметры.

***Юзер*** – физ./юр. лицо, кот. имеет доступ к БД и пользуется услугами для получ. инфы

create user .. identified by 12345

default tablespace .. quota unlimited on ..

temporary tablespace ..

profile ..

account unlock

password expire

1. Профиль безопасности СУБД Oracle и его основные параметры.

***Профиль*** – коллекция атрибутов, связ. с использованием ресурсов и паролей, кот. мб назначена юзеру

create profile .. limit

password\_life\_time 180 -- дней жизни пароля

sessions\_per\_user 3 -- кол. сессий для юзера

failed\_login\_attempts 7 -- кол. попыток входа

password\_lock\_time 1 -- дней блока

password\_reuse\_time 10 -- через ск. дней повтор пароль

password\_grace\_time default -- кол-во дней предупр. о смене пароля

connect\_time 180 -- минут соед

idle\_time 30 -- минут простоя

1. Таблица в СУБД Oracle и ее основные параметры. Типы данных базы данных. Ограничения целостности в таблицах.
2. Временные таблицы СУБД Oracle.
3. Индексы базы данных СУБД Oracle. Виды и особенности применения индексов.
4. Последовательность СУБД Oracle и ее параметры.
5. Кластер и его параметры.
6. Представление в СУБД Oracle и его параметры.
7. Материализованное представление и его параметры.

*привил:* create materialized view

*\* build immediate* – сразу созд  
*\* start with* – когда начнет вып  
*\* next* – когда следующий (1/1440 = 1/24\*60 = минута)

(start with – next = периодичность) ***COMMIT !!!!***  
**Отличие:** Хранят не только запрос, но и какие-то данные, кот. обновляются на нек. временном промежутке

1. Частные и публичные синонимы СУБД Oracle.

**Синоним** – способ обращаться к объекту БД без указ. обяз. полной идентификации объекта (хост – экземпляр – владелец – объект)

\* частный принадлежит юзеру, кот. его создал

\* публичный исп. совместно всеми юзерами БД

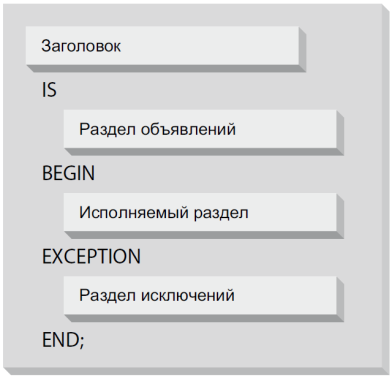
привил: create (public) synonym

словарь: dba.synonyms

м. указ. на: табл, проц, фун, послед, предст, пакеты, объекты в лок/уд БД

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

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

Нет имени (заголовка) – анонимный блок

DECLARE

Раздел объявл – не обяз. – объяв переменные

BEGIN

Исполняемый раздел – обяз., хоть один

EXCEPTION

Раздел исключений – не обяз.

END;

**Анонимный блок** – не может быть вызван из другого блока

Исп. как скрипт для вып PL/SQL

Простейший сост из к-ды *null* (не делает ничего, исп. когда специально хотим показать, что гасим всякую деят-сть)

\* не чувств к разбору (лучше маленьк, большими только sql-оп-ры: insert)

**Именованные блоки** – процедуры и функции

**when others** – обработка любого исключения

**sqlerrm** – функция, кот. возвр сообщ об ошибке

**sqlcode** – функция, кот. возвр № ошибки (обычно 5-значный)

блоков **when** ск. угодно, самый последний – **when others**

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

**Типы данных:**

- скалярные символ/строка, число, булев, дата/время

- ссылочные

- составные

- биг объекты

**Константы:** с пом constant

Псевдостолбцы **CURRVAL** (текущее значение) и **NEXTVAL** (следующее значение) для последовательностей.

Псевдостолбец **LEVEL** – текущий уровень иерархии в таблицах с учетом иерархического дерева

**ROWID** – псевдостолбец, явл-щий уник.идентиф.строки.

\*уник не только в рамказ табл, но и в рамках БД.

\*упрощ.работу с БД, т.к.позвол.1значно идентиф.любоую строку таблицы, что позвол.удалять/измен. строки табл бзе первичного ключа.

\*поиск по rowed – самый быстрый

!нельзя примен.при разраб.прилож, расчит.на работу с БД других типов.

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

!нельзя напрямую исп-ть в запросе (вернет ошибку)

\*позвол.вводить ограничение на кол-во выводимых записей

select \* from students where rownum < 10;

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

**NLS – National Language Support**

\* м. хранить д-е мн-ва нац. языков, используя Unicode или спец. кодировки (character set)

\* символы хранятся как коды символов, завис. от выбранного набора

\* в одной БД мб 2 набора: основной и доп.

\* устан при созд БД

\* измен: alter database (national) character set

\* кроме символов алфавита в набор вх. знаки преп, числ, $...

осн набор символов для:

- хран char, varchar2, clob, long

- опис имен объектов, пер-ных

- ввод и хран pl/sql модулей

доп набор символов для:

- хран nchar, nvarchar2, nlob

пер-ная окружения **NLS\_LANG = lang\_territory.charset**

lang – имена месяцев, дней, направл текста : по умолч american

territory – настр календаря, формат даты, денег

charset – отобр символов, заглавных букв,

словари: nls\_ [session | instance | database] \_parameters

**Семантика символов:**

\* байтовая – рассм строки как посл-сть байтов (по умолч)

\* символьная – рассм строки как посл-сть символов

задается nls\_length\_semantics

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

Тип переменной основан на известн.стр-ре д-х.

Скал.ссылка **%TYPE** для опред.пер-ной на осн.другой пер-ной / поля в табл.

Ссылка на запись **%ROWTYPE** для опред-ния стр-ры записи на осн.табл/курсора

1. Локальные процедуры и функции языка PL/SQL.

**Лок.прогр. модуль** – процедура/ф-ция, опред.в секции декларации pl/sql блока

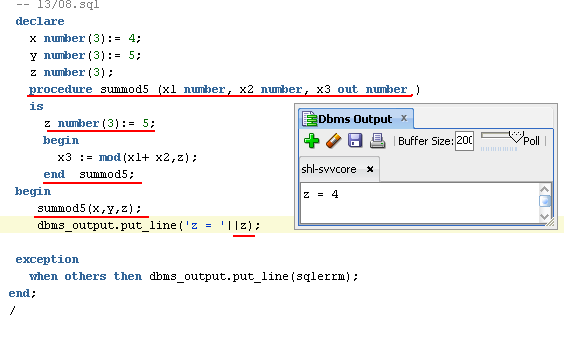
Лок.процедуры и ф-ции:

\*их объявление дб в конце секции декларир.после всех типов, записей, курсоров, пер-ных и исключений

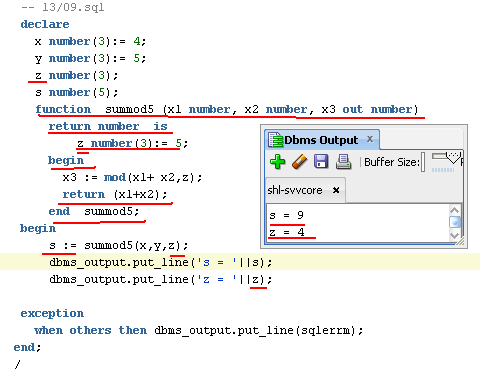
\*мб исп-ны только в рамках блока, в кот.они объявл

\*мб перегружены (разн.число пар, др.тип прогр.модуля, др.семейство пар-ров)

**Лок.процедура:**



**Лок.функция:**



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

**Запись** – стр-ра д-х, составл.из полей.

**Типы записей**:

\*табличные

\*курсорные

\*программно-определенные

**Объявление записей:**

\*на осн.таблицы (%rowtype)

declare

one\_book books%rowtype;

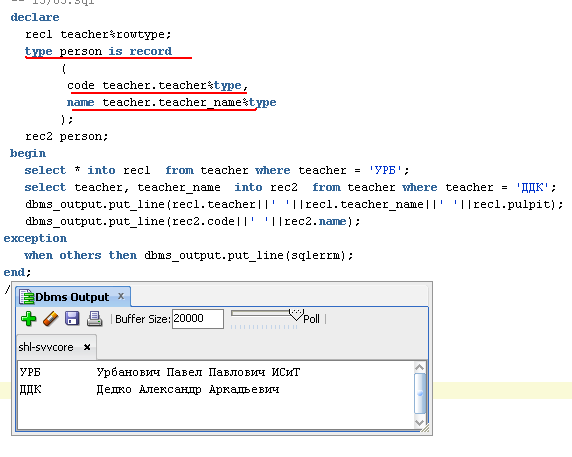
\*на осн.курсора (cursor + %rowtype)

declate cursor *myc* is

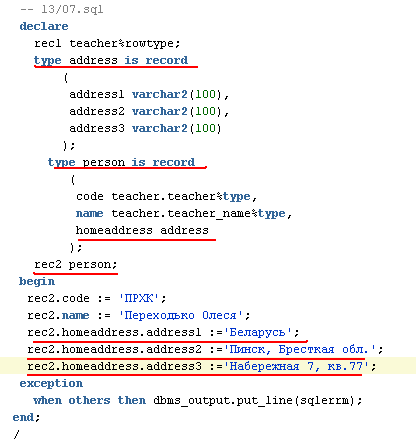
select \* from books where author like ‘%Fadeev%’;

one\_curs mycc%rowtype;

\*запись опр-мая прогером:



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



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

**Оператор IF**

1. if … then … end if;
2. if … then … else … end if;
3. if … then … elseif …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 1 выбир.знач и сравниваем его с чем-то. Во 2 – проверяем условия.

**Циклы loop, for, while**

|  |  |
| --- | --- |
| 19 | \*for – нельзя измен.знач.в цикле  \*loop – обяз.указ.когда вых. из цикла  \*while – вып. пока вып.усл.цикла  **Выход из цикла**:  \*exit – безусл.выход  \*exit when – выход при вып.усл.  \*goto – выход во внешн.контекст |

1. Курсоры. Виды курсоров. Схемы обработки курсора. Атрибуты курсора. Курсоры с параметрами. Динамические курсоры.

**Курсор** – объект БД, позв работать с записями построчно

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

**Виды:**

- явный (объяв разработчиком), неявный (нет объяв, open fetch close)

- стат (выражение опр при компил), динам (выраж опр при выполнении)

\* м. возвр 1/0/неск строк

\* для повторного созд. рез набора для др. значений параметров, курсор надо закрыть и открыть

**Операторы управл курсором:**

- declare – объяв явного

- open – откр курсор, создавая новый рез набор на базе указ. запроса

- fetch – послед извлеч строк из рез набора от начала до конца

- close – закр курсор и освоб его 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*!

**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;

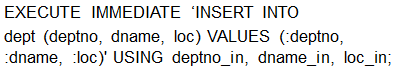
**Динамические курсоры.**

- execute immediate – одностроч запросы и ddl-команды

- open for, fetch, close – динам многострочные запросы

\* для улучш произв-сти вып sql-выражений м. исп динам. курсоры со связанными пер-ными

\* позв повторно использ разобранные SQL-выражения из разделяем. пула

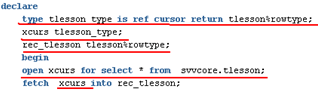


1. Курсоры. Курсорные переменные. Курсорные подзапросы. Использование конструкции CURRENT OF в курсорах.

**Курсорные пер-ные** – структуры д-х, указ. на курсорный объект

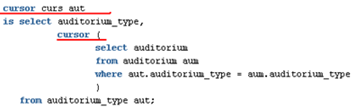
- для передачи курсора в кач параметра

- чтобы отложить связь курсора с SELECT-запросом до вып OPEN



Если курс. перем объяв с пом. REF CURSOR без **RETURN** – она мб связана с любым запросом, иначе – только с запросом, возвр результат точно соотв. числу и ТД в записи после фразы return

**Курсорные подзапросы**



**Использовании конструкции Current of в курсорах**

Если планир обнов/уд записи, на кот. ссылается SELECT FOR UPDATE:

update табл set … where current of имя курсора

delete from табл where current of имя курсора

это позв обн/уд запись, кот. была в курсоре последней

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

**Искл ситуация** – событие, возник в проге и треб. незамедлительной обработки

1. программно-определяемые
2. предопредел (стандартные)

\* sqlerrm – функция, возвр сообщ об ошибке

\* sqlcode – функция, возвр № ошибки

- ошибка, сгенер сервером / приложением / в рез. д-вий юзера

*Обработка искл* – перехват ошибки в секции искл

*Примеры сстемных исключений plsql:*

**NO\_DATA\_FOUND**

**TOO\_MANY\_ROWS**

**DUP\_VAL\_ON\_INDEX**

**INVALID\_NUMBER**

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

*Распространение искл* – процесс передачи искл от 1 блока другому, если искл не было обработано

**raise\_appl\_error** – команда, перехват. выполнение тек. блока

- определена в пакете dbms\_standart

- м. присвоить сообщ об ошибке

*При вып процедуры:*

- вып блока прерыв

- измен в аргументах in out / out откат

- измен в глоб структ не откат – надо явно rollback

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

**Встроенные функции:**

\*числовые функции (round, cos/sin, mod, sqrt, power ……)

\*символьные функции (ascii, asciistr, unistr, concat, initcap (1 заглавн), instr)

\*по работе с датами (current\_date, sysdate, dbtimezone, localtimestamp…)

\*конвертирование (convert, to\_number, to\_date, to\_char)

\*обработки ошибок (sqlerrm (сообщ.об ошибке), sqlcode (номер ошибки))

Для строк:

\*rpad (str1, x, str2) – возвращ.стр1 дополненную справа строкой 2 до размера х

\*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

1. Коллекции. Массивы переменной длины. Вложенные таблицы. Ассоциативные массивы.

**Коллекция** – стр-ра д-х, кот.сод.эл-ты одного типа.

\*эл-т коллекции – как скал.вел, так и композитные д-е.

\*эл-ты колл.м.сравнивать между собой на эквив.

\*можно перед.пар-ром

\*одномерная, но м.созд.коллекции коллекций

Коллекция **сост** из набора эл-тов, при чем каждый эл-т нах.в опр.поз (им.индекс эл)

Необх.объявить **тип колл** – командой type

Необх.**объявить колл –** пер-ную этого типа для дальнейшего исп-ния.

**Огранич**. Колл. - если заранее опред.границы возм.знач.индексов ее эл-тов,

иначе – **неогранич.**

Колл.типа varray всегда огранич.

Влож таблицы и ассоциативн.массивы – неогранич.

**Плотная колл** – все ее эл-ты опред.и каждому из них присв.нек.знач (в т.ч.null)

\*Массивы VARRAY всегда являются плотными

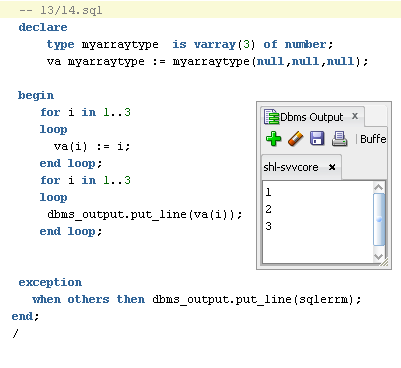
\*Влож.табл.первонач.всегда плотные, но по мере удал.нек.эл-тов -разреж

\*Ассоц.масс.мбкак разреж, так и плотными в завис.от способа их зап-ния

**Массивы пер-ной длины** – 1мерные, связанные коллекции 1типных эл-тов.

\*доступны в рамках pl/sql и в БД;

\*явл.плотными



**Вложенные таблицы**

**Влож.табл** – 1мерные, несвяз.коллекции 1типных эл-тов;

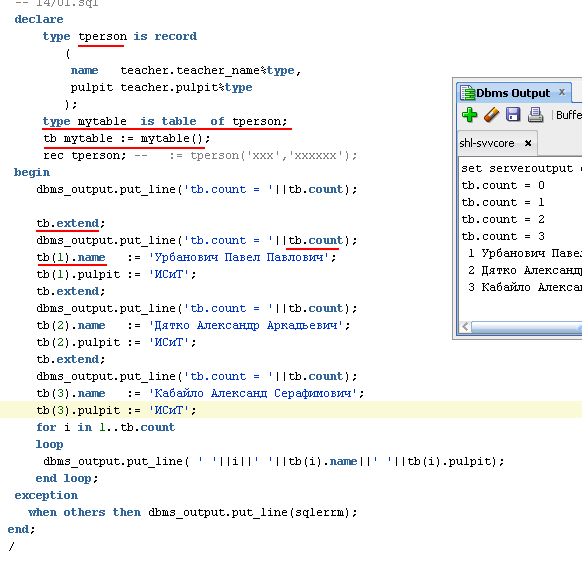
\*доступны в рамках pl/sql и как поля таблицы в БД

\*изнач.плотные, но мб стан.разреж.

\*объявление:

type *collection\_name* is table of *datatype;*

\*datatype – любой допуст.тип pl/sql за исключ ger cursor;

****

**Ассоциативные массивы –** 1мерные, неогранич (по макс.кол-ву эл-тов при созд) колл эл-тов

\*доступны только в рамках pl/sql

\*изнач.разреж, индекс м.принима.непоследоват.знач.

\*пободны хеш-таблицам

\*представл.собой мн-во пар знач, одно сод.в масс, в другое служит индексом

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

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

*Процедура* – именованный модуль, вып. 1/неск выраж и м. принимать / возвр значения через список парам

**Типы парам:**

\* in

\* out – при выполнении устан в null

\* in out – при выполн ост. неизменными

**Значение по умолч:**

in, in out – можно не задавать при вызове

**Передача парам:**

\* позиц – каждое значение в списке аргум вызова ставится в соотв формальному парам по порядку : *nn (23, name, surname)*

\* именован – явно связ аргументы при вызове с парам по именам :

*nn (in\_id => 23, out\_name => name)*

\* можно комбин оба метода, пока позиционные стоят слева

**Вызов процедуры:**

conn -> procedures -> выбрать -> Run

exec

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

*Функция* – именованный модуль, вып. 0+ выражений через фразу Return

**Вызов:**

- в присвоении нач. значения пер-ной

- в выражении присвоение

- в булевом выраж

- в SQL запросе

- как аргум в списке парам др. функции / процедуры

**Deterministic** – функция возвр. одно и то же значение при вызове с теми же парам

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

*Пакеты* – коллекция pl/sql объектов, сгруппир вместе

\* скрытие инфы

\* объектно-ориент дизайн

\* постоянство объектов в транзакциях

\* улучшенная произв-сть

***Спецификация пакета (package)*** – обяз, содерж список объектов для общего доступа из др. модулей или приложения

***Реализация (package body)*** – сод. весь прогр код для реализации процедур и функций из спецификации, приват. объекты и секцию иниц

1. 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. срабат по запускающему их событию

**Виды триггеров:**

\* *по привяз. объекту:* на табл, на предст

\* *по событиям запуска:* insert, update, delete

\* *по обл д-вия:* уровень оператора, ур. записи, составные

\* *по t сраб:* 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 к. строки BEFORE

\* вып. оператор

\* для к. строки AFTER

\* операторный AFTER

**Предикаты триггеров:**

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 д/табл : 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 |

1. Секционирование таблиц. Виды секционирования.

**Секционирование** – метод, кот.позвол.хранить сегмент д-х в виде неск.сегментов.

\*повыш.произ-сть обраб.д-х

\*упрощ.упр-ние крупными объектами д-х

\*обесп.доп.надежность системы

**Диапазонное (range) –** секционир., при кот.для кажд.секции опред.диап.знач. ключа секционирования.

\*ключ м.приним.знач даты и времени, числа, текста

! при загрузке новых д-х, необх.посто.расщеплять секцию maxvalue;

\*для задания диап.исп.ключ.слово *less than*

Пример:

create table T\_RANGE( docnum number) partition by range(docnum)

(partition n1 values less than(10) tablespace t1,

partition n2 values less than(20) tablespace t2,

partition n3 values less than(30) tablespace t3);

**Интервальное –** при загрузке новых д-х в табл.авто созд.новые секции для нового диап.ключей.

\*новые секции будут созд.авто (по 1ой оп-ции insert, не попад.в диап.секций)

create table T\_INTERVAL( t\_date date) partition by range(t\_date)

interval(NUMTOYMINTERVAL(3, 'MONTH'))

(partition d1 values less than (to\_date('01.01.2016', 'DD/MM/YYYY')),

partition d2 values less than (to\_date('01.03.2016', 'DD/MM/YYYY')));

**Хэш-секц** позвол.равномерн.распред.стоки между секциями, т.е.разбросать строки по разным секциям и сделать эти секции равновеликими.

!не означ, что строки распред.по секцияям случайным образом

create table T\_HASH( doctype varchar2(5))

partition by hash(doctype)

(partition p1 tablespace th1,

partition p2 tablespace th2)

**Списочное** – позвол.разбить таблицу по списку конкр.знач.

\*ключ спис.секц.мб только 1столбцовым

\*default – опис.все знач, не попавшие в другие списки

\*null – мб ключ.знач.

create table T\_LIST (l1 char(10)) partition by list(l1)

(partition ch1 values('a', 'b') tablespace tl1,

partition ch2 values('c') tablespace tl2);

**Композитное -** слишком большие таблицы – секционирование по 2 ур. (секции и подсекции)

1. Транзакции в СУБД Oracle. Виды транзакций. Понятие автономной транзакции.

**Транзакция** – неск. послед. инструкций SQL, кот. рассм как единое целое

**Операторы:**

\* **COMMIT** – фиксир все измен для тек. транз

\* **ROLLBACK** – откат незафикс. изменений: читает инфу из сегментов отката и восст блоки д-х в сост, в кот. они находились до нач. транз, освобожд блокировки

**Автономные транз** – позв созд *подтранзакции*, кот. м сохр/отмен изменения вне завис. от родительской транз

1. 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

SUBMIT – создание задания

ISUBMIT – создание задания с номером

REMOVE – удаление задания из очереди

RUN – немедленное выполнение задания в пользовательском сеансе

BROKEN – разрушение задания (16)

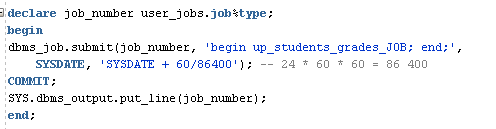
INSTANCE – указание экземпляра

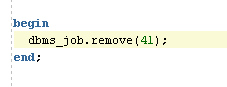
NEXT\_DATE – изменение времени выполнения

INTERVAL – изменение интервала выполнения

CHANGE – изменение параметров задания

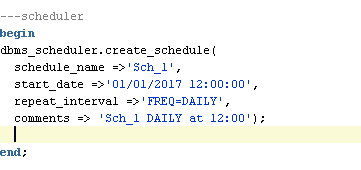
WHAT – изменение задания

****

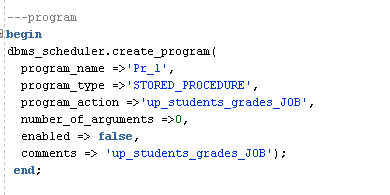
****

**DBMS\_SCHEDULER**

Schedule – расписание



Program – программа



Job – плановая программа

