SQL

*PL/SQL, SQL*Plus*

Справочник

Свободное интернет-издание

WWW.DELTACOM.CO.IL

27.01.2001

Справочник
SQL, PL/SQL , SQL*Plus

Эта книга предназначена для всех кто интересуется базами данных.

В книге описан язык запросов к базам данных SQL и его процедурное расширение PL/SQL

По мере сил и возможностей, на основании многолетнего опыта практического применением базы данных Oracle , автор постарался собрать справочный материал для практического применения при разработке клиент-серверных систем.

Для тех , кто не знаком с данным понятием можно сказать, что в настоящее время ни одну из промышленных систем обработки информации нельзя представить без хранения информации, а сервер базы данных предназначен для обработки поступающих запросов от различных программ на получение информации..

Знание языка запросов к базам данных актуально, поскольку развитие вычислительной техники привело к практически повсеместному распространению систем распределённой обработки и хранения информации.

Здесь учтены команды Oracle 8 и Oracle-7 и Oracle-6. Пособие может быть использовано как при работе на Personal Oracle-7, 8 , Oracle-7,8 Server, так и при работе на VAX station.

Автор не претендует на полноту охвата всех возможностей Oracle server, поскольку это не мыслимо в книге такого маленького объёма.

Терминологический словарь

Раздел содержит описание терминов используемых при работе с базой данных. При описании какой-либо темы неизбежно применение специальных терминов.

ANSI - American National Standard Institute (Американский Институт Стандартизации)

Центральный институт стандартизации США . Стандарт, принятый данным институтом, является рекомендуемым для других стран и обязательным для применения на территории США.

Изделия, не удовлетворяющие условиям стандарта не могут быть использованы на территории США

privileges access (Привилегии доступа)

Возможности выполнения определённых действий, устанавливаемые при соединении с базой данных. (Привилегии Доступа предоставляются через счета пользователя.).

account (Счет)

Счет - специально организованные данные, предназначенные для обеспечения управления возможностями пользователя . Счета обычно создаются и управляются системным администратором. Счет состоит из имени пользователя и пароля.

active database connection (Активное соединение с базой данных)

Текущее соединение с базой данных. .-Это соединение, которое установлено при издании директивы CONNECT. Текущее соединение определяет, какие данные пользователь может просмотреть и изменить и его возможности по обработке информации в базе данных.

advanced distribution features (Расширенные распределительные особенности)

Дополнительные возможности программного обеспечения, . используемые в компьютерных сетях В ORACLE-7это:

- Связи Базы данных,
- Распределенные Запросы,
- Распределенные Модификации,
- Кадры только для чтения
- SQL*Net- сетевая поддержка распределённых систем.
- SQL*Net 2 - дополнительное изменение сетевой поддержки, используемого Oracle Developer-2000

alias (Псевдоним)

Альтернативное имя для существующего сетевого объекта, типа главной станции или набора параметров.

В SQL - временное имя, назначенное для таблицы, представления, столбца, или значения внутри утверждения SQL.

Алиас используется, чтобы обеспечить ссылку на другой объект через новое имя.

Application Programming Interface (API) (Интерфейс Программирования Прикладной программы)

Набор функций, используемых для обеспечения возможности подготовки приложения на языках программирования. Используется программистами.

ASCII data file (ASCII файл данных)

Простой текстовый файл. ASCII - Американский Стандартный Код Обмена Информацией.

backup (Копия)

Процесс создание копии данных. Используется чтобы обеспечить возможность восстановить данные при аварийных ситуациях.

При проведении операций копирования проводится автоматическая архивизация (сжатие) информации для более компактного хранения информации.

client (Пользователь)

Пользователь, прикладная программа или компьютер, который запрашивает данные или обработку информации у другой программы или компьютера.

Задача клиентной программы - оформление запроса пользователя, визуализация информации и

посылка запроса на сервер.

cluster (Кластер)

Структура базы данных, которая содержит один или несколько таблиц.

Таблицы физически сохранены вместе внутри базы данных для более быстрого доступа к ним.

column (Столбец)

Вертикальное пространство в таблице. Столбец имеет имя и специфический тип данных ..

column name (Имя столбца)

Алфавитно-цифровой указатель столбца. Имя столбца - 30 символов без пробелов и первый символ обязательно алфавитный символ.

Национальные символы в большинстве систем не допустимы. Даже если это допустимо то при использовании национальных символов в качестве имён полей возникают проблеммы переноса данных па различные платформы (Пример – Microsoft Access). В Oracle это категорически запрещено.

communications protocol (Протокол связи)

Набор стандартных правил для управления передачей данных в компьютерной сети.

complex query (Сложный запрос)

Также называемый сложным утверждением. Утверждение INSERT, UPDATE, DELETE, или SELECT, содержащие подзапрос.

connect (Соединить)

- Возможность доступа к базе данных .
- Директива, с помощью которой осуществляется подключение к базе данных
- Процесс установки взаимосвязи с сервером.

connect descriptor (Соединитель - описатель)

Специально форматируемое описание сетевого соединения.

Указывает характеристики сети и метод доступа к сети на низком уровне доступа

connect string (Соединитель - строка)

Набор параметров, включая протокол, (SQL*NET) которые используются, чтобы присоединить сервер специфическим образцом.

database (База данных)

Совокупность данных, которая обрабатывается как модуль.

Состоит из логических и физических структур. Разрабатываются, чтобы сохранять и восстановить связанную информацию.

database administrator (Администратор базы данных)

Человек, ответственный за проводимые операции сопровождения и настройки базы данных.

database application (Прикладная программа базы данных)

Набор форм, меню, отчетов, клиентных программ и других компонентов, которые обеспечивают ввод, обработку и получение информации.

database connection (Соединение базы данных)

Доступ к базе данных , выполненный при подсоединении к ней.

database link (Связь базы данных)

Связь базы данных - объект в базе данных, который позволяет Вам обращаться к объектам на удаленной базе данных.

database object (Объект базы данных)

Объекты баз данных состоят из таблиц, представлений, индексов, синонимов, связей базы данных, ролей, кадров, и пользователей.

Каждый из этих объектов определяет некоторые характеристики проекта базы данных.

database owner (Владелец базы данных)

Обычно человек, создавший базу данных, - тот кто может предоставлять привилегии доступа к базе данных другим пользователям.

database role (Роль базы данных)

Роль базы данных - метод предоставления привилегий доступа.

Вы создаете объект базы данных называемый ролью, затем предоставляете привилегии созданной роли, затем предоставляете эту роль индивидуальным пользователям. (В роли указывается совокупность различных прав работы с объектами базы данных

database server (Сервер базы данных)

Центральная программа, принимающая запросы других программ на выполнение некоторых действий и отвечающая на них по имеющейся информации.

datatype (Тип данных)

Вид хранимой информации в сервере и вид переменных в PL/SQL процедурах .

В типе данных указывается какую информацию можно сохранить в поле базы данных

distributed database (Распределенная база данных)

Одиночная, логическая база данных, которая физически размещена на двух или больше компьютерах, объединенных через некоторую систему коммуникаций.

Существенная особенность распределенной базы данных - то, что пользователь и/или программа работает так, как будто он имеет доступ к целой базе данных в конкретном месте.

Распределённая база данных всегда состоит из тесно взаимосвязанных различных серверов .

distributed option (Распределенная опция)

Разрешает Вам обращаться к данным, постоянно находящимся на различных компьютерах в различных местах сети. Распределенная опция включает:

- Связи Базы данных
- Распределенные Запросы
- Распределенные Модификации
- Кадры только для чтения,.

distributed query (Распределенный запрос)

Запрос, который выбирает данные из баз данных на различных серверах базы данных сети. Распределенный запрос может ссылаться на данные в других серверах сети через текущий локальный узел.

При соединении используются объединения, вложенные запросы, или представления.

distributed update (Распределенная модификация)

Модификация, которая изменяет данные в локальной или удаленной базе данных в конкретной текущей транзакции.

domain name (Имя области)

Идентификатор области имен, который гарантирует, что имя уникальное. Области связаны иерархией вызовов.

DOMAIN структура имен распространяется как на сетевую организации , так и на внутренние имена в сервере и имена глобальных областей.

DNS Domain Name System (Система наименования доменов)

Специальная база данных, позволяющая осуществлять преобразование имён в визические характеристики.

Export Utility (Экспортная Утилита)

Вспомогательные программы , предназначенные для записи информации во внешние файлы операционной системы.

Файлы могут быть сохранены и переданы на другой сервер с целью их последующей загрузки в базу данных.

foreign key (Внешний ключ)

Один или большее количество столбцов таблицы, которые используются для обращения в другую таблицу.

foreign data source (Внешний источник данных)

Базы другого типа и прочии источники информации, используемые для загрузки базы данных.

host (Главный компьютер)

Компьютер, который обеспечивает общедоступнный ресурс на сети.

host string (Главная строка)

Ссылка на особенно форматируемое описание адресата сетевого соединения.

Import Utility (Утилита Импорта)

Вспомогательные программы Oracle для загрузки информации в базу данных из внешних файлов операционной системы.

index (Индекс)

Объект базы данных, созданный, чтобы увеличить эффективность поиска данных.

Индекс обеспечивает быстрый доступ к базе данных. Oracle автоматически использует индексы для уменьшения времени поиска.

INIT.ORA

Системный файл базы данных Oracle . Содержащий список параметров, которые устанавливают параметры загрузки базы данных. Файл текстовый. Может быть отредактирован в ручную.

instance (Отдельный случай, требование)

Требование старта системы. Для Oracle7 - база данных старта системы.

Personal Oracle7 поддерживает один образец базы данных начального запуска. В других случаях под instance понимается конкретная реализация объектного класса.

ISO/IES - International National Standard Institute / International Electro-technical Commission

Международный институт стандартизации / международная электро -техническая комиссия. Стандарты, принятые международным институтом стандартизации носят рекомендательный характер.

local database (Локальная база данных)

База данных, установленная на вашем компьютере. Это запущенная база данных при старте или вновь созданная база данных.

logon (Регистрация)

Обращение к базе данных, использующей назначенное имя пользователя и пароль.

Процесс установления идентичности пользователя по имени и паролю.

master table (Главная таблица)

Таблица, которая используется как основная при создании кадра в запросе.

migration (Перемещение)

Дополнительные действия, осуществляемые при установке Oracle7, обеспечивающие изменение версии системы, исправление ошибок и введение дополнительных возможностей .

network connection (Сетевое соединение)

Электронное соединение, обеспечивающее возможность обращения к ресурсам других компьютеров из локального компьютера.

Open Database Connectivity (ODBC) открытое соединение с базами данных

Специальный драйвер, предназначенный для обеспечения соединения клиентных приложений с базами данных, размещенных как на локальном компьютере, так и в распределённой вычислительной среде. Применяется в Windows. Внутрь ODBC загружаются драйвера для различных баз данных.

object (Объект)

Объединение данных, алгоритмов их обработки (методов обработки) и событий в единый блок, имеющий собственное имя.

В Oracle - это именованная структура в базе данных типа таблицы, индекса, формы или любого другого компонента, имеющее собственное имя (за исключением синонима).

Вы можете копировать, перемещать, или удалять в объект по его имени .

Oracle Call Interface (Интерфейс Обращения к Oracle)

(OCI) - интерфейс программирования прикладных программ (API)

Oracle Developer/2000 (Разработкик/2000)

Инструмент разработки, который дает возможность организациям формировать системы от рабочих групп до уровня предприятия.

Oracle Named Pipes Protocol Adapter (Oracle именованный Адаптер Протокола Канального доступа)

Система взаимодействия между серверами по имени компьютера в доменной структуре сети.

Oracle SPX Protocol Adapter (Oracle SPX Адаптер Протокола)

SQL*NET интерфейс прикладной программы , который позволяет SQL * NET связываться с удаленной базой данных через SPX протокол.

Oracle TCP/IP Protocol Adapter (Oracle TCP/IP Адаптер Протокола)

SQL * интерфейс прикладной программы Сети, который позволяет SQL*Net связываться с удаленной базой данных через TCP/IP Протокол.

Oracle7 DBMS (Oracle7 система управления базой данных)

Система управления базой данных, разработанная Корпорацией Oracle.

Компоненты системы управления базой данных включают ядро системы и различные вспомогательные утилиты для использования администраторами базы данных и пользователями базы данных.

package (Пакет)

Пакет - скрытая совокупность связанных процедур, функций, и других объектов программы, сохраненных вместе в базе данных.

parallel database (Параллельная база данных)

База данных, в которой, одновременно с основной, осуществляется сохранение информации всё время работы системы.

Создание параллельной базы данных осуществляется с целью обеспечения повышенной надёжности сохранения информации.

Параллельную базу данных и основную базу данных необходимо создавать на физически различных носителях информации.

При разрушении основной базы данных , параллельная база данных может бать запущена в тот -же момент.

password (Пароль)

Пароль пользователя -. секретное слово или фраза (30 символов; пробелы и запятые запрещены), связанная с именем пользователя.

Пароль используется для защиты данных и должен быть известен только владельцу.

Personal Oracle7 Navigator (Персональный Oracle7 Навигатор)

Программа, обеспечиваюшая экранное взаимодействие администратора базы данных с сервером базы данных.

Обеспечивает упрощение работы пользователя по созданию главных объектов приложения.

PL/SQL database language (PL/SQL язык баз данных)

Процедурное расширение языка SQL. PL/SQL объединяет легкость и гибкость SQL с процедурными функциональными возможностями языка структурного программирования

primary key (Первичный ключ)

Набор столбцов, используемых, чтобы установить уникальность строк.

Комбинация значений столбца уникальна для каждой строки в таблице.

Первичный ключ - наиболее часто используемые средства вызова к строкам. Так - же применяется чтобы отсортировать таблицу и защитить данные от дублированых значений.

privilege (Привилегия)

Право выполнять специфический тип утверждения SQL. Имеются два типа привилегий:

Привилегии Системы выполнять специальное действие на конкретном типе объекта; например, привилегия удаления строки любой таблицы.

Объектные привилегии позволяют пользователю выполнять действие на конкретном объекте; например, привилегия удаления строки в специфической таблице.

private synonym (Частный синоним)

Синоним для объекта базы, установленный с целью обеспечения возможности обращения только конкретных пользователей к конкретному объекту базы данных.

procedure (Процедура)

Утверждения SQL и PL/SQL, которые сгруппированы вместе как модуль (Связанная совокупность действий над данными).

Предназначены для обеспечения решения конкретной задачи обработки данных.

Процедура выполняется на сервере и результаты её выполнения передаются клиентной программе.

С помощью процедур уменьшается поток передачи информации от сервера клиенту.

project (Проект)

Связанная совокупность хранимой информации и методов её обработки .

Проект используется с целью обеспечения совместного хранения всей информации о разработанной системе.

public synonym (Общий синоним)

Синоним для объекта базы данных, определенного администратором базы данных. Определяется для того, чтобы все пользователи базы данных могли иметь доступ к конкретному объекту.

publish-and-subscribe replication (Изданный-и-Присоединённый ответ) (кадр только для чтения)

Возможность, которая гарантирует, что никакие действия других пользователей не повлияют на информацию, находящуюся в процессе обработки.

Отмена кадра только на чтения осуществляется изданием директивы COMMIT

raw (Сырое)

Специфический тип данных, используемый в базе данных, указывающий что в данной колонке могут быть любые данные, о которых база не делает никаких предположений о типе хранимой информации

read-only snapshot (Кадр только для чтения)

Точная копия главной таблицы.

Возможно создавать неограниченное число кадров только для чтения .

Только при обновлении в кадр только для чтения попадает новая информация.

relational database (Реляционная база данных)

База данных, составленная из таблиц взаимосвязанной информации.

Данные могут быть реорганизованы и представляться различными способами. Основные принципы реляционных баз данных -

- Простая структура хранения информации -таблицы
- Сложная , развитая система обработки информации

remote database (Удаленная база данных)

База данных, являющаяся доступной через сетевое соединение..

replica (Точная копия)

Копия главной таблицы. В реплике не отображаются данные , которые изменились после создания реплики.

Реплика используется с целью обеспечения целостности данных при работе различных пользователей над одними и теми же данными.

replication (Репродукция)

Специфическое действие базы данных, которое гарантирует, что модификация данных любым пользователем будет произведена без разрушения информации.

replication master site (копия владеющая местом)

Установленные главные копии информации, подлежащей модификации.

Репликационное главное место требуются для кадров, в которых производится изменение информации.

role (Роль)

Поименованная группа связанных привилегий. Роль используется для установки полномочий пользоватею или других ролей.

row (Строка)

Одиночная запись в базе данных. Строка таблицы.

select (Выбор)

Выбор строки данных из одной или нескольких таблиц .

sequence (Последовательность)

Создаёт последовательный список уникальных чисел. Не гарантируется непрерывность.

server (Станция)

Предназначен для распределённой обработки информации. Клиент посылает запрос на сервер, а сервер производит выполнение работы в соответствии с запросом и передаёт ответ клиенту.

Сервер обслуживает совокупность клиентов. При этом в качестве клиента может выступать другой сервер.

Серверы и клиенты могут работать на различных компьютерах сети. В многозадачных операционных системах сервер и клиент может находится на одном и том-же компьютере.

Такое деление обработки информации и называется архитектурой Client/Server.

Архитектура клиент-сервер не означает использование в качестве сервера - сервера SQL.

Сервер может размещаться как на выделенном компьютере так и на рабочей станции.

Сервер может обеспечивать связь с другими серверами для обеспечения перенаправления запроса на выборку информации. Подобная обработка называется распределённой обработкой информацией.

service name (Сервисное имя)

Имя, используемое, чтобы идентифицировать SQL*NET процесс Файл описания SQL*Net содержит соответствие по сервисному имени всех параметров сетевой коммуникации.

shut down (закрытие)

Процесс остановки текущей базы данных.

snapshot (Кадр, быстрый выстрел)

Локальная копия удаленной главной таблицы. Кадры могут также быть updatable (доступные для изменения) или доступные только для чтения. Кадр только для чтения может регенерироваться периодически.

snapshot refresh (Регенерация кадра)

Процесс отражения новой информации в кадре. Информация в кадре зафиксирована.

Любые изменения производимые другими пользователями в текущем кадре не отображаются .

Для отображения новой информации и предназначена регенерация кадра.

SQL (Structured Query Language) (Структурированный Язык Запросов))

Язык манипулирований для реляционных баз данных.

Всемирно принятый стандарт для реляционных систем.

ANSI X3.135-1992 “Database Langiage SQL”

ISO/IEC 9075:1992 “Database Langiage SQL”

Язык SQL является непроцедурным языком системы управления реляционной базой данных, который может обрабатывать записи группами и обеспечивает автоматический поиск нужных данных.

Непроцедурный язык - язык у которого отсуствует возможность управления последовательностью действий.

Язык SQL дает возможность указывать, какие данные необходимо получить или изменить, не заботясь о том, как найти эти данные .

Язык PL/SQL, в свою очередь, является процедурным языком, способным управлять ходом вычислений.

В языке PL/SQL есть условные операторы, циклы, средства для работы с курсорами ORACLE и обработки особых ситуаций.

SQL*DBA Utility (SQL * DBA Вспомогательные программы (Утилита))

Утилиты позволяют управлять Oracle7 Server в упрощённом для DBA режиме

SQL*Loader Utility (SQL * Утилиты Загрузчика)

Применяются для загрузки данных, обычно ASCII файлы данных из файлов операционной системы в

таблицы базы данных .

SQL*Net (SQL * Сеть)

Вспомогательные программы Oracle предназначенные для организации обмена данных через сеть. SQL*Net поддерживает распределенную обработку и хранение информации через различные виды серверов.

SQL*Plus (SQL * Плюс)

Основанный на SQL язык, обеспечивающий конфигурирование системы и работу с сервером в командном режиме.

Иногда используется как инструмент для работы конечного пользователя.

При работе администратора используется как средство автоматизации его работы.

starter database (База данных стартеров)

Заданная по умолчанию база данных, запускаемая автоматически при загрузке сервера

stored function (Сохраненная функция)

Набор утверждений PL/SQL сохранённых на сервере под единым именем. Вызов процедуры осуществляется по её имени

SPX/IPX - Sequenced Packed Exchange / Internet Packet Exchange

Упорядоченный Обмен Пакетами / Межсетевой Обмен Пакетами - протокол передачи информации локальных одноранговых сетях и сетях с выделенным сервером.

symmetric replication (симметричные изменения)

Поддержка многократных копий данных в различных местах распределенной системы.

synonym (Синоним)

Псевдоним для таблицы, представления , последовательности, или модуля программы.

Используется для ссылки на объект по другому имени . Применяется так же для назначения прав доступа к объекту.

system change number (SCN) (Системный номер выбора)

Приоритетность выполнения транзакции. Данная опция позволяет назначать порядок выполнения транзакции в распределённых системах.

По умолчанию все транзакции имеют одинаковый уровень. Однако при издании директив, можно устанавливать приоритет команды с помощью ключа FORCE

table (Таблица)

Базисный модуль хранения информации в реляционных системах управления базой данных. Таблица состоит из одного или нескольких информационных модулей (строк), каждая из которых содержит одинаковые значения (столбцы).

table security (Защита таблицы)

Способность ограничивать доступ к информации через предопределенный набор строк и столбцов таблицы.

text editor (Текстовый редактор)

Программа, выполненная операционной системой компьютера, используемой, чтобы создавать и редактировать текстовые файлы.

trigger (Вызовите)

Сохраненная процедура, которая автоматически выполняется при изменении данных в таблице (утверждения INSERT, UPDATE, или DELETE)

Transparent Network Substrate (TNS) -(прозрачная сеть обмена)

Программное обеспечение, устанавливающее технологию одиночной работы в сетях промышленного стандарта. Под одиночной работой понимается система прямого взаимодействия 2-х программ, установленных на различных узлах сети.

Ttansaction (Дело, Сделка)

Совокупность действий с базой данных, которые логически необходимо рассматривать совместно с целью поддержки целостности базы данных.

TCP/IP - Transmission Control Protocol / Internet Protocol

Протокол Управления Передачи / Межсетевой Протокол - Установленный алгоритм взаимодействия компьютеров в сетях промышленного стандарта для распределённых сетей и сетей с Dial-Up соединением компьютеров.

two-phase commit (Утвердить с двумя фазами)

Метод, используемый распределенной системой управления базой данных для обеспечения гарантии проведения транзакции .

updatable snapshot (допустимый для изменения кадр)

Локальная копия удаленной главной таблицы для данных , которые могут изменяться

user (Пользователь)

Любой человек или группа людей с определёнными правами обращения к базе данных.

user account (Счет пользователя)

Информация о пользователе, включая: имя пользователя, пароль, и роль. Счет пользователя используется, чтобы предоставить ему привилегии доступа к объектам .

user name (Имя пользователя)

Имя, под которым пользователь известен серверу. Каждое имя пользователя связано с паролем и должно быть введено, чтобы соединиться с базой данных.

user password (Пароль пользователя)

Секретное слово или фраза (30 символов, пробелы и запятые запрещены) связанные с именем пользователя. Пароль используется для защиты данных и должен быть известен только владельцу.

view (Представление, вид))

Настроенное представление данных из одного или нескольких таблиц. Представление есть виртуальная таблица не существующая реально но с которой можно обращаться как с обычной таблицей. В представлении объединены средства выборки данных и обработки информации.

В некоторой литературе при описании представления он описывается как курсор,

Это не совсем верно, поскольку под курсором понимается логический вид представления в программном блоке PL/SQL.

System Identification (SID) (Системный указатель)

Имя, присвоенное серверу с целью обеспечения нахождения сервера в таблице связи задач. По указанному имени система определяет к какой именно задаче необходимо перенаправить запрос системы. В Windows-95/ Windows-NT SID указывется в регистре системы. Для редактирования регистра системы используется системная программа REGEDIT. Для ORACLE по умолчанию устанавливается имя ORCL

National Language Support (NLS),-(поддержка национальных языков)

Специальная функция, установленная в серверах для обеспечения хранения информации на национальных языках и обеспечения перекодирования информации при передаче информации на клиентные компьютеры, имеющие различные операционные системы.

Зарезервированные символы

Зарезервированные символы – это символы , которые применяются для отображения некоторых операций.

+	ОПЕРАТОР ДОБАВЛЕНИЯ
:=	оператор присвоения
=>	оператор ассоциации
%	Индикатор атрибута
‘	Разделитель текстовой строки
.	Разделитель компонентов
||	оператор Объединения строк
/	Операция деления
**	Оператор возведения в степень
(	Выражение , разделитель списка
)	Выражение , разделитель списка
:	Индикатор глобальной переменной
<<	Идентификатор метки (конец)
<<	Идентификатор метки (начало)
>>	Идентификатор метки
*/	многострочные комментарии(Конец)
/*	многострочные комментарии (Начало)
*	Оператор умножения
!=	Сравнение на не равно
^=	реляционный оператор
=	Сравнение на равно
<	Меньше
<=	Меньше или равно
<>	Не равно
>	Больше
>=	Больше или равно
~=	реляционный оператор
@	Запуск на выполнение файла .SQL
--	Комментарии на одной строке
;	Завершение команды SQL
-	оператор вычитания / отрицания

Зарезервированные слова

PL/SQL Зарезервированные слова	**Перевод на русский**
ABORT	АВАРИЙНОЕ ПРЕКРАЩЕНИЕ РАБОТЫ
ACCEPT	ВВОД
ACCESS	ДОСТУП
ADD	ДОБАВЛЕНИЕ
ALL	ВЕСЬ
ALTER	ИЗМЕНИТЬ
AND	И
ANY	ЛЮБОЙ
ARRAY	МАССИВ
ARRAYLEN	Длинна массива
AS	КАК
ASC	ASC (сортировать по возростанию)
ASSERT	УТВЕРДИТЕ
ASSIGN	НАЗНАЧЬТЕ
AT	В
AUTHORIZATION	РАЗРЕШЕНИЕ

AVG	В СРЕДНЕМ
BASE_TABLE	Базовая таблица
BEGIN	НАЧАЛО
BETWEEN	МЕЖДУ
BINARY_INTEGER	Двоичное целое
BODY	ТЕЛО
BOOLEAN	БУЛЕВО (Двоичное логическое)
BY	Для
CASE	СЛУЧАЙ
CHAR	Символ
CHAR_BASE	Базовый символ
CHECK	ПРОВЕРКА
CLOSE	БЛИЗКО
CLUSTER	КЛАСТЕР
COLUMNS	СТОЛБЦЫ
COMMIT	УТВЕРДИТЕ
COMPRESS	СЖАТИЕ
CONNECT	СОЕДИНЕНИЕ
CONSTANT	КОНСТАНТА
CRASH	АВАРИЙНЫЙ ОТКАЗ
CREATE	СОЗДАЙТЕ
CURRENT	ТЕКУЩИЙ (АКТУАЛЬНЫЙ)
CURRVAL CURSOR	Текущее значение Курсора
DATABASE	БАЗА ДАННЫХ
DATA_BASE	База данных
DATE	ДАТА
DBA	DataBaseAdministrator Администратор базы данных
DEBUG OFF	DEBUGger OFF (Отключить отладчик)
DEBUG ON	DEBUGger ON (Отладчик включён)
DECLARE	ОБЪЯВИТЬ
DECIMAL	ДЕСЯТИЧНОЕ ЧИСЛО
DEFAULT	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ
DEFINITION	ОБЛАСТЬ ОПРЕДЕЛЕНИЯ
DELAY	ЗАДЕРЖКА
DELETE	УДАЛИТЬ
DELTA	Разность
DESC	DESC (сортировать по убыванию)
DIGITS	ЦИФРЫ
DISPOSE	РАСПОЛОЖИТЬ
DISTINCT	ОТЛИЧНЫЙ
DO	Делать
DROP	СНИЖЕНИЕ,Выбросить
ELSE	ЕЩЕ
ELSIF	ИНАЧЕ ЕСЛИ
END	КОНЕЦ
ENTRY	ВХОД
EXCEPTION	ИСКЛЮЧЕНИЕ
EXCEPTION_INIT	Инициализация исключения.
EXISTS	СУЩЕСТВУЕТ
EXIT	ВЫХОД
FALSE	НЕПРАВИЛЬНО
FETCH	ВЫБОРКА
FLOAT	С ПЛАВАЮЩЕЙ ТОЧКОЙ
FOR	ДЛЯ (Цикл)
FORM	ФОРМА
FROM	ИЗ
FUNCTION	ФУНКЦИЯ
GENERIC	РОДОВОЕ
GOTO	Перейти

GRANT	ПРЕДОСТАВЛЕНИЕ
GROUP	ГРУППА
HAVING	НАЛИЧИЕ
IDENTIFIED	ИДЕНТИФИЦИРУЯСЬ
IF	ЕСЛИ
IN	В
INDEX	ИНДЕКС
INDICATOR	ИНДИКАТОР
INSERT	ВСТАВКА
INTEGER	ЦЕЛОЕ ЧИСЛО
INTERSECT	ПЕРЕСЕКАНИЕ
INTO	В
IS	ЯВЛЯЕТСЯ
LEVEL	УРОВЕНЬ
LIKE	ПОДОБНО
LIMITED	ОГРАНИЧИВАЯСЬ
LOOP	ЦИКЛ
MAX	МАКСИМАЛЬНОЕ
MIN	МИНИМУМ
MINUS	МИНУС
MODE	РЕЖИМ
NATURAL	ЕСТЕСТВЕННЫЙ
NEW	НОВОЕ
NEXTVAL	следующее значение
NOCOMPRESS	не сжатое
NOT	НЕ
NULL	ПУСТОЙ (ОПЕРАТОР, УКАЗАТЕЛЬ)
NUMBER	НОМЕР
NUMBER_BASE	базовый номер
OF	ИЗ
ON	НА
OPEN	ОТКРЫТО
OPTION	ОПЦИЯ
OR	ИЛИ
ORDER	ПОРЯДОК
OTHERS	ДРУГИЕ
OUT	СНАРУЖИ
PACKAGE	ПАКЕТ
PARTITION	РАЗДЕЛ
POSITIVE	ПОЛОЖИТЕЛЬНО (УВЕРЕННЫЙ)
PRAGMA	ПСЕВДОКОММЕНТАРИЙ

Обзор

Язык SQL является непроцедурным языком работы с базой данных. При этом это означает что пользователь может только выполнить каждую команду отдельно и нельзя одновременно задать 2 и более оператора в командной строке.

Язык PL/SQL – процедурное расширение языка , позволяющее описать алгоритмические действия с данными.

Основные понятия

Команда - Указываемое Вами действие, которое должен выполнить SQL*Plus или RDBMS ORACLE.

Блок - Группа команд SQL и PL/SQL, логически связанные между собой по логике алгоритма.

Таблица - Основная единица хранения информации в RDBMS ORACLE.

Запрос - Оператор SELECT, предназначенный для извлечения данных из одной или из нескольких таблиц.

Результаты запроса - Данные, извлеченные из базы данных при выполнении запроса.

Отчет - Результаты запроса, отформатированные Вами с помощью команд SQL*Plus.

Таблицы, столбцы и строки

Данные хранятся в БД в виде таблиц, состоящих из столбцов и строк. В таблице с одним столбцом можно хранить одномерный список. Таблица с несколькими столбцами хранит информацию, для которой имеет значение сочетание строк и столбцов.

Каждый столбец представляет собой один атрибут строки. Каждая строка таблицы содержит одну запись и однозначно идентифицируется значением псевдостолбца ROWID. Один или несколько столбцов таблицы могут быть первичными, внешними или уникальными ключами. На таблицы и столбцы ссылаются, используя их имена (например, emp.sal).

Транзакции (Неделимая последовательность)

Транзакцией в SQL называется логически неделимая последовательность операторов, рассматриваемая как единое целое.

Результаты выполнения операторов, входящих в транзакцию, могут быть либо сохранены в БД при помощи оператора COMMIT, либо полностью аннулированы оператором RKLLBACK (или ROLLBACK до точки сохранения).

Транзакция начинается с 1-го выполняемого оператора, либо с 1-го оператора после COMMIT или ROLLBACK. Транзакция заканчивается при выполнении операторов COMMIT или ROLLBACK.

Чтобы оградить данные от модифицирования другими пользователями, в начале транзакции следует выполнить оператор SET TRANSACTION READ ONLY. При этом не допускается и изменение данным самим пользователем , издавшим директиву.

Форма описания синтаксиса

Синтаксис - это порядок описания некоторых действий выраженный условными словами или символами. При описании выражений любого языка программирования требуется описать как выражения соединены в единое целое.

(НФБ - нормальная форма Бэкуса.)

Обязательные элементы заключены в фигурные скобки {}. Вы ДОЛЖНЫ вводить один из разделенных вертикальной чертой элементов.

Необязательные элементы заключены в квадратные скобки []. Вы можете ввести один элемент из списка элементов, разделенных вертикальной чертой. Три точки после элемента означают, что предыдущий элемент может быть введен более одного раза.

Фигурные, квадратные скобки и вертикальные черточки вводить НЕ НАДО.

Списки параметров заключены в круглые скобки. Элементы списка параметров разделяются запятыми.

Слова, написанные ПРОПИСНЫМИ БУКВАМИ обозначают ключевые слова.

Слова, написанные строчными буквами, обозначают имена и выражения, которые могут быть различными.

Весь синтаксис описан с использованием данных правил.

Под синтаксисом понимается порядок и правила описания использования ключевых слов

Имена и пароли Personal-Oracle-7

Имя	Пароль	Права
SCOTT	TIGER	CONNECT RESOURCE
SYSTEM	MANAGER	DBA
SYS	SYS [Windows 95 идентификационный номер исключая пробелы]	CONNECT RESOURCE DBA EXP_FULL_DATABASE IMP_FULL_DATABASE

DEMO	DEMO	CONNECT RESOURCE
PO7	PO7	DBA
Internal	без пароля	используется при работе с SQL*DBA для проведения таких операций как запуск и остановка базы данных.

Идентификация пользователя

Пользователь, имеющий доступ к базе данных , идентифицируется в ней по имени и паролю.

Пользователь может владеть таблицами, представлениями данных и последовательностями. Также пользователю могут быть пожалованы права SELECT, INSERT, DELETE и UPDATE на аналогичные объекты других пользователей. Новый пользователь создается при помощи оператора GRANT, выполняемого польз., имеющим права DBA.

Ввод и выполнение команд:

Курсор на экране вашего дисплея (как правило курсор выглядит либо как подчеркивание, либо как светящийся прямоугольник, либо как вертикальная мигающая черта) находится в конце строки приглашения.

Курсор указывает на место, в котором появится следующий введенный Вами символ.

Команды можно вводить как строчными, так и прописными буквами.

При вводе команд SQL*Plus, SQL или PL/SQL вводите между словами хотя бы один пробел (или символ табуляции, а можно и [Ввод]).

Лишние пробелы, табуляция и переходы на новые строки можно использовать для облегчения последующего чтения команд. В ответ на приглашение можно вводить :

- команды языка SQL, предназначенные для работы с информацией в БД;
- блоки команд PL/SQL, предназначенные для работы с информацией в БД;
- процедуры , функции и пакеты для обеспечения хранения порядка некоторых действий на сервере и вызов этих десйствий с помощью одного оператора.
- хранимые процедуры выступают так-же как источник данных для клиентных приложений.
- команды SQL*Plus, предназначенные для форматирования результатов запроса, или для установки параметров, редактирования и сохранения команд SQL и PL/SQL.

Команды SQL, PL/SQL и SQL*Plus следует вводить в ответ на приглашение

```
SQL>
```

SQL*Plus сохраняет введенные Вами команды SQL и блоки PL/SQL в своем буфере (т.н. буфере SQL). Вы можете, если хотите, создать дополнительные буферы для хранения команд.

Команды SQL нужно заканчивать:

- символом ; (точка с запятой) в конце команды
- символом / (наклонная черта) на отдельной строке
- пустой строкой.

Точка с запятой (;) вызовет выполнение команды. В конце последней строки введите ;, затем нажмите [Enter] и команда выполнится. Если Вы нажмете [Enter]не введя предварительно ;, то SQL*Plus откроет новую строку и выведет на экран ее номер.

Косая черта (/), стоящая на отдельной строке, также вызывает выполнение команды (последней, которая находится в буфере SQL). После последней строки команды нажмите [Enter], затем на открывшейся строке введите / и снова нажмите [Enter].

Терминатор косая черта имеет более сильное действие, чем терминатор точка с запятой. Поэтому при редактировании буфера SQL с помощью текстового редактора следите за отсуствием двух терминаторов одновременно.

Ввод в буфер SQL нескольких команд SQL не оформленных в непоименованный блок не разрешается

Команды SQL*Plus служат для управлениея командами SQL и PL/SQL, а также для форматирования и печати результатов запросов.

Для экономии времени многие команды SQL*Plus можно сокращать до одной или нескольких букв. Фактически данные команды есть средство настройки сервера для работы.

Файлы, используемые SQL*Plus

Из SQL*Plus Вы можете записывать команды в командный файл.

Кроме этого, Вы можете записывать в файл результаты отчетов (в т.н. спул-файлы).

Физическая структура командных и спул-файлов зависит от операционной системы вашего компьютера. Модифицировать их можно при помощи соответствующих команд операционной системы.

Выполнение команд операционной системы

Из SQL*Plus можно выполнять команды операционной системы. Это может понадобиться, например, чтобы посмотреть содержимое файла.

Чтобы выполнить команду ОС наберите HOST и собственно команду. Например, чтобы выполнить команду ОС DIRECTORY *.SQL следует ввести:

```
SQL> HOST  DIR  *.SQL
```

Программа SQL*PLUS

Программа является клиентной частью системы Клиент/Сервер и обеспечивает обращение к серверу на низком уровне командного интерфейса. Альтернативным вариантом является программа SQL*DBA , с помощью которой возможно обращение в серверу ORACLE на низком уровне командного интерфейса.

В SQL*Plus можно работать на языке SQL и на его расширенном процедурном варианте PL/SQL. С помощью языка SQL можно хранить и извлекать данные из реляционной базы данных ORACLE. SQL*Plus позволяет выполнять команды языка SQL, блоки команд на языке PL/SQL и многое другое. В частности, работая в SQL*Plus Вы можете:

- вводить, редактировать, сохранять, читать и выполнять команды SQL и блоки PL/SQL;
- использовать в вычислениях результаты выполнения запросов, а также форматировать и сохранять эти результаты;
- увидеть описание структуры любой таблицы;
- обрабатывать информацию в других БД и копировать данные из одной БД в другую;
- обмениваться сообщениями с пользователями.

Понятия языка

Алиас (альтернативное имя)

Таблице или столбцу можно присвоить альтернативное имя (алиас), которое действие будет действительно только в пределах оператора, в котором оно определено. Если альтернативное имя стоит после имени столбца в списке оператора SELECT, то оно будет использоваться вместо настоящего имени как заголовок для данного столбца.

```
SQL>  SELECT ename “ИМЯ”, empno “НОМЕР”, sal “ОКЛАД”
FROM emp
WHERE job = ‘ANALYST’
ORDER BY NAME;
```

Альтернативное имя таблицы можно использовать при соединении таблицы с самой собой в соотносящемся запросе.

При создании курсора альтернативное имя создается другим способом - заданием после имени курсора перечисления новых имён полей (указывать в скобках через запятую)

При использовании таблиц с одинаковыми именами полей требуется указание имени таблицы перед именем колонки таблицы.

```
SQL>SELECT wkr.ename, wkr.sal, sup.ename, sup.sal
FROM emp wkr, emp sup
WHERE wkr.mgr = sup.empno
AND wkr.sal > sup.sal;
```

Арифметические операторы языка SQL

позволяют обрабатывать числовые значения. Существуют также операторы для обработки дат.

Опер.	Действие	Пример
()	Изменяет старшинство операций	SELECT (x + y)/2...

+ -	Положит., отрицат. Значение	...WHERE bal <= -50
* /	Умножение, деление	SELECT discount * 100...
+ -	Сложение, вычитание	SELECT price - discount...

Текстовый оператор || выполняет конкатенацию (объединение) данных типа CHAR (строк символов). Пробелы учитываются, только если они стоят внутри одинарных кавычек (‘ ‘).

Булевы сравнения

Булевы операторы сравнения используются описания условий. Эти условия используются для управления выполнением программы на PL/SQL.

Оператор	С числами	С текстом	С датами
<	меньше чем	лексикографически меньше	раньше чем
>	больше чем	лексикографически больше	позднее чем
=	равно	совпадает с	тогда же
!=	не равно	не совпадает	в другое время
<=	меньше или равно	лексикографически меньше или равно	раньше чем или тогда же
>=	больше или равно	лексикографически больше или равно	позднее чем или тогда же

Также можно сравнивать и переменные типа BOOLEAN: TRUE > FALSE , FALSE < TRUE , TRUE != FALSE

Операторы сравнения

Оператор Значение/действие в SQL

=	равно
!= <>	не равно
>=	больше или равно
<=	меньше или равно
IN	равен любому элементу в
NOT IN	не равен любому элементу в
ANY	Сравнивает значение. с любым из значений. в списке. Употребляется после =, !=, >, <, <=, >=
ALL	Сравнивает значение. с каждым значением. в списке. Употребляется после =, !=, >, <, <=, >=
BETWEEN	больше или равно знач1 и меньше или равно знач2
NOT	Не больше и не равно
EXISTS	Истина, если подзапрос извлек хотя бы одну строку
IS NULL	Истина, если значение пусто (NULL) (проверка вида x = NULL не годится)

IS NOT NULL	Истина, если значение не пустое (не пишите x != NULL)

Все вышеперечисленные операторы вырабатывают значения TRUE (истина), FALSE (ложь) или NULL (пустое значение).

Условия

Условием в языке SQL называется сочетание одного или нескольких выражений и логических операторов, вырабатывающих значение TRUE (истина) или FALSE (ложь). Условия могут использоваться:

- в предложении WHERE операторов SELECT, INSERT, UPDATE, и DELETE, например:
- SELECT ename, sal FROM emp WHERE job = 'ANALYST'...
- в предложениях CONNECT BY, START WITH, и HAVING оператора SELECT, например:
- SELECT ename, mgr FROM emp
- CONNECT BY PRIOR empno = mgr START WITH ename = 'KING';

В языке PL/SQL условия употребляются в операторах IF и WHILE.

Ограничение FOREIGN KEY/REFERENCES:

- не дает выполниться операторам INSERT и UPDATE, если соответствующих значений нет в таблице, содержащей первичный ключ.
- не позволяет выполниться оператору DELETE, если его результат приводит к нарушению ограничения REFERENCES.
- должно ссылаться на столбец первичной таблицы (т.е. содержащей первичный ключ), определенный как PRIMARY KEY или UNIQUE.
- если не указан столбец (столбцы), то ограничение будет использовать тот столбец первичной ключа, который определен как PRIMARY KEY.
- может ссылаться только на таблицу. На представление данных или кластер ссылаться нельзя.
- требует, чтобы таблица, хранящая первичный ключ, принадлежала Вам и чтобы Вы имели к доступ с правом REFERENCE либо к таблице, либо только к столбцам, на которые Вы ссылаетесь при описании ограничения.
- никак не ограничивает использование тех же таблиц в описании других ограничений.
- требует совпадения типов данных в столбце (столбцах) внешнего ключа и в столбце (столбцах), к которым описывается ограничение.
- может ссылаться на ту же таблицу, что и в операторе CREATE TABLE.
- в одном ограничении не может ссылаться на столбец более одного раза.

DBA(Администратор базы данных)

Пользователь с именем SYSTEM является DBA (АБД—администратором БД). Он владеет таблицей PRODUCT_USER_PROFILE и имеет все права доступа к ней.

Все остальные пользователи могут иметь на эту таблицу только право SELECT (только выборка). Командный файл PUPBLD, при выполнении, предоставляет всем пользователям (PUBLIC) право SELECT на таблицу PRODUCT_USER_PROFILE.

Если DBA захочет запретить какому-то пользователю выполнять определенную команду SQL или SQL*Plus, то он может (если он пользователь DBA) внести в таблицу PRODUCT_USER_PROFILE запись со следующей информацией:

столбец USERID = имя пользователя

ATTRIBUTE = команда

CHAR_VALUE = слово DISABLED

столбцы SCOPE, NUMERIC_VALUE и DATE_VALUE должны быть пустыми.

Чтобы снова разрешить пользователю выполнять запрещенную команду, DBA должен стереть строку, описывающую запрет.

Пользователь с правом DBA может использовать таблицу PRODUCT_USER_PROFILE для обеспечения недоступности след. команд:

CONNECT	QUIT	ALTER	DROP
EDIT	RUN	AUDIT	GRANT
EXIT	SAVE	CONNECT	INSERT
GET	SPOOL	CREATE	LOCK
HOST	START	DELETE	NOAUDIT
RENAME	REVOKE	SELECT	UPDATE
VALIDATE			

Столбцы в таблице PRODUCT_USER_PROFILE имеют след. назначение:

PRODUCT -

содержит название продукта (в нашем случае SQL*Plus). Шаблоны (% и _) использовать нельзя. USERID содержит имя_пользователя (написанное прописными буквами), к которому относится запрет.

Чтобы распространить область действия запрета на более чем одного пользователя, можно использовать шаблонный символ % или делать несколько записей.

ATTRIBUTE -

содержит запрещаемую для выполнения команду (прописными буквами). Нельзя использовать шаблоны.

SCOPE-

не имеет значения для SQL*Plus. Рекомендуется заносить NULL.

NUMERIC_VALUE -

не имеет значения для SQL*Plus. Рекомендуется заносить NULL.

CHAR_VALUE -

должно содержать слово “DISABLED”. Шаблоны использовать нельзя.

DATE_VALUE

не имеет значения для SQL*Plus. Рекомендуется заносить NULL.

Разделители

( )	оператор; служит для объединения параметров
+ - * /	арифметические операторы
= < >	логические операторы
;	признак конца оператора SQL
%	признак системного атрибута PL/SQL
,	разделитель элементов в списке
.	ссылка на таблицу.имя_поля и запись.имя_поля (в PL/SQL) (Точка . на отдельной строке свидетельствует об окончании блока PL/SQL)
@	ссылка на удаленную БД; @ имя_файла выполняет командный файл
‘ ‘	ограничивает строки символов (‘еклмн оклжп чшц’)
“ “	позволяет писать в идентификаторах что угодно
:	простые переменные или переменные связи включающего языка
<> != ~= ^= <= >=	операторы сравнения в PL/SQL
<< >>	ограничители имен циклов, блоков и меток (для GOTO) в PL/SQL
--	начало строки комментария в блоке PL/SQL
/*	начало многострочного комментария
*/	конец многострочного комментария
..	диапазон числового цикла FOR в PL/SQL
:=	присвоение значения переменной в PL/SQL
**	возведение в степень
||	объединение строк

Ключи

Первичным ключом (PRIMARY KEY) называется один или несколько столбцов, предназначенных для однозначной идентификации строк в таблице. Значения первичных ключей не должно изменяться и не должно быть пустым.

Внешним ключом (FOREIGN KEY) называется столбец, содержащий значения первичного ключа другой таблицы. Внешний ключ может принимать пустое значение. К примеру, если столбец DEPTNO является первичным ключом в таблице DEPT, то столбец DEPTNO в таблице EMP будет внешним ключом, т.к. он содержит ссылки на DEPTNO таблицы EMP.

Уникальный ключ (UNIQUE KEY) во всем аналогичен первичному, за исключением того, что он предназначается для обеспечения уникальности каждой записи, например, номера телефона или номера водительских прав.

Первичный ключ может и не содержать никакой полезной информации, а служить только для идентификации строк. В таблице может быть несколько первичных ключей.

Обработка ошибок

Определенные заранее (внутренние) или определенные пользователем условия, возникновение которых считается ошибкой, называются особыми состояниями. В PL/SQL можно определить действия, выполняемые в случае возникновения ошибки.

Особые состояния можно описывать в секции DECLARE, а затем проверять их наличие в теле блока. Если особая ситуация существует, то можно выполнить оператор RAISE, который остановит нормальное выполнение блока и передаст управление обработчику особых ситуаций, описанному в секции EXCEPTION данного блока.

Функции диагностирования ошибок

SQLCODE

Возвращает код внутреннего состояния, вызвавшего передачу управления обработчику особых состояний.

Если состояние, вызвавшее обработчик, не является внутренним для ORACLE, а описано пользователем в текущем (или объемлющем) блоке, то SQLCODE возвращает +1. Вне обработчика особых состояний SQLCODE возвращает 0.

Функция SQLCODE позволяет распознавать особые ситуации в обработчике состояний OTHERS.

SQLERRM(код_ошибки)

возвращает сообщение, соответствующее заданному коду ошибки. Чтобы получить сообщение , соответствующее текущему коду ошибки в SQLCODE, нужно опустить код ошибки.

Вне обработчика особых ситуаций SQLERRM выдает сообщение “normal, successful completion” (успешное завершение).

Чтобы использовать SQLCODE или SQLERRM в операторе SQL, нужно присвоить их значение переменной, например:

```
sql_code_num := SQLCODE
```

Выражения

Выражениями в языке SQL называются совокупность литералов, переменных и констант, соединенных операторами, выполняющими над ними какие-либо действия. Значение, получающееся в результате вычисления выражения, будет иметь тот же тип данных, что и элементы выражения.

Выражения можно использовать:

- в списке оператора SELECT
- в условиях после предложений WHERE и HAVING
- в предложениях CONNECT BY, START WITH и ORDER BY
- в предложении VALUE оператора INSERT
- в предложении SET оператора UPDATE

Порядок вычисления (старшинство операций)

- 1: эл-ты, заключенные в скобки
- 2: возведение в степень
- 3: одноместные операторы
- 4: умножение и деление
- 5: сложение, вычитание и конкатенация

Форматирование дат

Элемент	Значение в TO_CHAR
CC или SCC	Век ил -век до н.э.
YYYY или SYYYY	Год или -год до н.э.
YYY или YY или Y	Последние 3,2,1 цифры года
Y,YYY	Год с запятой в указанном месте
YEAR или SYEAR	Произношение года
AD или BC	Индикатор до н.э.—AD (anno Domini) или BC (before Christ)
A.D. или B.C.	A.D. или B.C.
AM или PM	Индикатор меридиана
A.M. или P.M.	Индикатор меридиана с точками
HH или HH12	Часы (1-12)
HH24	Часы (0-23)
MI	Минуты (00-59)
SS	Секунды (00-59)
SSSSS	Число секунд после полуночи (0-86399)
Q	Квартал (1-4)
MM	Месяц (01-12)
MONTH	Название месяца (длина поля - 9 символов)
MON	3-х буквенное название месяца (ЯНВ)
WW	Неделя года (1-52)
W	Неделя месяца (1-5)
DDD	День года (1-366)
DD	День месяца (1-31)
D	Номер дня недели (номер) (1-7)
DAY	Название дня (длина поля - 9 символов)
DY	3-х буквенное сокращение названия дня (Птн)
J	Юлианская дата: число дней считая с 1 Января 4712 г. до н.э.

Модификаторы формата

Модификаторы регистра букв в SQL:

MON = APRIL Mon = April mon = april

DAY = FRIDAY Day = Friday day = friday

Элемент	Смысл
fm	Префикс, вызывающий подавление дополнения пробелами в форматах MONTH и ему подобных, что приводит к получению результ. строк переменной длины. Подавление остается в силе вплоть до следующего использования fm, восстанавливающего дополнение пробелами.
TH	производит числительное (DDth для 5th)
SP	произношение числа (ddSP для пяти)
THSP, SPTH	произношение числительного (DdthSP для Fifth)
‘string’	Вставляет строку в результирующую.

Форматирование чисел

В параметрах преобразования TO_CHAR можно использовать след. элементы описания формата в SQL.

Элемент	Пример	Описание

9	99999	Число девяток задает ширину поля.
0	099	Вывести перед числом незначащие нули.
$	$999	Перед числом вывести знак доллара ($).
B	B99	Изображать нули как пробелы, а не как '0'.
MI	999MI	Вывести минус после отрицательного числа.
PR	999PR	Заключить отрицательное число в угловые скобки (<123>).
,	9,999	Вывести запятую в указанном разряде.
.	9.999	Вывести точку в указанном разряде.
V	99V99	Умножить значение на 10 столько раз, сколько есть девяток после 'V'.
E	9.99EEEE	Вывести число в экспоненциальном формате.
DATE	DATE	Записывает дату в численном виде (в формате ММ/ДД/ГГ).

Идентификаторы

Идентификаторами называются имена, присваиваемые переменным и константам, таблицам, представлений. данных, столбцам, индексам и прочим объектам БД.

Идентификатор должен начинаться с латинской буквы и быть не длиннее 30 символов.

Идентификатор может состоять из латинских букв, цифр и символов $ _ и #.

Идентификатор не должен содержать пробелы и совпадать с каким-нибудь ключевым словом SQL или PL/SQL. Регистр букв (т.е. прописные/строчные) не имеет значения.

Если идентификатор заключен в кавычки ("Id"), то он может совпадать с ключевым словом и содержать любые символы, включая пробелы.

Идентификаторы должны отделяться друг от друга как минимум одним пробелом или знаком препинания.

ДОПУСТИМО	НЕ ДОПУСТИМО	ПОЧЕМУ НЕДОПУСТИМО
Balance_Due	BALANCE DUE	содержит пробел
client_1	1st_client	начинается с цифры
"row"	row	ключевое слово
largest_$	$largest	начинается не с буквы

Неявные преобразования

PL/SQL в некоторых случаях преобразует тип данных переменной в требуемый путем неявного вызова функций TO_NUMBER, TO_CHAR или TO_DATE. Например:

Присваивания (CHAR в NUMBER или DATE, NUMBER или DATE в CHAR)

```
переменная := выражение
INSERT INTO таблица VALUES...
UPDATE таблица SET столбец = выражение...
SELECT выражение INTO переменная FROM...
```

Вычисление выражений (CHAR в NUMBER, CHAR в DATE)

```
Простые выражения        :  bal + '44'
Булевы выражения         : bonus > salary / '20'
Функции и вызовы процедур  : MOD(counter, '3')
```

Условия в предложении WHERE: WHERE hiredate = '15-MAR-89'

Для преобразования из CHAR в DATE требуется строка в виде ДД-МЕС-ГГ.

Пределы

Эти ограничения имеются на многих системах,

Параметр	Предельное значение в SQL*Plus
длина имени файла	зависит от ОС (в Personal Oracle-7 -- 8.3)
длина имени пользователя	30 символов
длина имени переменной	30 символов
длина значения переменной	240 символов

число пользовательских переменных	1024
кол-во перем. в списке INSERT INTO	50
переменных в команде SQL	100
длина командной строки	500 символов
длина знач. Типа LONG в SQL*Plus	250 символов
размер выходной строки	500 символов (минимум = 5 символов)
длина строки после подстановки переменных	1 000 символов (внутренний параметр)
число строк в команде	500 (предполагая 80 симв. в строке)
число строк на странице	50 000
макс. Ширина строки таблицы	60 000 символов в ОС VMS в других = 32,767
кол-во строк в выборке в массив	5 000 строк
вложенные командные файлы	20 в ОС VMS, CMS, Unix В других – 5
номеров страниц	99 999 страниц

Связи

Связь с базой данных делает возможной работу с данными в удаленной базой данных.

Чтобы пользоваться связью ,нужно знать имя пользователя в удаленной базе. При этом между системами (местной и удаленной) должно быть установлено соединение и SQL*Net должен быть активным и на местной, и на удаленной.

Для связи с БД, отличающимися от ORACLE, нужен активный SQL*Connect. При ссылке на таблицу, находящуюся в удаленной БД, в предложении FROM запроса или подзапроса необходимо указать имя связи. Например, запрос SELECT * FROM powers.dept@PACIFIC через связь с именем PACIFIC извлекает данные из таблицы DEPT, принадлежащей пользователю POWERS.

Связь с БД используется в распределенных запросах. Кол-во связей в простом операторе как правило ограничено четырьмя. Через связь нельзя извлекать данные типа LONG.

Литералы

Литералами в языке SQL называются числа или строки символов.

Числовые литеры:

могут быть целыми числами (типа 3, 1444, 23) или вещественными (типа 3.14159, .25, -12, 2.e, 7E). В списках числовые литералы отделяются друг от друга запятыми. Для облегчения читаемости неплохо после каждой запятой добавлять пробел.

Строковые литеры:

один или несколько символов, заключенные в апострофы (‘одинарные кавычки’). Апостроф в самом литерале записывается в виде двух смежных апострофов. Например,

‘You’’re a strong, happy person, aren’’t you?’. Заметьте, что используется именно два апострофа, а не кавычки

(“). В кавычки иногда заключают идентификаторы.

Литеры дат:

строковые литералы в формате дат (‘09-ЯНВ-42’).

Булевы литеры:

предопределенные константы TRUE (истина), FALSE (ложь) и NULL (пусто).

Логические операторы.

Логические операторы в SQL позволяют обрабатывать результаты вычисления условий.

Пустые значения

Пустое значение означает, что данные не известны, отсутствуют или бессмысленны в данном контексте. Пустое значение могут иметь столбцы с любым типом данных, если они не были определены как NOT NULL.

Пустые значения применяются в случаях, когда конкретные значения неизвестны или бессмысленны. Пустое значение и значение=ноль представляют собой совершенно разные вещи. Все выражения, содержащие хотя бы одно пустое значение, равны также пустому значению.

Для оперативной подстановки непустых значений вместо пустых следует использовать функцию NVL. Например, NVL(comm, 0) вернет значение comm, если оно не пустое, или 0, если оно пустое.

При проверке на пустое значение надо использовать только операторы IS или IS NOT. Любой другой оператор при сравнении с пустым значением всегда вернет пустое значение.

Групповые функции при вычислениях игнорируют пустые значения.

Прочие операторы

Оператор	Смысл или действие в SQL
(+)	Предыдущий столбец является столбцом избыточного соединения (в соединении таблиц).
(*)	Извлекать все столбцы таблицы (SELECT * FROM...). таблица.* Извлекать все столбцы указанной таблицы (SELECT таблица.* FROM...).
ALL	В запросе или группировании не отбрасывать повторяющиеся значения, а извлекать все.
DISTINCT	В запросе или группировании отбрасывать повторяющиеся данные (извлекать только одно вхождение). Подразумевается всегда ALL.
PRIOR	Используется в предложении START WITH ... древовидного запроса для определения отношения отец-сын.

Оператор избыточного соединения

```
SELECT ...
FROM таблица1, таблица2, ...
WHERE таблица1.столбец = таблица2.столбец (+)
```

Оператор избыточного соединения (+) вызывает извлечение из соединенных таблиц не только строк, удовлетворяющих условию соединения, но еще и строк, не отвечающих условию, из таблицы.столбца, помеченных оператором (+).

Избыточное соединение в операторе SELECT можно производить только с одной таблицей.

Особые ситуации в PL/SQL

DUP_VAL_ON_INDEX	ORA-00001	Оператор INSERT/UPDATE попытался повторно занести какое-то значение в столбец с атрибутом UNIQUE (уникальный).
INVALID_CURSOR	ORA-01001	В операторе неправильно указан курсор
INVALID_NUMBER	ORA-01722	Ошибка при преобразовании из CHAR в NUMBER: строка не является числом
LOGIN_DENIED	ORA-01017	Ошибочные имя_пользователя/ пароль
NO_DATA_FOUND	ORA-01403	Оператор SELECT не извлек ни одной строки
NOT_LOGGED_ON	ORA-01012	Обращение к ORACLE без предварительного подсоединения к нему
PROGRAM_ERROR	ORA-06501	Внутренняя ошибка PL/SQL
STORAGE_ERROR	ORA-06500	Область памяти PL/SQL заполнена или испорчена
TIMEOUT_ON_RESOURCE	ORA-00051	Ресурс не освободился за указанное время
TOO_MANY_ROWS	ORA-01427	Оператор SELECT извлек более чем одну строку
VALUE_ERROR	ORA-06502	Ошибка при преобразовании (арифм/числ/строк)
ZERO_DIVIDE	ORA-01476	Попытка деления на 0

Дополнительно смотри справочную систему Oracle (Oracle-7 Error Message)

Любую ошибку можно сделать доступной для анализа с помощью директивы

PRAGMA EXCEPTION_INIT

В приложении можно устанавливать свои коды ошибок от -20,999 до -20,000

Параметры файлов SQL (&1, &2).

Параметрами называются переменные, обозначаемые & (амперсандом) и следующим за ним числом (например, &1). Параметры используются только в командных файлах. Значения для каждого параметра задаются как аргументы команды START. Если параметр может быть датой или текстовой строкой, то его следует заключить в апострофы:

```
SELECT * FROM EMP
```

```
WHERE JOB='&1'  /*текстовые данные*/
AND SAL=&2;     /*число*/
```

Теперь, при выполнении командного файла, в команде START нужно вводить значения для этих параметров. Значения следует писать после имени самого запускаемого файла. 1-значение получит параметр &1, 2-е—параметр &2 и т.д. Например, приведенная ниже команда запускает командный файл MYFILE и присваивает параметру &1 знач. 'СЕКРЕТАРЬ', а пар. &2 -- 7900.

```
SQL> START MYFILE CLERK 7900
```

Каждый параметр, получивший значение из аргументов команды START, автоматически описывается командой SQL*Plus DEFINE.

Псевдостолбцы

Псевдостолбцы в SQL очень схожи с обыкновенными столбцами, за исключением того, что они не хранятся ни в одной таблице. Из псевдостолбцов можно извлекать данные, но нельзя добавлять, обновлять и удалять (т.е. нельзя делать INSERT, UPDATE и DELETE).

Псевдостолбец	Содержит
последоват. NEXTVAL	следующее сгенерированное число
последоват. CURRVAL	последнее сгенерированное число
ROWID	уникальный идентификатор строки таблицы
ROWNUM	порядковый номер строки в результатах выборки из одной или нескольких таблиц
LEVEL	1= отец, 2= сын, 3= внук, и т.д.

QUIT (покинуть)

{QUIT | EXIT} [SUCCESS | FAILURE | WARNING | n | переменная]

Команда QUIT подтверждает все неподтвержденные изменения в БД и завершает работу SQL*Plus, возвращая управление операционной системе.

QUIT или EXIT

полностью взаимозаменяемы

SUCCESS	нормальное завершение
FAILURE	завершение с возвратом соотв. кода ошибки
WARNING	завершение с возвратом кода, означающего предупреждение.

Команда QUIT без каких либо параметров вызывает такое же завершение, как и QUIT SUCCESS

n

n = возвращаемый при завершении код (целое число) переменная определенная польз. или системная переменная (например, SQL.SQLCODE). Команда QUIT переменная заканчивает работу, возвращая значение переменной в качестве кода завершения.

Благодаря команде QUIT можно использовать коды возврата, принятые в конкретной ОС и, следовательно, распознавать неожиданные ситуации при выполнении командных файлов SQL*Plus в пакетном режиме. Способ распознавания возвращаемых кодов зависит от ОС. Слова SUCCESS, WARNING и FAILURE

представляют значения, свойственные ОС, с которой работает ваша ЭВМ. В некоторых системах коды WARNING и FAILURE однозначны. Чтобы выполнить

QUIT по возникновению какого-то условия, используйте оператор WHENEVER SQLERROR.

Пример: В данном примере программа возвращает код результата последнего выполненного оператора в блоке.

```
SQL> QUIT SQL.SQLCODE
```

Способ получения возвращенного кода зависит от ОС.

Представления данных и индексы

Представления данных (иногда называемые виртуальными таблицами) не содержат данные как таковые, а являются неким отображением информации, хранящейся в какой-то одной или в нескольких таблицах (т.н. базовых таблицах). Представления. данных позволяют:

- Ограничить доступ к данным, хранящимся в некоторых столбцах и (или) строкам.
- Скрыть повышенную сложность информации; При помощи одного предст. можно обрабатывать информацию из нескольких таблиц.
- Уменьшить сложность синтаксиса; Одно предст. может отображать данные из нескольких таблиц, не требуя каждый раз выполнения операций

- соединения.
  Преподносить данные с разных точек зрения; Столбцам представления. можно присваивать любые имена, не изменяя при этом имена столбцов базовой таблицы. Индекс, в свою очередь, обеспечивает быстрый доступ к строкам таблицы.

  Если команда SQL обрабатывает меньше 25% строк, то использование индекса существенно увеличивает скорость выполнения команды.

  Уникальный индекс гарантирует, что в таблице не будет строк с совпадающими значениями некоторого столбца (являющегося первичным ключом).

Запрос(требование извлечь данные)

Запросом называется изложенное на языке SQL требование извлечь данные, хранящиеся в одной или в нескольких таблицах. Подзапросом называется запрос, находящийся в теле другого оператора. Запросы используются:

- Для определения набора строк обрабатываемой таблицы в операторах COPY, INSERT и CREATE TABLE.
- Для получения значений при сравнении в предложениях WHERE, HAVING и START WITH, употребленных в операторах SELECT, UPDATE или DELETE.
-

SELECT...WHERE выражение оператор запрос
...HAVING выражение оператор запрос
...START WITH выражение оператор запрос
UPDATE...SET(столбец, столбец,..) оператор запрос
...WHERE выражение оператор запрос
DELETE...WHERE выражение оператор запрос
INSERT...запрос

Удаленные базы данных

RDBMS ORACLE используется на нескольких разных КОМПЬЮТЕРАХ, объединенных в сеть.

Использование в сети называется распределённым хранением и обработкой информации

Эти КОМПЬЮТЕРЫ могут находиться на любом расстоянии друг от друга. Например, два КОМПЬЮТЕРА могут стоять в соседних комнатах, а третья находится в другом городе.

Удаленной базой данных называется любая БД, находящаяся на другом или на вашем копьютере, но не являющаяся для Вас рабочей. Работать с информацией в удаленной БД можно, если на ней есть SQL*Net и совместимый с местной БД сетевой драйвер. Получить доступ к удаленной БД можно прямо из SQL*Plus (с помощью команды CONNECT), или при запуске SQL*Plus

Доступ к удаленной БД из SQL*Plus

Чтобы подсоединиться к удаленной БД из SQL*Plus надо в оператор CONNECT включить описание этой БД в форме, пригодной для SQL*Net.

Например:

CONNECT SCOTT@описание_БД

CONNECT SCOTT/TIGER@описание_БД

После подсоединения эта база станет вашей рабочей БД до тех пор, пока Вы не сделаете одно из трех:

1. соединитесь с другой БД
2. отсоединитесь от данной БД (оператор DISCONNECT)
3. закончите работу в SQL*Plus

Форма описания БД в операторе CONNECT зависит от эксплуатируемого на вашем компьютере протокола SQL*Net.

Соединение с удаленной БД при запуске SQL*PLus

Чтобы подсоединиться к удаленной БД при запуске SQL*Plus, надо включить в команду SQLPLUS описание этой БД.

SQLPLUS SCOTT@описание_БД

SQLPLUS SCOTT/TIGER@описание_БД

имя_пользователя, пароль, и описание_БД должны быть допустимыми на удаленной БД. В этом

случае SQL*Plus после запуска соединит Вас с указанной БД и она будет вашей рабочей до тех пор, пока Вы не сделаете одно из трех:

1. соединитесь с другой БД
2. отсоединитесь от данной БД (оператор DISCONNECT)
3. закончите работу в SQL*Plus

Управление защитой данных

GRANT (дозволить)

Оператор GRANT можно использовать для достижения 3-х целей:

- для предоставления возможности доступа к БД;
- для предоставления возможности доступа к области хранения (можно также указать предельное кол-во пространства для каждого пользователя);
- для предоставления возможности доступа к объектам БД (типа таблиц, представлений данных, генераторов последовательностей и т.п.).

Для 1-й цели

```
GRANT право_в_БД, право_в_БД,...
TO имя_пользователя, имя_пользователя,...
IDENTIFIED BY пароль, пароль,...
```

Такая форма оператора GRANT позволяет предоставить пользователю (пользователям) возможность получать доступ к БД. При помощи данного оператора любой польз. может изменить свой пароль. Для других операций при помощи GRANT нужно иметь права DBA.

право_в_БД

предоставляемое право: CONNECT, RESOURCE и/или DBA.

пароль

пароль для соотв. имени польз. Если GRANT используется для предоставления дополнительных прав уже существующему польз., то пароль можно опустить. Если используется несколько имен польз. и паролей то 1-й пароль должен относиться к 1-му имени, 2-й—ко 2-му и т.д.

Предоставление права CONNECT регистрирует в БД новое имя польз. и позволяет этому пользователю подсоединяться к БД, манипулировать с объектами на которые он имеет соотв. права и создавать представления данных, синонимы и связи с удаленными БД.

Право RESOURCE разрешает пользователю создавать в БД различные объекты, включая таблицы, индексы, кластеры и последовательности.

Право DBA дает пользователю возможность обойтись без многих прав, которые обычно требуются для использования объектов БД. Пользователи с правом DBA также могут выполнять различные административные действия, типа

CREATE TABLESPACE (создать область хранения) и CREATE ROLLBACK SEGMENT

(создать сегмент отката).

Для 2-й цели

```
GRANT RESOURCE квота K | M
ON область_хранения
TO PUBLIC | польз., польз.,...
```

Такая форма оператора GRANT делает область хранения доступной для пользователей. Можно также указать предельное кол-во пространства, которое может использовать конкретный пользователь.

RESOURCE

дает пользователям право создавать объекты в указанной области хранения

квота K или M

кол-во байтов в области хранения, которое может использовать пользователь. Если его ограничивать не нужно, то указание квоты следует опустить.

Чтобы отобрать право RESOURCE, укажите квоту = 0. K означает, что квота задается в килобайтах

(квота x 1024), M -- -- что в мегабайтах (квота x 1 048 576).

TO PUBLIC или польз., польз.,...

дает право RESOURCE всем (PUBLIC) или только перечисленным пользователям.

Для 3-й цели

```
GRANT право_на_объект, право_на_объект,... | ALL
ON польз.объект
TO PUBLIC | польз, польз,...
WITH GRANT OPTION
```

Такой оператор GRANT делает возможным доступ к объектам БД типа таблиц, представлений данных и последовательностей). Чтобы выполнить такой оператор, Вы должны либо быть владельцем указанного объекта, либо иметь на него право GRANT OPTION, либо иметь право DBA.

право_на_объект, право_на_объект,...

для таблиц,	ALTER, DELETE, INDEX, INSERT, REFERENCES, SELECT, UPDATE
для предст. Данных,	DELETE, INSERT, SELECT, UPDATE
для последовательностей	ALTER или SELECT

Право UPDATE позволяет ограничить возможность обновления определенных столбцов. Синтаксис в этом случае следующий:

GRANT UPDATE столбец, столбец,...

ALL PRIVILEGES

предоставляет все возможные права на этот объект.

WITH GRANT OPTION

позволяет польз., получившему (получившим) указанные права, предоставлять их другим польз.

REVOKE (отобрать)

Оператор REVOKE может использоваться для достижения одной из 3-х целей:

1. чтобы отобрать у одного или сразу у нескольких пользователей какие-то права на базу данных;
2. чтобы лишить одного или нескольких польз. возможности создавать объекты в указанной области хранения (и, след., предотвратить увеличение объема пространства, занимаемое их объектами);
3. чтобы сделать какие-то таблицы, предст. данных или последовательности недоступными для одного или нескольких пользователей..

1-й случай

```
REVOKE CONNECT, RESOURCE, DBA
FROM польз, польз,...
```

Такой оператор REVOKE лишает перечисленных пользователей всех прав в базе данных. Перед тем как лишить пользователя права CONNECT рекомендуется предварительно уничтожить все принадлежащие ему объекты. Чтобы выполнить оператор REVOKE ,нужно иметь права DBA.

2-й случай

```
REVOKE RESOURCE ON область_хранения
FROM польз, польз,...
```

Такой оператор REVOKE лишит указанных пользователей возможности создавать объекты в указанной области хранения и предупредит увеличение занимаемого ими объема.

3-й случай

```
REVOKE право_на_объект, право_на_объект,... | ALL
ON польз.объект
FROM PUBLIC | польз, польз,...
```

Такой оператор REVOKE лишает перечисленных пользователей прав доступа к таблицам, предст. данных и последовательностям. Чтобы выполнить этот оператор, Вы должны быть владельцем объектов, либо иметь на таблицу право

WITH GRANT OPTION.

Если польз. получил право доступа к таблице от более чем одного человека, то он может пользоваться им до тех пор, пока все давшие ему это право не лишат его такового.

Право_на_объект ;

Для таблиц:	ALTER, DELETE, INDEX, INSERT, REFERENCES, SELECT UPDATE
Для представлений :	DELETE, INSERT, SELECT UPDATE
Для последовательностей:	ALTER SELECT

CONNECT (подсоединиться)

CONN[ECT] [вход]

Команда CONNECT подключает пользователя к ORACLE.

вход

Указывается в следующем формате:

имя_польз[/пароль][@описание_БД]

| /

имя_польз[/пароль]

Задает имя и пароль пользователя, с которыми вы хотите подключиться к ORACLE. Если вы опустили username и пароль, SQL*PLUS выдаст подсказку для их ввода. Если вы ввели наклонную черту (/) или просто ввели [Return] в ответ на подсказку для ввода username, SQL*PLUS зарегистрируется, используя значения по умолчанию (см "/" ниже).

Если вы опустили только пароль, SQL*PLUS выдаст подсказку для ввода пароля. Во время ввода пароля после подсказки, SQL*PLUS не выводит его на экран.

Представляет умалчиваемый (ops$) вход в систему. Вы не можете вводить database_specification, если вы используете вход в систему по умолчанию. При входе в систему по умолчанию, SQL*PLUS пытается зарегистрируется, используя имя пользователя OPS$name, где name- это ваше имя(пользователя)в операционной системе.

описание_БД

состоит из SQL*NET-строки подключения. Ее синтаксис зависит от вашего SQL*NET протокола. За информацией смотрите соответствующее руководство по SQL*NET или обратитесь к АБД (администратор БД).

SQL*PLUS не выдает подсказки для ввода спецификации БД, но он использует вашу БД по умолчанию, если вы не задали данный параметр.

CONNECT записывает произведенные изменения в базу данных

(завершает текущую транзакцию), заканчивает сеанс вашей работы с ORACLE и вновь выполняет регистрацию с указанным именем пользователя.

Примеры: Чтобы зарегистрироваться с именем SCOTT с паролем TIGER к БД по

умолчанию к DECnet-узлу "corp", введите:

SQL> CONNECT SCOTT/TIGER@d:corp

Чтобы зарегистрироваться с именем SCOTT с последующим запросом пароля, введите:

SQL> CONNECT SCOTT

DISCONNECT (отсоединиться)

Команда DISCONNECT Вносит текущие изменения в базу данных и заканчивает ваш сеанс работы с ORACLE, но не завершает работы с SQL*PLUS.

Используйте DISCONNECT в командном файле, чтобы закрыть пользователю доступ к БД, когда вы хотите отключить пользователя от ORACLE, но чтобы пользователь не покидал SQL*PLUS. Используйте EXIT или QUIT для отключения от ORACLE и передаче управления операционной системе.

Примеры: Ваш командный файл может начинаться командой CONNECT и заканчиваться командой DISCONNECT, как показано ниже:

```
SQL> GET MYFILE
1     CONNECT ...
                .
                .
                .
                .
15* DISCONNECT
```

Создание объектов базы данных

CREATE USER (создать пользователя)

Любой, кто работает с базой данных должен быть зарегистрированным. Для создания пользователя используется директива следующего формата.

```
CREATE USER my_user
IDENTIFIED BY my_password
DEFAULT TABLESPACE my_tablespace
TEMPORARY TABLESPACE temp_ts;
```

При создании имени пользователя с табличной областью по умолчанию любой объект будет создаваться с области по умолчанию.

CREATE CLUSTER (создать кластер)

```
CREATE CLUSTER польз.кластер
столбец тип_данных, столбец тип_данных,...
PCTUSED целое  PCTFREE целое
SIZE целое
INITRANS целое MAXTRANS целое
TABLESPACE область_хранения
STORAGE хранение
```

Оператор CREATE CLUSTER создает кластер для одной или нескольких таблиц.

Кластеризация вынуждает базу данных хранить таблицы с одинаковыми кластерными ключами вместе, что сокращает время доступа к ним. Как правило, имеет смысл объединять в кластер таблицы, над которыми часто производится операция соединения.

'Столбец' -- это имя или имена столбца (столбцов), которые являются кластерными ключами. Тип_данных' не может быть NULL или NOT NULL. Параметр SIZE задает средний размер области, необходимой для хранения всех строк с одинаковыми значениями ключей.

Остальные параметры—PCTUSED, PCTFREE, INITRANS, и MAXTRANS—имеют такое же значение, как и в операторе CREATE TABLE.

CREATE DATABASE (создать базу данных)

Данная команда доступна только в SQL*DBA

(Файл ORAWIN95\BIN\SQLDBA.EXE для PERSONAL ORACLE-7 В меню Windows-95 его нет)

```
CREATE DATABASE база_данных
   CONTROLFILE REUSE --
   LOGFILE файл, файл,.
   MAXLOGFILES целое
DATAFILE файл, файл,...
MAXDATAFILES целое
MAXINSTANCES целое
ARCHIVELOG|NOARCHIVELOG
• подразумевается NOARCHIVELOG
SHARED | EXCLUSIVE
• подразумевается SHARED
```

При указании REUSE Если в момент выполнения оператора база данных с таким же именем уже существует, то все данные в ней будут уничтожены.

Оператор CREATE DATABASE выполняет начальную подготовку базы данных для ее дальнейшего использования.

База_данных - это имя базы данных. Максимальная длина имени базы -- 8 символов. Если имя не указано, то подразумевается имя, заданное параметром DB_NAME в файле INIT.ORA.

CONTROLFILE REUSE

вынуждает повторно использовать файлы, описанные параметром CONTROL_FILES в файле INIT.ORA. При этом стирается вся информация, находящаяся в них в данный момент. Этот параметр ,как правило, не используется при первичном создании базы данных. LOGFILE файл указывает на файлы, используемые для повтора работы. Если параметр опущен, то ORACLE создаст два таких файла. Их имена и размеры зависят от операционной системы.

MAXLOGFILES целое

устанавливает максимально возможное число файлов повтора работы. Это число должно находиться в диапазоне от 2 до 256. Увеличить это число можно только создав базу заново. Указание большего числа не повлияет на работу базы.

Файл -описывает файл. Писать следует так:

'имя_файла' SIZE размер K | M REUSE.

Целое после SIZE определяет размер файла в байтах; Если после размера стоит K, то в килобайтах (размер 1024 байт); если M, то в мегабайтах (размер x 1 048 576 байт). Если размер не указан, то подразумевается 10M для файлов данных и 500K для файлов повтора работы.

DATAFILE файл

определяет один или несколько файлов как файлы для хранения базы данных. Если имена опущены, то ORACLE сам создаст один файл. Его имя зависит от операционной системы.

MAXDATAFILES целое

устанавливает максимально возможное число файлов данных, которое может иметь база. Как правило, это число лежит в пределах от 1 до 255. Это число может быть увеличено только путем повторного создания базы. Излишки, вызванные указанием большого числа, не повлияют на работу базы.

MAXINSTANCES целое

устанавливает максимальное число инстанций, имеющих возможность одновременно открыть базу. Диапазон значений -- 1 - 255.

ARCHIVELOG или NOARCHIVELOG

определяет начальный режим использования файлов повтора работы. Параметр ARCHIVELOG сделает необходимым архивирование файлов перед их повторным использованием. NOARCHIVELOG сделает архивирование необязательным. Изменить режим использования файлов повтора работы у существующей базы можно при помощи оператора ALTER DATABASE.

SHARED или EXCLUSIVE

определяет доступность базы после ее создания. Если использовать слово SHARED, то к базе смогут подключаться несколько инстанций. Если использовать слово EXCLUSIVE, то только одна инстанция сможет получать доступ к базе. Для установки доступности базы при дальнейшей ее эксплуатации используйте команду SQL*DBA STARTUP.

CREATE DB LINK (создать связь с БД)

```
CREATE PUBLIC DATABASE LINK имя_связи
CONNECT TO имя_пользователя
IDENTIFIED BY пароль
USING 'строка_sql*net'
```

Оператор CREATE DATABASE LINK открывает связь между местной базой и пользователем в удаленной базе данных. Чтобы извлечь данные из таблицы в удаленной базе, следует после имени таблицы (в предложении FROM оператора SELECT) добавить @имя_связи.

PUBLIC

делает связь доступной для всех пользователей, кроме тех, кто имеет частную связь с таким же именем. Если в операторе нет слова PUBLIC, то данная связь будет частной—т.е. доступной только создавшему ее пользователю.

имя_пользователя, пароль -

имя и пароль существующего пользователя в удаленной базе. Если эти параметры опущены, то

подразумеваются имя и пароль пользователя, выполняющего оператор.

'строка_sql*net'

описывает доступную через SQL*Net удаленную базу данных.

CREATE INDEX (Создать индекс)

Oracle-7 Автоматически использует индексы ,Однако только из тех ,которые есть в наличии . Автоматически создаются индексы для главного ключа и для уникальных полей таблицы . Для всех остальных требуется издать данную директиву

```
CREATE UNIQUE INDEX   индекс ON  таблица
(столбец ASC|DESC, столбец ASC|DESC,...) | CLUSTER кластер
INITRANS целое
MAXTRANS целое
TABLESPACE область_хранения
STORAGE   хранение PCTFREE = 10 | n NOSORT
```

Оператор CREATE INDEX создает индекс к таблице или кластеру. Индексирование сокращает время доступа к данным, обеспечивая прямое обращение к строкам таблицы. Индекс также может быть использован для обеспечения уникальности значений. Индексировать данные можно максимум по 16 столбцам. Можно создавать несколько индексов к разным сочетаниям столбцов , однако следует помнить, что каждый индекс увеличивает время, затрачиваемое на обновление данных.

UNIQUE

гарантирует, что в таблице никогда не будет строк с одинаковыми значениями во всех индексируемых столбцах. Ключ не обязателен.

индекс, таблица

- имя создаваемого индекса и имя таблицы, к столбцам которой создается индекс.

столбец -

имя столбца таблицы.

ASC или DESC -

параметр, используемый для обеспечения совместимости с DB2. Индексы создаются в восходящем порядке (ASC).

кластер -

имя кластера, к которому создается индекс.

область_хранения -

имя области хранения, в которой будет храниться индекс.

PCTFREE

процент пространства в каждом индексном блоке, оставляемого пустым для дальнейших вставок и обновлений. При указании 0 вставка не допустима

NOSORT

отключает сортировку строк перед созданием индекса (если строки уже отсортированы в восходящем порядке). Употребление этого параметра может значительно ускорить процесс создания индекса. Параметр NOSORT не может применяться при создании кластерного индекса.

Параметры INITRANS и MAXTRANS имеют такое же значение, как и в операторе CREATE TABLE.

CREATE ROLLBACK SEGMENT (создать сегмент отката)

```
CREATE PUBLIC ROLLBACK SEGMENT имя_сегмента_отката
TABLESPACE область_хранения
```

STORAGE хранение

Оператор CREATE ROLLBACK SEGMENT создает один сегмент отката (аннулирования транзакции). Область хранения, к которой добавляется сегмент , должна быть активной (подключенной).

PUBLIC

указывает, что сегмент будет доступен для любой инстанции. Если слово PUBLIC опущено, то сегмент отката будет частным.

имя_сегмента_отката - идентификатор сегмента отката (макс. длина -- 30 символов).

область_хранения -область хранения, в которой создается сегмент. Если область хранения не указана, то будет использована область SYSTEM.

CREATE SEQUENCE (создать последовательность)

Последовательностью называется объект ORACLE, генерирующий неповторяющиеся целые числа. Полученные из последовательности числа очень часто используются в качестве значений для первичных ключей. Чтобы пользоваться не принадлежащей Вам последовательности., нужно иметь на нее право выборки (SELECT).

Доступные последовательности перечисленных в представлениях словаря данных

USER_SEQUENCES и ALL_SEQUENCES. При создании последовательности устанавливаются:

Рост - Числа, создаваемые послед., могут либо возрастать постоянно, либо только до определенного предела, либо, по достижении предела, начинать возрастание заново, с начального значения.

Приращение - Послед. может создавать цепочки как увеличивающихся чисел, так и уменьшающихся. Можно задавать также и приращение значений.

Псевдостолбец последовательность.NEXTVAL содержит новое созданное число, а псевдостолбец последовательность.CURRVAL содержит последнее созданное число.

```
CREATE SEQUENCE польз.имя_последовательности
INCREMENT BY n
START WITH n
MAXVALUE n | NOMAXVALUE
MINVALUE n | NOMINVALUE
CYCLE | NOCYCLE
CACHE 20 | n | NOCACHE
ORDER | NOORDER
```

Оператор CREATE SEQUENCE создает в базе данных объект, при помощи которого пользователи могут генерировать неповторяющиеся целые числа. Производимые таким образом значения могут использоваться в качестве первичных ключей.

Польз - владелец объекта 'последовательность' (подразумевается пользователь, выполняющий оператор CREATE SEQUENCE).

INCREMENT BY n

задает приращение для генерируемых чисел. Если n положительно ,то значения будут возрастать; если n отрицательно -- значения будут уменьшаться. Подразумеваемое значение данного параметра = (+1).

START WITH n

задает 1-ое генерируемое число. Подразумеваемыми 1-ми числами являются: для восходящих последовательностей MINVALUE, для нисходящих MAXVALUE.

MAXVALUE n | NOMAXVALUE

Самое большое число, которое сгенерирует данная последовательность. Подразумеваемое значение = 10e- для восходящих и = 1 для нисходящих последовательностей.

MINVALUE n | NOMINVALUE

Наименьшее число, которое сгенерирует данная последовательность. Подразумеваемое значение: 1 для восходящих последовательностей, 10e2-1 для нисходящих.

CYCLE | NOCYCLE

если задать параметр CYCLE, то после выдачи MAXVALUE восходящая последовательность снова начнет генерацию с MINVALUE (а нисходящая после MINVALUE начнет с MAXVALUE). Если задать NOCYCLE (что подразумевается), то генерация чисел данной последовательностью прекратится после выдачи соответствует максимальному (или минимального) числа.

CACHE n | NOCACHE

параметр CACHE вызывает опережающее размещение генерируемых чисел в памяти, что приводит к увеличению скорости выполнения. Значение n должно быть меньше разности MAXVALUE и MINVALUE. Подразумеваемым значением для CACHE является 20.

ORDER | NOORDER

параметр ORDER гарантирует, что числа будут генерироваться соответственно порядку обращения к последовательности, что очень важно для некоторых приложений. Но ,даже, если задать NOORDER (что подразумевается), числа как правило генерируются в правильном порядке.

При использовании последовательности :

- для выбора следующего номера - использовать вызов типа my_seq.NEXTVAL
- для выбора текущего номера -вызов тип my_seq.CURRVAL
- для выбора в переменную PL/SQL вызов типа

SELECT my_seq.CURRVAL INTO my_per FROM DUAL

- из последовательности my_seq выбирается новое значение в переменную my_per

CREATE SYNONYM (создать синоним)

Синонимами называются дополнительные имена, присваиваемые таблицам, представлениям данных или последовательностям. Синонимы имеет смысл использовать для имен чужих или удаленных таблиц или представлений, что позволяет не указывать каждый раз владельца или БД, в которой расположен данный объект. Синонимы бывают частными или общедоступными.

Это определяется при их создании. Имя нового частного синонима должно отличаться от всех имен объектов, принадлежащих его создателю. Синонимы создаются при помощи оператора CREATE SYNONYM.

```
CREATE PUBLIC SYNONYM польз.имя_синонима
FOR польз.таблица_или_представл_данных @связь_с_БД
```

Оператор CREATE SYNONYM создает синоним для обозначения таблицы или представления данных. Частный синоним должен иметь имя, отличное от имен любых объектов, принадлежащих создающему синоним пользователю.

Только с помощью синонима процедуру или функцию можно сделать доступной для других пользователей и установить для нее параметры PUBLIC (Общие)

PUBLIC

сделает созданный синоним доступным для всех пользователей. Если параметр PUBLIC не задан, то синоним будет частным (т.е. доступным только для пользователя, который его создал).

Польз -

имя пользователя. Если оно не указано, то подразумевается что таблица (или представление данных) принадлежит пользователю, выполняющему оператор.

связь_с_БД -

указывает на существующую связь с удаленной базой. Если конкретный пользователь не указан, то синоним будет создаваться для объекта, принадлежащего пользователю, описанному в связи с удаленной БД.

CREATE TABLE (создать таблицу)

```
CREATE TABLE польз.имя_таблица
( описание_столбца | эл-т_таблицы,
описание_столбца | эл-т_таблицы,...)
PCTFREE n  PCTUSED n
INITRANS n  MAXTRANS n
TABLESPACE область_хранения
STORAGE хранение
CLUSTER кластер столбец, столбец,...
AS запрос
```

Оператор CREATE TABLE создает таблицу в базе данных. Таблица может иметь от 1 до 254 столбцов.(В Personal Oracle-7 до 1000) Созданная таблица будет пуста (если не был указан запрос в предложении AS). Строки данных как правило добавляются в таблицу с помощью оператора INSERT.

Польз -владелец создаваемой таблицы (подразумевается пользователь, выполняющий оператор).

описание_столбца - описание столбцов и возможных ограничений таблицы.

***эл-т_таблицы* -** описание ограничений для таблицы.

PCTFREE n -

определяет процент пространства (n) в каждом блоке таблицы, резервируемого для дальнейших обновлений и вставок данных. Число n должно быть целым от 0 до 100. Подразумеваемое значение n -- 10. n = 100 не имеет смысла, т.к. данные не смогут быть вставлены.

Сочетание значений параметров PCTFREE и PCTUSED определяет, будут ли новые данные записываются в существующие блоки или для них будут созданы дополнительные.

PCTUSED n

задает процент минимального использования пространства в каждом блоке таблицы. n должно быть целым числом в диапазоне от 0 до 100. Подразумеваемое значение -- 40. Блок будет считаться свободным (т.е. готовым к принятию новых данных) если он заполнен меньше, чем на указанное после PCTUSED число процентов. Блок перестает считаться свободным, когда он заполняется данными до предела PCTFREE. Большие значения PCTUSED позволяют более эффективно использовать пространство таблицы (и соответственно дисковое пространство), правда, за счет понижения производительности.

INITRANS n

устанавливает начальное число транзакционных записей в каждом блоке. n должно быть целым числом в диапазоне от 1 до 255 (если опущено, то подразумевается n = 1). При первом использовании блока ORACLE резервирует в его свободной части 23 байта для каждой транзакции. Когда число одновременных транзакций превысит INITTRANS, новые транзакционные записи будут размещаться динамически до тех пор, пока их число не достигнет MAXTRANS или пока не заполнится блок.

MAXTRANS n

устанавливает максимальное число транзакций, имеющих возможность конкурентно модифицировать данные в блоке. n—целое, от 1 до 255. Подразумеваемое значение n = 255.

область_хранения-

область хранения, в которой будет существовать таблица. Если параметр TABLESPACE опущен, то подразумевается область хранения STORAGE.

STORAGE

устанавливает схему распределения пространства для хранения таблицы.

CLUSTER кластер столбец, столбец,...

включает таблицу в указанный кластер. Кластер должен принадлежать пользователю, выполняющему оператор.

Указывайте один столбец таблицы для каждого кластерного ключа.

Первый столбец таблицы должен соответствовать первому кластерному столбцу, и т.д. Имена таблицы и кластерного столбца могут быть разными. Кластеризуемые столбцы как правило являются первичным ключом (или частью первичного ключа) таблицы.

Запрос

есть предложение SELECT.

Столбцы, извлекаемые запросом, должны соответствовать столбцам, перечисленным в операторе CREATE TABLE.

Тип данных и ширина столбцов создаваемой таблицы при этом получаются такими же, как у столбцов, извлекаемых запросом.

Если столбцы в запросе имеют полностью определенные уникальные имена, то имена столбцов после CREATE TABLE можно опустить - они будут унаследованы от запроса.

После создания таблицы в нее загружаются данные, извлеченные запросом.

В запросе нельзя употреблять предложения ORDER BY и FOR UPDATE OF.

Параметры элементы_таблицы, в случае использования запроса, могут содержать только имена столбцов.

Чтобы добавить столбец или ограничение, следует использовать оператор ALTER TABLE...ADD.

Чтобы модифицировать ограничение его нужно сначала удалить (ALTER TABLE.. DROP CONSTRAINT), а затем вновь добавит (ADD).

Пример: Чтобы описать таблицу staff (предполагается, что Вы польз. scott), следует ввести:

```
CREATE TABLE staff (
      empno     NUMBER    NOT NULL   PRIMARY KEY,
      ename     CHAR(20)  NOT NULL   CHECK (ename = UPPER),
      job       CHAR(10),
      mgr       NUMBER    REFERENCES scott.staff(empno),
```

```
        hiredate    DATE          CHECK (hiredate >= SYSDATE - 7),
              sal           NUMBER(10,2) CHECK (sal > 800),
              comm        NUMBER(9,2)  DEFAULT NULL,
              deptno      NOT NULL     REFERENCES scott.dept(deptno)
    ) PCTFREE 5 PCTUSED 75;
```

CREATE TABLESPACE (создать область хранения)

```
CREATE TABLESPACE область_хранения
DATAFILE (файл, файл,...)
DEFAULT STORAGE хранение
ONLINE | OFFLINE
```

Оператор CREATE TABLESPACE создает область хранения, состоящую из указанных файлов, имеющую определенные параметры хранения и состояние ONLINE (подключенное) или OFFLINE (отключенная).

Для выполнения этого оператора необходимо иметь право DBA.

Файл-файл базы данных, описаний след. образом:

'имя_файла SIZE целое K | M REUSE

Параметр REUSE вынудит систему использовать заново файл с таким же именем, предварительно стерев хранящуюся в нем информацию.

Областью хранения (TABLESPACE) в базе данных называется логическая единица, содержащая таблицы, индексы, временные сегменты и сегменты отката. Фактически это аналогия виртуального диска

Также она является логической единицей для процедур создания резервной копии и восстановления. При создании базы данных автоматически создается одна область хранения с именем SYSTEM.

Системные файлы можно добавлять потом по мере надобности. Перед уничтожением (DROP) области хранения ее необходимо перевести в отключенное состояние.

CREATE TABLESPACE tabspace_2

DATAFILE 'C:\User\table_sp.dat' SIZE 20M

DEFAULT STORAGE (INITIAL 10K NEXT 50K

MINEXTENTS 1 MAXEXTENTS 999

PCTINCREASE 10)

ONLINE ;

Рекомендуется создать в области хранения хотя бы один сегмент отката.

CREATE VIEW (создать представление данных)

Представление есть виртуальная таблица, доступ к которой осуществляется так же как к реально существующей таблице.

```
CREATE VIEW польз.имя_предст альт_имя, альт_имя,...
AS запрос
WITH CHECK OPTION CONSTRAINT ограничение
```

Оператор CREATE VIEW создает "логическое окно" (виртуальную таблицу, представление данных) для одной или нескольких таблиц или таких представлений.

Запрос -

есть оператор SELECT (без предложений ORDER BY и FOR UPDATE OF),который определяет, из каких столбцов и строк будет состоять создаваемое представление данных.

WITH CHECK OPTION

указывает, что следу/*ет не допускать вставки и обновления данных через данное представление данных, если они не будут "видны" через него. Это бывает полезно, когда представление данных основывается на других представлениях.

Представление данных можно использовать вместо настоящей в операторах SELECT, INSERT, UPDATE и DELETE, но при этом в операторе SELECT должны присутствовать:

объединение

предложения GROUP BY, CONNECT BY или START WITH

предложение DISTINCT, псевдо-столбцы (например, ROWNUM) или выра-жения в списке столбцов. Можно также обновлять вирт. таблицы, содержащие псевдо столбцы (с помощью оператора UPDATE), если

не пытаться изменить их значения.

```
CREATE VIEW ed AS
SELECT e.empno, e.ename, d.deptno, d.loc
FROM emp e, dept d
WHERE e.deptno = d.deptno;

View created.

SELECT column_name, updatable
FROM user_updatable_columns
WHERE table_name = 'ED';
COLUMN_NAME UPD;

_____ ___
ENAME          YES
DEPTNO         NO
EMPNO          YES
LOC            NO
```

CREATE OR REPLACE PROCEDURE (Создать хранимую процедуру)

Директива создаёт или изменяет процедуру, которая будет хранится на сервере для её последующего использования с помощью директивы EXECUTE <имя процедуры>

В отличии от блоков процедура подвергается компиляции один раз при её создании.

В хранимых процедурах нельзя применять макроподстановки (символы &, &&) для постоянной работы внутри процедуры. Макроподстановка может быть использована для оптимизации процесса генерации различных процедур из одного текста SQL файла генерации процедуры

```
CREATE OR REPLACE PROCEDURE имя_процедуры
(имя_параметра Вид_параметра  Тип_переменной ,  .....)
AS
BEGIN
тело процедуры
END;
```

имя_процедуры - имя, определяющее как возможно сослаться (вызвать) данную процедуру. Имя процедуры не должно содержать пробелы (до 30 алфавитно-цифровых символов). При создании процедуры в файлах SQL не рекомендуется указывать имя_владельца.

Вид параметра - IN - входной, OUT - выходной , IN OUT - входной и выходной.

Тело_процедуры - описание полностью соответствует непоименованному блоку.

CREATE ROLE

Используется для того, чтобы под одним именем объединить ряд привилегий пользователя.

```
CREATE ROLE teller
  IDENTIFIED BY cashflow
```

CREATE PROFILE (Создать профиль)

Профиль - набор ограничений ресурсов базы данных. Если Вы назначаете профиль пользователю, тот пользователь не может превышать эти ограничения.

Вы должны иметь, привилегию CREATE PROFILE

```
CREATE PROFILE профиль
LIMIT
  Параметр значение
          ..................................
```

Наименование параметра	Описание параметра
Профиль	Является именем профиля, который будет создан.
SESSIONS_PER_USER	Ограничивает пользователя целочисленными параллельными сеансами.
CPU_PER_SESSION	Ограничивает время для сеанса.
CPU_PER_CALL	Ограничивает время обращения (синтаксический анализ, выполняться, или выборка).
CONNECT_TIME	Ограничивает общее время сеанса. Это значение выражено в минутах.
IDLE_TIME	Ограничивает периоды непрерывного неактивного времени в течение сеанса. Это значение выражено в минутах. Долго текущие запросы и другие операции не подчинены этому ограничению.
LOGICAL_READS_PER_SESSION	Ограничивает номер чтения блоков данных в сеансе, включая чтение блоков из памяти и диска, на целочисленные блоки.
LOGICAL_READS_PER_CALL	Ограничивает номер чтения блоков данных для обращения,
PRIVATE_SGA	Ограничивает количество личного пространства,
COMPOSITE_LIMIT	Ограничивает общую стоимость ресурса для сеанса. Вы должны выразить значение этого параметра в сервисных модулях. Oracle7 вычисляет общую стоимость ресурса как сумму следующих ресурсов: CPU_PER_SESSION CONNECT_TIME LOGICAL_READS_PER_SESSION PRIVATE_SGA
UNLIMITED	Указывает, что пользователь назначил этот профиль, может использовать неограниченное количество этого ресурса.
DEFAULT	Опускает ограничение для этого ресурса в этом профиле. Пользователь назначил этот профиль, подчинен ограничению для этого ресурса, определенного в ЗАДАННОМ ПО УМОЛЧАНИЮ профиле.

Пример

```
CREATE PROFILE system_manager
   LIMIT   SESSIONS_PER_USER
             UNLIMITED
              CPU_PER_SESSION
              UNLIMITED
               CPU_PER_CALL 3000
CONNECT_TIME  45
LOGICAL_READS_PER_SESSION
DEFAULT
LOGICAL_READS_PER_CALL 1000
PRIVATE SGA 15K
COMPOSITE_LIMIT 5000000
```

Изменения обьектов

ALTER CLUSTER (изменить кластер)

кластер - это определённая область данных, используемая RDBMS чтобы организовать специфическое хранение данных. Используется чтобы ускорить процесс поиска и обработки данных.

```
SQL> ALTER CLUSTER польз.кластер
•    польз. и фразы ниже необязательны
PCTUSED целое—мин. используемое пространство в блоке (40)
   PCTFREE целое    --% пространства в блоке для изменений  (10)
   SIZE целое              --число ключей, хранимых в кластере;
   INITRANS целое       --начальное число элементов транзакций;
   MAXTRANS целое    --макс. число одновременных транзакций;
   STORAGE предложение      --управляет распределением памяти;
```

Оператор ALTER CLUSTER переопределяет порядок последующего хранения кластера. Вы можете изменить любой из указанных параметров, но Вы не можете сделать следующее:

- изменить кол-во или имена столбцов в кластерном ключе;
- изменить параметр MINEXTENTS;

- изменить какие-либо параметры, относящиеся к уже существующему блоку;
- удалить таблицы из кластера (для этого надо использовать операторы DROP CLUSTER и DROP TABLE);
- изменить область размещения кластера;

ALTER DATABASE (модифицировать базу данных)

```
SQL> ALTER DATABASE
база_данных ADD LOGFILE 'имя_файла'
SIZE целое K | M REUSE,...
DROP LOGFILE  'имя_файла', 'имя_файла',...
RENAME FILE 'имя_файла', 'имя_файла',...
TO 'имя_файла', 'имя_файла',...  ARCHIVELOG |
NOARCHIVELOG MOUNT  SHARED | EXCLUSIVE
DISMOUNT OPEN | CLOSE NORMAL | IMMEDIATE
```

Оператор ALTER DATABASE используется для того, чтобы:

- Смонтировать базу данных в разделенном или исключительном режиме;
- Открыть или закрыть базу данных;
- Добавить, удалить или переименовать файл повтора работы.

Также можно сделать файл повтора работы архивируемым (полезно для восстановления носителя информации) или не архивируемым (пригодно только для восстановления инстанции). Если имя базы данных опущено, то подразумевается имя базы данных, указанное в текущем файле INIT.ORA (параметр DB_NAME).

ALTER INDEX (модифицировать индекс)

```
SQL>  ALTER INDEX польз.индекс
INITRANS целое
MAXTRANS целое
STORAGE предложение
```

Оператор ALTER INDEX переопределяет порядок последующего хранения индекса. Параметры INITRANS, MAXTRANS, и STORAGE имеют то же значение, что и в операторе CREATE TABLE (создать таблицу).

ALTER ROLLBACK SEGMENT (модифицировать сегмент отката)

```
SQL> ALTER PUBLIC ROLLBACK SEGMENT сегмент
STORAGE предложение
```

Оператор ALTER ROLLBACK SEGMENT позволяет сделать сегмент отката (сегмент аннулирования транзакции) доступным всем пользователям (PUBLIC), а также изменить параметры, управляющие схемой его хранения. Чтобы сделать сегмент общедоступным, используйте слово PUBLIC. Чтобы сделать сегмент частным, надо уничтожить его (DROP) и создать заново (CREATE).

Предложение STORAGE влияет на дальнейшее распределение пространства в сегменте. При изменении существующего сегмента нельзя использовать параметры INITIAL и MINEXTENTS.

ALTER SEQUENCE (изменить последовательность)

```
SQL>ALTER SEQUENCE польз.последовательность
INCREMENT BY целое
MAXVALUE целое | NOMAXVALUE
MINVALUE целое | NOMINVALUE
CYCLE | NOCYCLE
CACHE целое | NOCACHE
ORDER | NOORDER
```

Оператор ALTER SEQUENCE предназначен для:

- изменения приращения при генерации последовательности чисел;
- переустановки или удаления параметров MINVALUE или MAXVALUE;
- включения или выключения характеристик CACHE и ORDER.
-

Значение CACHE должно быть меньше разности между MINVALUE и MAXVALUE.

Если Вы используете параметр CYCLE, то восходящая последовательность после достижения MAXVALUE выдаст MINVALUE, а нисходящая—после достижения MINVALUE выдаст MAXVALUE.

Чтобы начать генерацию последовательности с нового числа надо удалить (DROP), а затем заново создать (CREATE) последовательность.

ALTER TABLE (модифицировать таблицу)

```
ALTER TABLE польз.таблица
ADDописание_столбца | ограничение_таблицы,
описание_столбца | ограничение_для_таблицы,...
MODIFY описание_столбца, описание_столбца,...
DROP CONSTRAINT ограничение, ограничение,...
PCTFREE целое  PCTUSED целое
INITRANS целое  MAXTRANS целое
STORAGE предложение
BACKUP
```

Оператор ALTER TABLE позволяет добавлять в таблицу столбцы и ограничения, изменять размер и тип данных существующих столбцов, изменять установки NULL/NOT NULL, удалять ограничения, изменять схему хранения таблицы и заносить в словарь данных информацию о том, что была создана резервная копия таблицы (т.е. был выполнен выполнен BACKUP).

ALTER TABLESPACE (модифицировать область *хранения)*

```
ALTER TABLESPACE область_хранения
ADD DATAFILE ‘имя_файла’ SIZE целое K | M
REUSE,...
RENAME DATAFILE ‘имя_файла’, ‘имя_файла’,...
TO ‘имя_файла’, ‘имя_файла’,...
DEFAULT STORAGE предложение
ONLINE | OFFLINE NORMAL | IMMEDIATE
BEGIN BACKUP | END BACKUP
```

Оператор ALTER TABLESPACE позволяет добавлять и переименовывать файлы базы данных, изменять подразумеваемые характеристики области хранения, подключать и отключать области хранения и запускать или останавливать запись резервной копии (BACKUP).

Для того, чтобы переименовать один или более файлов данных, нужно проделать следующее:

- Отключить область хранения;
- Переименовать файлы при помощи средств операционной системы;
- Переименовать файлы базы данных оператором ALTER TABLESPACE;
- Подключить область хранения.

ALTER ROLE(Изменить пароль роли)

Обеспечивает изменение пароля роли для существующей роли

```
ALTER ROLE teller
  IDENTIFIED BY letter
```

ALTER USER (модифицировать пользователя)

```
SQL>ALTER USER имя_пользователя
IDENTIFIED BY пароль
DEFAULT TABLESPACE область_хранения
TEMPORARY TABLESPACE область_хранения
```

При помощи оператора ALTER USER можно изменить пароль пользователя, изменить подразумеваемую область хранения для создаваемых им объектов или область хранения для создаваемых им непостоянных сегментов.

Пароль пользователя также можно изменить при помощи оператора GRANT.

Подразумеваемая область хранения устанавливается и в случае предоставления пользователю права RESOURCE для какой-то области хранения, если это 1-я область, на которую пользователь получает такое право.

Подразумеваемой областью для непостоянных объектов также становиться 1-я предоставленная пользователю область хранения.

Удаления объектов

DROP CLUSTER (Удалить кластер)

```
DROP CLUSTER польз.кластер
INCLUDING TABLES
```

Оператор DROP CLUSTER удаляет указанный кластер из базы данных. Уничтожение кластера вызывает также уничтожение индекса кластера и освобождение занимаемого пространства в области хранения. Чтобы уничтожить кластер, созданный другим пользователем, Вы должны иметь права DBA.

INCLUDING TABLES

вызывает уничтожение всех таблиц, входящих в кластер. Если эта фраза опущена, то таблицы должны быть удалены до уничтожения кластера.

Удаление кластеризованной таблицы исключает ее из кластера. Таблицу нельзя просто “раскластеризовать”. Для этого нужно создать такую же таблицу, только без использования параметра CLUSTER. Например:

```
CREATE TABLE new
AS SELECT * FROM old
Затем уничтожьте старую таблицу:  DROP TABLE old
и переименуйте новую:  RENAME new TO old
DROP DATABASE LINK            (уничтожить связь с DB) SQL
DROP PUBLIC DATABASE LINK имя_связи
```

Оператор DROP DATABASE LINK уничтожает указанную связь с БД.

Слово PUBLIC нужно использовать, если связь доступна для всех пользователей.

Чтобы уничтожить общедоступную связь или связь, принадлежащую другому пользователю, необходимо иметь права DBA.

DROP INDEX (уничтожить индекс)

```
DROP INDEX польз.индекс
```

Оператор DROP INDEX удаляет указанный индекс из базы данных. Чтобы сделать это, Вы должны либо быть владельцем индекса, либо иметь права DBA.

Если имя польз. опущено, то подразумевается пользователь, выполняющий оператор. После уничтожения индекса освобождается все пространство, занимаемое им в области хранения.

DROP ROLLBACK SEGMENT (уничтожить сегмент отката)

```
DROP PUBLIC ROLLBACK SEGMENT сегмент
```

Оператор DROP ROLLBACK SEGMENT уничтожает указанный сегмент отката. Слово PUBLIC нужно использовать, если сегмент создан как PUBLIC (общедоступный).

Уничтожить сегмент отката может только пользователь с правами DBA. Уничтожать можно только неиспользуемые сегменты.

После уничтожения сегмента освобождается все занимаемое им пространство в области хранения.

DROP SEQUENCE (уничтожить последовательность)

```
DROP SEQUENCE польз.последовательность
```

Оператор DROP SEQUENCE удаляет из базы данных указанную последовательность.

Чтобы удалить последовательность, Вы должны либо быть ее владельцем, либо иметь права DBA.

Если имя польз. опущено, то подразумевается имя пользователя, выполняющего оператор.

Единственный способ “перезапустить” последовательность - это уничтожить и заново создать ее. К примеру: предположим, что принадлежащая Вам последовательность POWERS находится на числе 288, а Вы хотите заново начать генерацию чисел с 12. Тогда нужно выполнить следующее:

```
DROP SEQUENCE powers
CREATE SEQUENCE powers . .
START WITH 12
```

DROP SYNONYM (уничтожить синоним)

DROP PUBLIC SYNONYM польз.синоним

Оператор DROP SYNONYM удаляет из базы данных указанный синоним.

Слово PUBLIC нужно употреблять, если синоним доступен всем пользователям. Чтобы уничтожить общедоступный или чужой синоним, необходимо иметь права DBA.

Для модификации синонима его следует уничтожить и создать заново.

DROP TABLE (уничтожить таблицу)

DROP TABLE польз.таблица

Оператор DROP TABLE удаляет из базы данных указанную таблицу и всю хранящуюся в ней информацию.

Чтобы уничтожить ,таблицу Вы должны быть либо ее владельцем, либо иметь права DBA.

Все соответствующие индексы уничтожаются вместе с таблицей.

Представления данных и синонимы, ссылающиеся на уничтоженную таблицу, остаются, но становятся ошибочными. Их нужно или тоже уничтожить, или откорректировать 0 так, чтобы они стали действительными.

Блоки в соответствующей области хранения, которые занимали данные и индексы таблицы, освобождаются (если таблица не была кластеризована).

Оператор DROP CLUSTER INCLUDING TABLES уничтожает все таблицы, входящие в кластер.

Небесполезно перед уничтожением таблицы просмотреть представление. данных USER_CROSS_REFS в словаре данных, чтобы найти все ссылки на подлежащую уничтожению таблицу.

DROP TABLESPACE (уничтожить область хранения)

DROP TABLESPACE область_хранения INCLUDING CONTENTS

Оператор DROP TABLESPACE удаляет из базы данных указанную область хранения. Область хранения по имени SYSTEM не может быть удалена. Удалить область хранения может только пользователь с правами DBA.

INCLUDING CONTENTS

уничтожает все объекты, находящиеся в указанной области, включая таблицы, кластеры, сегменты отката и временные сегменты.

Если фраза INCLUDING CONTENTS опущена, то все объекты в области должны быть предварительно уничтожены.

Перед удалением области хранения ее нужно перевести в отключенное состояние.

Область хранения не может быть уничтожена, когда пользователи работают с находящимися в ней данными, индексами, сегментами отката или временными сегментами.

DROP VIEW (уничтожить представление данных)

DROP VIEW польз.предст_данных

Оператор DROP VIEW удаляет из базы данных указанное представление данных (виртуальную таблицу). Чтобы уничтожить представление данных нужно быть либо его владельцем, либо иметь права DBA.

После уничтожения представления данных все ссылающиеся на него представления и синонимы остаются, но становятся недействительными. Их нужно либо удалить, либо откорректировать так, чтобы они стали действительными.

Чтобы изменить представление данных, его следует уничтожить и создать заново. Альтернативный вариант - использование слов CREATE OR REPLACE

Перед уничтожением представления необходимо просмотреть представлении данных USER_CROSS_REFS в словаре данных, чтобы найти все ссылки на подлежащее уничтожению представление.

Манипуляция данными

SELECT (Выбрать)

SELECT ALL | DISTINCT * | таблица.* | выражение alias,...

```
FROM польз.таблица, таблица alias,...
WHERE условие
CONNECT BY условие  START WITH условие
GROUP BY выражение, выражение... HAVING условие
UNION | INTERSECT | MINUS  SELECT...
ORDER BY выражение | ASC | DESC,...
FOR UPDATE OF столбец, столбец... NOWAIT;
```

Оператор SELECT извлекает данные из столбцов одной или нескольких таблиц. Оператор SELECT сам по себе является запросом. Если он используется как предложение внутри другого оператора, то он называется подзапросом.

В операторе SELECT обязательно должно присутствовать предложение FROM.

Остальные предложения не являются необходимыми. При совместном использовании нескольких предложений они должны быть записаны в указанном выше порядке.

SELECT INTO(выбрать и поместить в переменную)

```
SELECT_перечень INTO список_переменных
FROM таблица, таблица,...
остальная_часть_оператора;
```

Оператор SELECT...INTO извлекает из таблицы (таблиц) значения указанных столбцов и помещает их в соотв. переменные, перечисленные в списке_переменных. Типы данных переменных и соотв. столбцов должны либо совпадать, либо быть совместимыми.

Извлеченную строку данных можно хранить в переменной, объявленной с помощью атрибута %ROWTYPE.

Список SELECT(выбор по полям)

```
SELECT list
FROM...
```

Список оператора SELECT состоит из одной или нескольких единиц информации, расположенных между словами SELECT и FROM и отделенных друг от друга запятыми. Порядок элементов в этом списке определяет порядок расположения столбцов в результате. Элементами списка SELECT могут быть:

- Имена столбцов и их альтернативные имена (ename “Фамилия”, deptno)
- выражения со столбцами и ограничениями, или переменные включающего языка
- Групповые функции (MAX(sal), MIN(comm))
- Негрупповые функции (SYSDATE, INITCAP(ename))

Если в оператор SELECT используется предложение GROUP BY, то элементами списка SELECT могут быть только групповые функции, выражения, стоящие после предложения GROUP BY, константы и функции без параметров (типа SYSDATE).

Соотносящийся подзапрос

Соотносящийся подзапрос вычисляет запрос для КАЖДОЙ СТРОКИ запроса-родителя. Для ясности рекомендуется использовать альтернативные имена таблиц или столбцов.

```
SELECT список_столбцов  FROM таблица1 алиас1
WHERE выраж оператор
(SELECT список_столбцов
FROM таблица2 алиас2
WHERE алиас1.столбец оператор алиас2.столбец);
UPDATE список_столбцов SET столбец =
(SELECT выражение
FROM таблица2 алиас2
WHERE алиас1.столбец = алиас2.столбец);

DELETE FROM таблица1 алиас1
WHERE столбец оператор
(SELECT выражение
FROM таблица2 alias2
WHERE алиас1.столбец = алиас2.столбец);
```

Соединение

```
SELECT столбец, таблица.столбец, ..—‘таблица.’ необязательна
```

```
FROM таблица1, таблица2, ...
WHERE таблица1.столбец1 = таблица2.столбец1
```

- предложение WHERE необязательно

Операция соединения позволяет извлекать данные и комбинировать строки из двух или более таблиц одновременно. В результате каждая извлеченная строка может содержать данные из разных таблиц.

Предложение WHERE задает порядок составления строк.

Если извлекаемый столбец существует в нескольких соединяемых таблицах, то перед ним необходимо указывать имя таблицы. Например, столбец sal таблицы emp: emp.sal

Простое соединение позволяет извлечь строки из двух таблиц. Извлекаться будут только те строки, у которых значения столбцов отвечают условию в предложении WHERE. Если предложения WHERE нет, то будут извлечены все строки. К примеру, соединив без WHERE 2 таблицы по 300 строк каждая, мы получим 90 000 строк—вряд ли это то, что нам нужно.

INSERT (добавить)

```
INSERT INTO имя_таблицы/представления (столбец, столбец,...)
VALUES(значение, значение,...);   --список столбцов необязателен
INSERT INTO имя_таблицы/представления (столбец, столбец,...) SELECT...;
```

Оператор INSERT предназначен для добавления строк в таблицу или представление данных. Имена в списке столбцов могут быть перечислены в любом порядке. В столбцы, не указанные в списке, заносится пустое значение.

Все столбцы с признаком NOT NULL должны быть указаны и иметь предназначающиеся для них значения.

В предложении VALUES перечисляются конкретные значения столбцов в добавляемой строке.

Каждый указанный столбец должен иметь соответствующего. ему значение в предложении VALUES. Типы данных значения и столбца должен быть совместимы или преобразуемы.

Значения типа CHAR и DATE надо заключать в одинарные кавычки (‘абв’).

Чтобы добавить строки из другой таблицы ,следует использовать подзапрос. Оператор SELECT в этом подзапросе должен извлекать значения для каждого перечисленного столбца.

UPDATE (обновить)

```
UPDATE таблица/имя_предст_данных SET
столбец_имя = выражение
столбец_имя = (SELECT_с_одним_результатом) --допускается в SQL
(столбец_имя, столбец_имя,...) = оператор_SELECT—допускается в SQL
WHERE_предложение;
WHERE CURRENT OF имя_курсора; --допускается в PL/SQL
```

Оператор UPDATE заменяет значения одного или нескольких указанных столбцов на значения выражений или результат запроса.

Оператор SELECT в этом запросе должен возвращать как минимум одну строку и обеспечивать значения для каждого столбца, стоящего слева от знака =. Этот оператор SELECT не может содержать фразы INTO. Для определения набора строк, подлежащих обновлению, используется предложение WHERE. В нем указываются условия, которым должна отвечать обновляемая строка. Если предложение WHERE опустить, то будут обновлены все строки.

В PL/SQL, предложение WHERE CURRENT OF вызывает обновление текущей строки курсора. (Данное выражение допускается только в PL/SQL)

GROUP BY (и HAVING)- группировка

```
SELECT cgbcjr...            --требуется только одно выражение
  FROM таблица, таблица...      -- треб. только одна таблица
  WHERE условие                 -- необязательно
GROUP BY выражение, выражение,...
```

- требуется только одно выражение

HAVING условие—необязательно

При использовании предложения GROUP BY оператор SELECT вычисляет на основе каждой выбранной группы строк одну итоговую строку. Каждый элемент в списке после SELECT должен:

- быть константой или функцией без параметров (типа SYSDATE); либо
- быть элементом, содержащим групповую функцию (типа SUM, COUNT, или MAX);
- соответствовать выражению в GROUP BY.

Условие HAVING определяет, какие группы, выбранные по GROUP BY, попадут в результат.

Оператор LIKE

```
SELECT...
WHERE симв LIKE 'образец'
   ...
```

Оператор SQL LIKE сравнивает строку 'симв' с образцом. В образце можно использовать метасимволы % и _. % (знак процента) соотв. любым символам или пустой строке _ (подчеркивание) соотв. любому одиночному символу.

Символьные литералы должны заключаться в апострофы ('ABC'). При сравнении учитывается регистр букв (прописные/строчные).

Выражение.	.вызовет извлечение	но не...
lname LIKE 'S_ile%'	Smile, Stiles, Skilerson	smile, Samiler
title LIKE '%ager'	Manager, dowager, pager	MANAGER, agent

Распределенный запрос

Для выполнения распределенного запроса необходимо

наличие связи между RDBMS, расположенными на разных компьютерах , через SQL*Net.

Синтаксис распределенного запроса такой же, как и у обычного, только после имени таблицы надо указывать имя связи с удаленной БД (или его синоним).

```
SELECT столбец2, столбец3,...
FROM удаленная_таблица@связь_с_БД, местная_таблица
WHERE удаленная_таблица.столбец1 = местная_таблица.столбец_1;
```

Ссылки на удаленные таблицы возможны в предложении FROM операторов SELECT и INSERT INTO...SELECT. Также, они допустимы в префиксах для ссылки на столбцы таблиц в этих операторах.

FROM (из)

Оператор FROM указывает из каких таблиц или представлений следует извлекать данные. Возможно указание нескольких таблиц через запятые.

```
SELECT список
FROM таблица WHERE...          --для SELECT требуется FROM
   ...

DELETE FROM таблица WHERE..—FROM и WHERE необязательны
   ...

FROM польз.таблица альт_имя @связь_с_БД,..—параметр в SELECT
...                                  и DELETE
```

Предложение FROM определяет, из какой таблицы или таблиц надо извлекать записи (оператор SELECT), или в какой таблице надо стирать строки (оператор DELETE).

Если таблица принадлежит не Вам, то следует указать имя ее владельца. связь_с_БД используется для обращения к таблицам, находящимся в удаленных БД.

ORDER BY (упорядочив по)

```
SELECT ...                  SELECT ...
   ORDER BY выражение ASC,...     ORDER BY выражение DESC,...
   ORDER BY позиция ASC,...       ORDER BY позиция DESC,...
```

Предложение ORDER BY определяет, как при выводе будут упорядочены данные, извлеченные запросом. 'выражение' может быть одним или несколькими столбцами из списка SELECT, или из предложения FROM. 'позиция' может быть порядковым номером столбца в предложении SELECT. ASC и

DESC указывают порядок сортировки:

- ASC—восходящий,
- DESC—нисходящий.

Подразумеваемый порядок—восходящий.

Предложение ORDER BY нельзя использовать в подзапросах операторов

INSERT, UPDATE, CREATE TABLE и CREATE VIEW.

Предложение ORDER BY отменяет действие предложения CONNECT BY, если они используются в одном операторе.

START WITH

Условие в предложении START WITH определяет “корень” (“корни”) дерева.

Чтобы начинать с каждой строки, отвечающей условию WHERE, следует опустить предложение START WITH.

```
SELECT list  FROM...
WHERE...
CONNECT BY PRIOR выраж1 оператор выраж2
                          --сначала вывести выраж1
   CONNECT BY выраж1 оператор PRIOR выраж2
                          --сначала вывести выраж2
   START WITH условие;                --необязательно
```

CONNECT BY (Соединитесь)

```
SELECT список  FROM... WHERE...
CONNECT BY PRIOR выраж1 оператор выраж2
снач. вывести выраж1
CONNECT BY выраж1 оператор PRIOR выраж2
снач. вывести выраж2
START WITH условие; --не обязательно
```

Предложение CONNECT BY связывает строки в древовидную структуру, определенную предложением PRIOR. Предложение PRIOR ставится перед порождающим выражением.

```
CONNECT BY PRIOR EMPNO = MGR
```

Предложение CONNECT BY нельзя использовать в подзапросах и в операциях соединения.

Операторы обработки множеств

Операторы обработки множеств объединяют результаты нескольких запросов в один.

ОПЕРАТОР	ДЕЙСТВИЕ В SQL	СИНТАКСИС
UNION	Объединяет неповторяющиеся	SELECT...
	строки, извлеченные ЛЮБЫМ	UNION SELECT...
	из запросов.	
INTERSECT	Сочетает запросы так, чтобы	SELECT...
	извлечь все неповторяющиеся	INTERSECT SELECT...
	строки, возвращаемые	
	КАЖДЫМ запросом.	
MINUS	Выдает все неповторяющиеся	SELECT...
	строки, извлеченные ПЕРВЫМ,	MINUS SELECT...
	а не вторым, запросом.	

Подзапрос

Подзапросом называется оператор SELECT, стоящий в правой части выражения после слова WHERE. Такой запрос является вложенным в другой. Подзапросы можно помещать в другие подзапросы

.

```
SELECT...WHERE выражение оператор (SELECT...);
UPDATE...WHERE выражение оператор (SELECT...);
DELETE...WHERE выражение оператор (SELECT...);
```

```
SELECT ename FROM emp WHERE cityname IN
(SELECT id FROM city WHERE stateabbr IN
(SELECT stateabbr FROM state WHERE
name = 'NEW YORK'));
```

В SQL можно использовать операторы сравнения с результатом. запроса, извлекающего более чем одно значение. Для этого надо после оператора сравнения использовать слова ANY, ALL или SOME.

```
SELECT ename FROM emp
WHERE job <> 'MANAGER' AND sal > ANY
(SELECT sal FROM emp WHERE job='MANAGER');NEXTVAL (следующее значение)
```

Псевдостолбец последовательность.NEXTVAL предназначен для получения следующего числа из последовательности, в то время как псевдостолбец последовательность.CURRVAL служит для получения последнего числа, сгенерированного последовательностью. Если

Вы используете NEXTVAL и CURRVAL в одном операторе SELECT, то вы все равно получите число NEXTVAL. Для выбора номера из последовательности в PL/SQL блоке необходимо использовать оператор

```
SELECT последовательность.NEXTVAL
INTO ИмяВременнойПеременной
FROM DUAL;
```

Древовидный запрос(предложение CONNECT BY)

Древовидным запросом называется запрос, в котором присутствует предложение CONNECT BY, предназначенное для отображения строк результата в определенном иерархическом порядке. Начиная с корня, описанного предложением START WITH, запрос просматривает каждую соединенную с корнем ветвь. Например: список зав. отделами (из таблицы EMP) и их подчиненных (директор—KING).

WHERE (где, условие отбора)

SELECT ... UPDATE ... DELETE ...
WHERE условие WHERE условие WHERE условие

На основе указанных условий, предложение WHERE решает, какие строки в таблице будут обработаны оператором (SELECT, UPDATE или DELETE), в котором это предложение записано.

Каждое условие состоит из имени столбца, оператора сравнения и некоторого критерия оценки, имеющего тот же тип данных, что и оцениваемый столбец. В условиях можно использовать и логические операторы. Оператор NOT превратит результат оценки в обратный; AND позволяет указывать несколько обязательных условий; OR требует выполнения одного из двух условий. Например, если нужно найти клиентов, имеющих задолженность больше $100, просроченную на 90 или более дней, то можно выполнить след. запрос:

```
SELECT * FROM clients WHERE bal > 100 AND SYSDATE - duedate >= 90;
```

PL/SQL

Комментарии (SQL и PL/SQL)

Комментарии—это поясняющие замечания, вставляемые в текст программы. Комментарии служат для более понятного восприятия программы

Они никак не влияют на ход выполнения программы. В PL/SQL можно отметить начало комментария двумя минусами (--). Такой комментарий оканчивается вместе со строкой. Например:

```
x NUMBER := 0; -- иниц. локальной переменной x
BEGIN
FOR i IN 1..4 LOOP
x := x+1;          --прирастить x
```

В SQL (и в PL/SQL тоже) можно начать комментарий символами /* и продолжить его на сколько угодно строк. Такой комментарий заканчивается символами */. Эти цепочки (/* и */) можно использовать для временного исключения частей текста программы.

```
SELECT ename, sal, mgr
/* Извлечь имена, оклады $2000 или больше. */
FROM emp
WHERE sal >= 2000;
```

SQL в PL/SQL

Язык SQL является непроцедурным языком системы управления реляционной базой данных, который может обрабатывать записи группами и обеспечивает автоматический поиск нужных данных. Язык SQL дает возможность указывать, какие данные необходимо получить, не заботясь о том, как найти эти данные.

Язык PL/SQL, в свою очередь, является процедурным языком, способным управлять ходом вычислений.

В языке PL/SQL есть условные операторы, циклы, средства для работы с курсорами ORACLE и обработки особых ситуаций.

Операторы - В блоках PL/SQL, вдобавок к установкам переменных PL/SQL (var := var + 1), операторам управления ходом выполнения программы (IF, GOTO, EXIT) и циклам (LOOP, FOR...LOOP, WHILE...LOOP), можно использовать любые операторы SQL, относящиеся к категориям обработки данных и управления транзакциями.

Функции - Внутри блока PL/SQL оператор SQL может обращаться к любой функции языка SQL. Операторы PL/SQL внутри блока могут использовать любые функции языка SQL, кроме групповых.

Предикаты - В PL/SQL можно использовать все условия, пригодные для применения в предложение WHERE и любые операторы сравнения.

Переменные

Переменные могут использоваться для хранения результатов запроса, либо для вычисления значений, заносимых в таблицы. Переменные могут иметь любой допустимый в SQL или PL/SQL тип данных: NUMBER, CHAR, DATE или BOOLEAN. Например: balance NUMBER(5,2); wkphone CHAR(12); sex BOOLEAN

Существует два способа присвоения значений переменным. Это:

1) оператор присваивания :=

например, discount := price * 0.15)

2) Выполнение SELECT INTO или FETCH INTO:

например:

```
SELECT price * 0.15 INTO discount FROM pricelist
WHERE stockdate <= SYSDATE - 60
```

Чтобы определить константу, следует после имени написать CONSTANT и сразу присвоить значение. Например:

```
discount CONSTANT NUMBER(2,2) := 0.15;
```

При объявлении переменных как параметры процедур и функций нельзя указывать размер данных.

WHENEVER SQLERROR (в случае ошибки SQL)

```
WHENEVER SQLERROR {EXIT [SUCCESS|FAILURE|WARNING|n|переменная]|CONTINUE}
```

Оператор WHENEVER SQLERROR указывает, что в случае возникновения ошибки при выполнении команды SQL или PL/SQL, следует завершить работу SQL*Plus.

EXIT [SUCCESS | FAILURE | WARNING | n | переменная]

вынудит SQL*Plus, в случае возникновения ошибки SQL, завершить работу, предварительно подтвердив все неподтвержденные изменения в БД. Ошибки команд SQL*Plus не будут вызывать завершение работы.

Правила видимости

Идентификаторы (переменные, константы, записи, курсоры и особые ситуации) являются локальными для блока, в котором они объявлены, и глобальными для всех блоков, находящихся внутри такового.

Если глобальный идентификатор заново объявить в одном из вложенных блоков, то это новое объявление заменит глобальное и станет действительным для этого блока и для вложенных в него других блоков.

В блоке нельзя ссылаться на идентификаторы, объявленные в других блоках,расположенных на том же уровне вложенности.

Если идентификатор переобъявлен в блоке и нужно использовать его описание из внешнего блока, то во внешнем блоке следует перед объявлением идентификатора. написать <<имя_метки>>, и затем перед именем идентификатора. писать имя метки с точкой.

```
<<top>>
DECLARE
      day  DATE;
BEGIN
                ...
DECLARE
      day  DATE;
BEGIN
IF day = top.day THEN...
```

Непоименованные блоки

Непоименованный блок - программа PL/SQL которая пройдёт компиляцию и немедленно выполнится.

Блоки можно “вкладывать” блоки друг в друга.

```
DECLARE           --Основной блок
имя_переменной1  тип_данных (выражение);
  ..              -- объявляться могут:
  ...               переменные, константы,
  ...               курсоры и особые состояния.
  ...               Секция DECLARE необязательна.
BEGIN
      DECLARE       --1-й подблок
          имя_переменной2   (выражение)...;
BEGIN
выполняемые_операторы...;
EXCEPTION
обработчики_исключ_сост;
• Секция EXCEPTION необязательна.
END;
DECLARE         --2-й подблок
имя_переменной3  тип_данных (выражение)...;
BEGIN
• выполняемые_операторы...;
END;
EXCEPTION
обработчики_исключ_сост;
END;
```

Выполнение блоков PL/SQL

Блок на языке PL/SQL начинается либо с операторов DECLARE или BEGIN, либо с имени блока.

Операторы блока вводятся так же, как и команды SQL, за исключением того, что точка с запятой (;) и пустые строки не завершают ввод блока.

Ввод блока PL/SQL нужно заканчивать точкой (.), стоящей на отдельной строке. Чтобы выполнить блок, хранящийся в активном буфере, надо ввести команду RUN или / (наклонная черта).

Например, Вы можете ввести и выполнить следующий блок PL/SQL:

```
SQL> DECLARE
2    x  NUMBER := 100;
3    BEGIN
4    FOR i IN 1..10 LOOP
5       IF TRUNC(i,2) = i / 2 THEN    --i = четное число
6...       .INSERT INTO temp VALUES (i, x, ‘i—четное число’);
7....    ELSE
8....       INSERT INTO temp VALUES (i, x, ‘i—нечетное число’);
9....   END IF;
10...  x := x + 100;
11....END LOOP;
12....END;
13.....


SQL> Процедура PL/SQL успешно завершена.
```

При исполнении блока команды SQL могут вести себя чуть-чуть иначе, чем если бы они исполнялись не в блоке.

DECLARE (объявить переменные)

```
DECLARE
```

```
переменная1  таблица.столбец%TYPE
переменная2  переменная_или_константа%TYPE
```

Атрибут %TYPE используется для того, чтобы сделать тип данных какой-либо переменной таким же, как у другой переменной, константы или столбца таблицы. Выигрыш от этого такой: не нужно точно знать тип данных или параметры столбца на который Вы ссылаетесь, а если определение столбца изменится, то тип данных переменной изменится так же.

Атрибут %ROWTYPE позволяет объявить переменную для хранения данных одной строки таблицы или представления данных, либо для хранения строки, выбранной оператором FETCH. Ссылаться на поля переменной, объявленной при помощи %ROWTYPE, следует так: имя_записи.имя_столбца.

```
DECLARE
   CURSOR nextclient
      IS SELECT fname, lname, phone
            FROM clients;
    callrecord nextclient%ROWTYPE;...
BEGIN
OPEN nextclient;
FETCH nextclient INTO callrecord;
.............
END;
```

BEGIN (начать)

```
<<метка>>
   BEGIN
      блок
   END метка;
```

Оператором BEGIN заканчивается секция объявления данных DECLARE (если она есть) и начинается выполняемая часть блока. Каждый блок должен иметь хотя - бы один выполняемый оператор. Вместо выполняемого оператора можно использовать пустой оператор NULL; Каждый оператор BEGIN должен иметь парный ему оператор END, который заканчивает блок.

<<метка>> не обязательна. Однако, если Вы использовали ее, то к оператору END, заканчивающему этот блок, должна быть добавлена метка .

END (конец блока)

Оператор END заканчивает парный ему оператор начала блока.

Если перед блоком была поставлена метка - то обязательно необходимо указать после END имя метки

Оператор END должен быть последним в каждом блоке PL/SQL. Имя_метки позволяет ссылаться на переменные объемлющего блока, или на индекс внешнего цикла.

В блоке может использоваться пустой оператор NULL;

<<имя_метки_1>>

<<имя_метки>> необязательно. Указывется для:

- 1 - именовании блока для более наглядного восприятия
- 2 - создания внутренних блоков обработки исключительных ситуаций
- 3 - обеспечения выхода из аварийных ситуаций по оператору GOTO

FETCH (извлечь запись)

```
FETCH имя_курсора INTO имя_переменной1, имя_переменной2,...;
FETCH имя_курсора INTO имя_записи;
```

Оператор FETCH извлекает очередную строку данных из явно описанного и открытого в данный момент курсора и помещает данные в: INTO имя_переменной: список простых (скалярных) переменных.

Каждому извлеченному столбцу должна соответствовать одна переменная с аналогичным ему типом данных (или с совместимым, автоматически преобразуемым типом).

Данный оператор есть единственно возможный метод выборки данных из курсора. При этом следует учитывать ,что после извлечения данных производится сдвижка курсора на следующую запись в таблице.

INTO имя_записи: переменная для хранения записи, описанная при помощи %ROWTYPE.

Для извлечения множества строк данных можно использовать цикл FOR с курсором. В таком случае не будет необходимости открывать (OPEN), извлекать данные (FETCH) и закрывать (CLOSE) курсор для каждой строки.

FOR UPDATE OF (чтобы обновить)

```
SELECT list  FROM...
FOR UPDATE OF столбец, столбец,...—обязателен хотя-бы один
NOWAIT;                                  -- столбец
```

Блокировка FOR UPDATE OF “запирает” извлеченные строки таблицы перед выполнением одного или нескольких операторов UPDATE...WHERE.

Остальные пользователи не смогут заблокировать или изменить эти строки до тех пор, пока они не будут освобождены операторами COMMIT или ROLLBACK.

Предложение FOR UPDATE OF можно использовать также и с операторами INSERT и DELETE.

Оператор FOR UPDATE OF нельзя использовать вместе с операциями над множествами, с функциями от множеств и с фразами DISTINCT, GROUP BY, UNION, INTERSECT или MINUS.

Слово NOWAIT вызовет пропуск оператора, если хотя бы одна из выбранных строк заблокирована другим пользователем (вместо ожидания снятия блокировки).

CLOSE (закрыть)

CLOSE имя_курсора;

Оператор CLOSE выполняет отключение триггера, оставляя активный набор неопределенным. Оператором CLOSE можно закрыть только явно описанный и открытый в данный момент курсор.

Чтобы произвести выборку (FETCH) из закрытого курсора, его необходимо открыть заново. После этого система сделает следующее:

- Выполнит оператор OPEN
- Вычислит заново все параметры
- Инициализирует активный набор (т.е. строки, извлеченные запросом)

DECLARE CURSOR (объявить курсор)

При объявлении курсора не следует второй раз писать оператор DECLARE . Курсор объявляется так же как любая переменная PL/SQL.

```
DECLARE
CURSOR имя_курсора
(имя_параметра тип_данных),...—необязательно
IS SELECT_оператор
FOR UPDATE OF имя_столбца;
```

FOR UPDATE необходимо, если в данном блоке есть операторы UPDATE или DELETE с предложением WHERE CURRENT OF.

Оператор DECLARE CURSOR присваивает курсору имя и запрос.

Если Вы используете параметры ‘имя_параметра’, то необходимо использовать их также и в операторе SELECT_оператор, описывающем запрос для данного курсора.

Во избежание путаницы рекомендуется давать параметрам имена, отличающиеся от имен столбцов.

Курсоры

Курсор -- это рабочая область PL/SQL, хранящая обрабатываемую в данный момент строку данных . Курсор используется в процедурах на PL/SQL.

OPEN курсор (открыть курсор)

OPEN курсор(знач_парам1, знач_парам2...); --не обязательно

Оператор OPEN вычисляет запрос, прикрепленный к указанному курсору. Курсор должен быть объявлен до выполнения OPEN.

Если курсор был описан (в DECLARE CURSOR) с параметрами, то в операторе OPEN после имени курсора должны присутствовать значения для каждого параметра.

Если курсор был описан с параметрами по умолчанию то при открытии курсор может открываться как с параметрами так и без них.

Атрибуты курсора

cursor_name%FOUND	Есть строки в курсоре
cursor_name%NOTFOUND	Нет больше строк в курсоре
cursor_name%ROWCOUNT	Количество строк в курсоре
cursor_name%ISOPEN	Если курсор открыт

Обращение к %FOUND, %NOTFOUND, и %ROWCOUNT возбуждают исключительное состояние INVALID_CURSOR (ORA-01001), если курсор не находится в состоянии OPEN.

Цикл FOR с курсором (PL/SQL)

```
FOR имя_записи IN имя_курсора(параметр, параметр,...)
LOOP
операторы...
END LOOP;
```

При выполнении цикла FOR с курсором PL/SQL производит следующие действия:

- порождает и неявно открывает курсор, затем делает неявную выборку (FETCH) в (INTO) область имя_записи,
- извлекая строки, соответствующие прикрепленному к курсором запросу.
- Для каждой извлеченной строки выполняется последовательность операторов ‘операторы...’.
- • После извлечения всех строк PL/SQL неявно закрывает курсор (CLOSE) и выходит из цикла.

Используемый курсор должен быть предварительно объявлен и закрыт к началу выполнения цикла.

Необязательные параметры будут использованы (если они есть) до выполнения 1-го прохода по циклу при неявном открытии курсора.

Типы данных этих параметров должны быть совместимы с типами данных объявленных параметров курсора, или хотя бы преобразуемы.

Область ‘имя_записи’ неявно описывается как переменная типа имя_курсора%ROWTYPE. Для ссылки на содержащиеся в этой области значения следует использовать следующую форму записи:

имя_записи.имя_столбца

Курсор SQL%

При выполнении любого оператора SQL, не связанного с явно описанным курсором, PL/SQL автоматически открывает неявный курсор, называемый “курсор SQL%”. Атрибуты курсора SQL% позволяют получить полезную информацию о ходе и результатах выполнения операторов DELETE, UPDATE и INSERT.

АТРИБУТ	ЕСЛИ СТРОКИ БЫЛИ УДАЛЕНЫ ОБНОВЛЕНЫ ИЛИ ДОБАВЛЕНЫ	ЕСЛИ НИ ОДНА СТРОКА НЕ БЫЛА УДАЛЕНА, ОБНОВЛЕНА ИЛИ ДОБАВЛЕНА.
SQL%NOTFOUND	FALSE	TRUE
SQL%FOUND	TRUE	FALSE
SQL%ROWCOUNT	Имеет знач. = числу удаленных/обновленных/добавленных строк	= 0

Управление ходом вычислений

NULL (Пустой оператор);

Оператор NULL необязателен и никак не влияет на ход вычислений в блоке. Он может использоваться для улучшения наглядности условий. Например:

```
EXCEPTION
WHEN VALUE ERROR
THEN  GOTO calc_bonus;
INSERT INTO parts
VALUES (‘ZZ’, 9999);

WHEN OTHERS THEN
NULL:
END IF;
```

Оператор NULL не имеет никакого отношения к пустым значениям. Операторы IS NULL и IS NOT NULL являются операторами сравнения.

IF (если)

```
IF условие_1 THEN
последовательность_операторов;
ELSIF условие_2 THEN
последовательность_операторов;
ELSIF условие_3 THEN
последовательность_операторов;...
ELSE
последовательность_операторов;
ENDIF;
```

Оператор IF вычисляет условие_1 и, если оно истинно (имеет значение TRUE), то будет выполнена последовательность_операторов.

В противном случае управление передается на оператор ELSIF, который таким же образом обрабатывает условие_2, и т.д. до оператора ELSE, чья последовательность _операторов будет выполнена только если все условия в IF и ELSIF ложны.

EXCEPTION (особая ситуация)

Любая системная или программная ошибка возбуждает особую ситуацию. При этом программа может проанализировать данную ситуацию, произвести необходимые действия по её устранению.

Для выхода из режима особой ситуации есть единственный способ - переход по команде GOTO на обработку внутрь программы. Следует учитывать ,что :

- При возникновении особой ситуации выполнение программы останавливается.
- Для работы с особой ситуацией её имя должно быть объявлено так-же как тип в объемлющем блоке

```
EXCEPTION --данная секция необязательна.
WHEN имя_искл_ситуации
THEN операторы;
WHEN имя_искл_ситуации
OR имя_искл_ситуации
THEN операторы;
..WHEN OTHERS –необязательно прочие ситуации
THEN последовательность_операторы;
```

Процедуры обработки особых ситуаций описываются перед оператором END, заканчивающим блок.

В каждом предложении WHEN указывается имя особой ситуации и действия, которые необходимо предпринять, если эта ситуация возникнет.

Определяемые пользователем особые ситуации должны быть описаны в секции DECLARE текущего (или объемлющего) блока. PL/SQL также имеет набор предопределенных особых ситуаций.

Фраза OTHERS описывает порядок обработки всех ситуаций, не перечисленных во фразах WHEN. Эта фраза должна быть последней.

EXCEPTION ситуация - ситуация обработки ошибок при этом следует учитывать незавершённую транзакцию. И для её отмены следует использовать ROLLBACK

EXIT (выход)

```
EXIT имя_метки
WHEN условие_plsql;
```

Оператор EXIT без имени метки и фразы WHEN вызывает выход из текущего цикла. Чтобы выйти во внешний цикл с именем, следует использовать его имя в качестве имени_метки.

При издании директивы в режиме работы SQL*PLUS> производится выход из программы.

Если употребить фразу WHEN и условие, то выход будет происходить только если указанное условие истинно.

При выполнении в теле цикла FOR с курсором, оператор EXIT закрывает курсор.

Численный цикл FOR

```
FOR индекс_цикла IN 1-ое_целое..2-е_целое LOOP
последовательность_операторов
END LOOP;
FOR индекс_цикла IN REVERSE 1-ое_целое..2-е_целое LOOP
последовательность_операторов
END LOOP;
```

Последовательность_операторов в численном цикле FOR выполняется по одному разу для каждого значения индекса цикла.

Индекс цикла увеличивается на 1 с каждой итерацией, начиная со значения=1-ое_целое и заканчивая 2-м _целым. Когда индекс становится больше 2-го целого, выполнение цикла заканчивается.

Если использовано предложение IN REVERSE, то перед началом цикла индексу присваивается значение=2-е_целое. Индекс неявно описывается как принадлежащий к типу данных NUMBER. Он “видим” только в теле цикла и обрабатывается как константа: т.е. на его значение можно ссылаться, но нельзя изменять.

LOOP (оператор цикла)

```
<<имя_метки>> --<<имя_метки>> необязательно
LOOP
последовательность_операторов
END LOOP имя_метки;
```

Если перед оператором LOOP стоит имя_метки, то оно же должно стоять и в операторе END LOOP.

Циклы чаще всего используются вместе с оператором FETCH. Для прекращения работы в цикле можно использовать операторы GOTO, RAISE или EXIT.

```
<<first>>
LOOP
FETCH имя_курсора INTO имя_записи;
EXIT WHEN имя_курсора %NOTFOUND;
обработка_данных
END LOOP first;
```

PRAGMA EXCEPTION_INIT(установить особую ситуацию)

```
PRAGMA EXCEPTION_INIT(имя_особ_ситуации, код_ошибки);
```

Оператор PRAGMA EXCEPTION_INIT позволяет присвоить коду ошибки неописанной внутренней особой ситуации некоторое имя. ‘имя_особ_ситуации’ должно присутствовать в секции DECLARE текущего или объемлющего блока.

Одно ‘имя_особ_ситуации’ следует присваивать только одной особой ситуации. ‘код_ошибки’ может быть любым кодом, возвращаемым функцией SQLCODE.

Слово PRAGMA передает оператор PL/SQL при компиляции (т.е. он выполняется при компиляции блока, а не при его исполнении).

RAISE(возбудить особую ситуацию)

RAISE(имя_особой_ситуации);

Оператор RAISE прекращает выполнение блока блока PL/SQL и передает управление программе обработки указанной особой ситуации.

Если PL/SQL не находит обработчик данной особой ситуации в текущем блоке, то ситуация передается в объемлющий блок и т.д., пока обработчик не

будет найден. Если обработчик не обнаружится ни в одном блоке, то возникает ошибка “необработанная особая ситуация” и выполнение прекратится.

WHERE CURRENT OF (работать по курсору)

Предложение PL/SQL

Предложение WHERE CURRENT OF имя_курсора применяется в программах на языке PL/SQL для ссылки на последнюю выбранную оператором FETCH строку.

Это предложение используется вкупе с операторами UPDATE и DELETE. Указываемый курсор должен быть открытым и иметь выбранную строку. Чтобы использовать предложение WHERE CURRENT OF,

курсор должен быть описан с характеристикой FOR UPDATE OF.

Атрибуты неявного курсора SQL% и всех явных курсоров позволяют получить полезную информацию о выполнении операторов UPDATE и DELETE.

Если курсор был объявлен по одной таблице, а обновление данных проводится по другой таблице и при этом используется указанный курсор, то возникает ошибка - неправильный номер ROWID.

WHILE ("Пока" Оператор цикла)

```
WHILE условие LOOP
группа_операторов
END LOOP;
```

Цикл WHILE выполняет группу_операторов до тех пор, пока в начале каждой итерации указанное условие будет истинно. Цикл заканчивается после того, как условие станет ложным или пустым (NULL).

GOTO (перейти к)

```
<<имя_метки>>
оператор...
 ...
GOTO имя_метки;
```

Оператор GOTO передает управление на оператор, следующий после метки <<имя_метки>>.

<<имя_метки>> должно находиться в том же блоке операторов, что и GOTO.

В цикле LOOP или в ветвлении IF оператор GOTO вызовет переход на метку_3 только если <<метка_3>> находится внутри LOOP или IF. <<имя_метки>> должно быть уникальным в блоке.

Совпадающие имена могут употребляться в разных или во вложенных блоках.

Дополнительные возможности

ROLLBACK (откат)

ROLLBACK WORK;

ROLLBACK TO SAVEPOINT точка_сохранения;

Оператор ROLLBACK аннулирует все изменения, внесенные в БД после последнего оператора COMMIT и уничтожает все точки сохранения.

Оператор ROLLBACK TO точка_сохранения аннулирует все изменения, внесенные в БД после указанной точки сохранения.

ORACLE автоматически создает неявную точку сохранения перед операторами INSERT, UPDATE и DELETE, с тем, чтобы выполнить откат до этой точки, если оператор не выполнится успешно.

Слово WORK и SAVEPOINT не являются обязательными. Их можно использовать для большей наглядности.

SAVEPOINT (создать точку отката)

SAVEPOINT имя_точки;

Оператор SAVEPOINT ставит в ходе выполнения транзакции отметку, которую можно использовать в операторе ROLLBACK для указания точки, до которой нужно отменить изменения в БД.

Если в одной транзакции повторно использовать имя точки сохранения, то предыдущая точка с этим именем стирается.

Оператор ROLLBACK TO имя_точки аннулирует все изменения, внесенные в БД после установки указанной точки сохранения.

Операторы COMMIT и ROLLBACK без параметров уничтожают все точки сохранения.

SET TRANSACTION (установить режим отбора)

SET TRANSACTION READ ONLY

Оператор SET TRANSACTION READ ONLY устанавливает для текущей транзакции режим работы "только чтение". После этого оператора все запросы, выполняющиеся в текущей транзакции, будут

игнорировать изменения данных, сделанные другими пользователями. Оператор SET TRANSACTION READ ONLY особенно полезен, если нужно создать отчет на основе данных из множества таблиц, постоянно обновляемых разными пользователями. Действие оператора

SET TRANSACTION READ ONLY заканчивается при выполнении операторов COMMIT

или ROLLBACK.

При активности действия оператора SET TRANSACTION READ ONLY нельзя выполнять операторы DELETE, INSERT и UPDATE. Разрешается только выполнение запросов. Оператор SET TRANSACTION READ ONLY должен быть первым оператором в транзакции.

SET ROLE

Set role identified by <пароль роли>

Данная директива применяется для переназначения роли пользователя. Следует учитывать, что переход по ролям можно осуществлять только по тем из них, которые были установлены пользователю. Если роль не была создана с паролем, то она автоматически при ее назначении становится доступной пользователю. Роль пользователя, защищенная паролем, становится доступной только после издании дерективы Set role identified be <пароль роли>

AUDIT (ревизия)

Оператор AUDIT может использоваться для достижения двух целей: либо для установки параметров ревизии доступа к базе данных, либо для активизации отслеживания доступа к объектам базы (например, к таблицам).

**1-й случай: установка параметров ревизии **

```
SQL>AUDIT системн_параметр, системн_параметр,... |
ALL
WHENEVER SUCCESSFUL |
WHENEVER NOT SUCCESSFUL
```

Такая форма оператора AUDIT может включает регистрацию следующих событий:

- подключение к базе и отключение от нее;
- выполнение операторов, требующих права RESOURCE (системн. параметр RESOURCE);
- выполнение операторов, требующих права DBA (системн. параметр DBA);
- выполнение операторов, возвращающих ошибку ORA-942 -- “таблица или представление данных не существует” (системн. Параметр NOT EXISTS).

Ревизия включается на системном уровне при помощи параметра AUDIT_TRAIL

в файле INIT.ORA. Если ревизия не включена, то оператор AUDIT все равно выполнятся не вызвав ошибки, однако регистрационная информация не будет записываться. Все результаты ревизии (отслеживания) поступают в таблицы словаря данных. По мере заполнения ревизионных таблиц словаря, Вы можете архивировать старые записи, а затем стирать их. Но никогда не удаляйте сами ревизионные таблицы словаря данных.

*** 2-й случай: включение слежения за доступом к объектам ***

```
SQL>AUDIT табл_параметр, табл_параметр,... | ALL
ON польз.таблица | DEFAULT
BY SESSION | BY ACCESS
WHENEVER SUCCESSFUL | WHENEVER NOT
SUCCESSFUL
```

Такая форма оператора AUDIT включает регистрацию поручений доступа к таблицам и представлениям данных.

BY SESSION или BY ACCESS Определяет, как информация будет записываться в таблицу

AUDIT_TRAIL. Если употреблено предложение BY SESSION, то для каждого сеанса работы пользователя отводится одна строка (запись), изменяемая при каждой регистрации события. Предложение BY ACCESS, наоборот, вызовет создание новой записи для каждого регистрируемого события. Подразумеваемым является предложение BY SESSION.

WHENEVER SUCCESSFUL или WHENEVER NOT SUCCESSFUL

Определяет, выполнение каких операторов будет регистрироваться успешно закончившихся (WHENEVER SUCCESSFUL), или вызвавших ошибку (WHENEVER NOT SUCCESSFUL).

COMMIT [WORK] (утвердить транзакцию)

Оператор COMMIT производит регистрацию всех изменений произведенных в данной транзакции. Имя WORK не обязательно. Оно оставлено для совместимости с более ранними версиями. До проведения COMMIT изменения , проводимых пользователем , не видны другими пользователями.

```
COMMIT;
COMMIT WORK;(Утвердить работу)
```

Оператор COMMIT окончательно вносит в базу данных любые изменения, сделанные после предыдущего оператора COMMIT, и делает эту новую информацию доступной для других пользователей.

Все блокировки таблиц и строк снимаются. Все точки сохранения, объявленные после предыдущего COMMIT или ROLLBACK, стираются.

До того, как Вы подтвердите изменения оператором COMMIT, Вы можете отменить произведенную работу при помощи оператора ROLLBACK. При этом все сделанные после предыдущего COMMIT изменения отменяется

База данных остается точно в таком состоянии, в каком она была после последнего COMMIT.

Для обеспечения совместимости со стандартом ANSI рекомендуется писать оператор так: COMMIT WORK. В Personal Oracle -7 Директива аналогична директив COMMIT

COMMENT (комментарий)

```
COMMENT ON TABLE польз.таблица
IS ‘текст’
COMMENT ON COLUMN польз.таблица.столбец
IS ‘текст’
```

Оператор COMMENT позволяет внести в словарь данных любые замечания к таблице

или столбцу. Текст этих замечаний будет храниться в столбце REMARKS представлений данных словаря данных .

```
ALL_COL_COMMENTS, ALL_TAB_COMMENTS,
```

USER_COL_COMMENTS или USER_TAB_COMMENTS.

Чтобы удалить комментарий из базы данных, введите в качестве текста после IS пустую строку—два апострофа (‘’).

CONSTRAINT (Ограничения)

Под ограничениями понимаются условия, накладываемые на таблицу При этом данные, не удовлетворяющие данному условию, не попадут в таблицу

Синтаксис описания ограничений для таблицы:

```
UNIQUE | PRIMARY KEY столбец, столбец,...
CONSTRAINT имя_ограничения
FOREIGN KEY столбец, столбец,...
REFERENCES польз.таблица столбец, столбец,...
CONSTRAINT имя_ограничения
CHECK условие CONSTRAINT имя_ограничения
```

Синтаксис описания ограничений для таблицы:

```
столбец NULL | NOT NULL CONSTRAINT имя_ограничения
UNIQUE | PRIMARY KEY CONSTRAINT имя_ограничения
REFERENCES польз.таблица столбец CONSTRAINT имя_ограничения
CHECK условие CONSTRAINT имя_ограничения
```

Предложение, описывающее ограничение, позволяет определить диапазон допустимых значений одного столбца (ограничение для столбца) или группы столбцов (ограничение для таблицы).

Каждый оператор INSERT, UPDATE и DELETE будет проверяться на соответствие правилам, установленных ограничением, и будет выполнен успешно только если он удовлетворяет этим правилам.

Ограничения для таблиц являются частью глобального описания таблицы, типа

```
CREATE TABLE employee
(PROJECT NUMBER,
EMPLOYEE NUMBER,
PRIMARY KEY (PROJECT, EMPLOYEE))
```

В свою очередь ограничения для столбцов являются локальными для указанных столбцов, к примеру,

```
CREATE TABLE dept
(DEPTNO NUMBER PRIMARY KEY,...)
```

Ограничения описываются в операторах CREATE TABLE и ALTER TABLE. Оператор ALTER TABLE позволяет добавлять и уничтожать ограничения. Все определенные ограничения заносятся в словарь данных. Если Вы не дали ограничению имя, то оно будет названо SYS_Cn, где n -- целое число, используемое для обеспечения уникальности имени в базе данных.

Столбец-имя столбца, к которому относится ограничение

NULL or NOT NULL

разрешает или запрещает столбцу иметь пустое значение (NULL)

UNIQUE

указывает, что все значения в этом столбце должны быть разными.

Каждый столбец должен быть описан с атрибутом NOT NULL. Столбец не должен быть первичным ключом, поскольку первичный ключ и так уникальный

PRIMARY KEY

определяет столбец как первичный ключ

FOREIGN KEY столбец, столбец... REFERENCES польз.таблица столбец,

столбец...-определяет столбец (или столбцы) как внешний ключ к польз.таблица столбец. Подразумеваемый столбец—первичный ключ таблицы.

CHECK условие -устанавливает условие, которому должно удовлетворять значение столбца. В описании условия CHECK для столбца можно ссылаться только на тот столбец, к которому оно относится. Условие CHECK для таблицы может ссылаться на несколько столбцов.

NOAUDIT(без ревизии)

Оператор NOAUDIT может быть использован в двух целях:

- чтобы частично или полностью отменить ревизию (AUDIT) работы системы
- чтобы частично или полностью отменить действие последнего оператора AUDIT.
-

1-й случай

NOAUDIT сист_параметр, сист_параметр,... | ALL
WHENEVER SUCCESSFUL | WHENEVER NOT SUCCESSFUL

Использование оператора NOAUDIT в такой форме приведет к частичной или полной отмене ревизии работы системы. Чтобы выполнить этот оператор ,необходимо иметь права DBA.

параметр (или ALL)

параметром могут быть CONNECT, DBA, NOT EXISTS или RESOURCE.

Слово ALL означает все возможные параметры.

WHENEVER SUCCESSFUL или WHENEVER NOT SUCCESSFUL

(если успешно или если неуспешно)

отключает слежение соответственно за успешными или за неуспешными обращениями к системе. Если не указано явно, то отключается любое слежение.

2-й случай

NOAUDIT параметр, параметр,... | ALL
ON объект | DEFAULT
WHENEVER SUCCESSFUL | WHENEVER NOT SUCCESSFUL

Такой оператор NOAUDIT частично или полностью отменит действие предыдущего оператора AUDIT. Указанный ‘объект’ должен принадлежать Вам. Чтобы использовать слово DEFAULT нужно иметь права DBA. Оператор NOAUDIT вызывает подтверждение текущей транзакции.

параметр (или ALL)

Параметры ревизии

для таблиц:	ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT или UPDATE
для представлений. данных:	AUDIT, DELETE, GRANT, INSERT, LOCK, RENAME,SELECT или UPDATE
для последовательностей:	ALTER, AUDIT, GRANT или SELECT
для синонимов:	такие же, как для таблиц/представлений данных, к которым они относятся.

Слово ALL означает все возможные параметры.

объект | DEFAULT

либо принадлежащие Вам таблица или представление данных, либо (DEFAULT) подразумеваемые параметры ревизии для новых таблиц.

WHENEVER SUCCESSFUL или WHENEVER NOT SUCCESSFUL

(если успешно или если неуспешно)

отключает слежение соответственно за успешными или за неуспешными обращениями к таблице. Если не указано явно, то отключается любое слежение.

NOWAIT (без ожидания освобождения)

```
SELECT список... FROM...
FOR UPDATE OF столбец, столбец,... NOWAIT;   --требуется хотя бы один столбец
LOCK TABLE [польз.]таблица [,[польз.]таблица]...
IN режим MODE [NOWAIT]
```

Предложение NOWAIT используется в операторах SELECT и LOCK TABLE. В операторе SELECT предложение NOWAIT указывает СУБД ORACLE, что если строка таблицы заблокирована другим пользователем, то не надо ждать ее освобождения, а просто не выполнять SELECT и вернуть управление польз. В операторе LOCK TABLE предложение. NOWAIT означает что ORACLE не должен ждать освобождения таблицы, заблокированной кем-то другим, а должен в таком случае вернуть управление пользователю.

RENAME (переименовать)

RENAME текущее TO новое

Оператор RENAME заменяет текущее имя таблицы, представления данных или синонима на новое. Все дозволения и индексы старого объекта переходят к новому. Оператор RENAME не может переименовать столбец. Чтобы сделать это, используйте оператор CREATE TABLE...AS SELECT. Например, если нужно переименовать единственный столбец таблицы STATIC OLDCOL в NEWCOL, то нужно ввести:

```
CREATE TABLE temporary (newcol) AS SELECT (oldcol) FROM static
DROP TABLE static
RENAME temporary TO static
```

VALIDATE INDEX (проверить правильность индекса)

VALIDATE INDEX польз.индекс

Оператор VALIDATE INDEX проверяет, правильно ли указывает каждый индекс на блок данных. Однако, это не означает, что у каждой строки будет соотв. элемент индекса. Если ORACLE не выдаст сообщение о том, что индекс проверен, то следует уничтожить индекс и снова создать его.

LOCK TABLE (заблокировать таблицу)

```
LOCK TABLE польз.таблица, польз.таблица...
                    --польз. необязателен
IN режим MODE NOWAIT;
                     --NOWAIT необязательно
```

Оператор LOCK TABLE ограничивает доступ к одной или нескольким таблицам.

РЕЖИМ	ПОЗВОЛЯЕТ	ЗАПРЕЩАЕТ
SHARE	запросы, блокировки	блокировки, обновления,
EXCLUSIVE	только запросы	все прочие операции
ROW SHARE	конкурентное использование.	блокировку всей таблицы
ROW EXCLUSIVE	конкурентное использование.	блокировка. в режиме SHARE
SHARE ROW EXCLUSIVE	запросы	блокировка. в режиме SHARE,обновления.

Таблица остается заблокированной на протяжении всей операции DELETE, INSERT или UPDATE. Блокировка снимается при выполнении COMMIT или ROLLBACK. Если таблица уже заблокирована другим пользователем, то параметр NOWAIT вынудит вернуть управление (а не ждать освобождения таблицы).

Типы данных

BOOLEAN (тип данных)

BOOLEAN --параметров нет

Используется исключительно в PL/SQL. Переменные типа BOOLEAN (Булевы) могут принимать только следующие предопределенные константные значения:

TRUE (ИСТИНА), FALSE (ЛОЖЬ) или NULL (ПУСТОЕ).

В переменную типа BOOLEAN нельзя поместить значение, извлеченное из базы данных, так же как и нельзя записать (INSERT) такую переменную в базу.

CHAR(размер) (тип данных)

размер необязателен; если не указан, то подразумевается 1.

Тип данных CHAR служит для описания столбцов и переменных, предназначенных для хранения символьной информации в виде строк. Параметр ‘размер’ определяет максимальное кол-во символов. ‘Размер’ не может быть больше чем 255.

DATE (Формат данных типа дата)

Каждое значение типа DATE состоит из века, года, месяца, числа, часа,минут и секунд. Время хранится в 24-х часовой форме исчисления.

Подразумеваемые значения:

Если используется дата без времени, то время будет = 00:00:00 (полночь). Если используется время без даты, то дата будет сегодняшней

(SYSDATE). Если не указан формат даты, то подразумевается ДД-МЕС-ГГ (‘09-JUL-97’).

К датам можно прибавлять и отнимать от них числовые значения. Например,

```
SYSDATE + 1 = завтра;
SYSDATE - HIREDATE = всего отработанных дней
```

Представлением дат в отчетах можно управлять с помощью функций обработки дат, форматов дат и модификаторов форматов дат.

По умолчанию в операторе SELECT формат отображается в виде Даты без отображения часов, минут и секунд

Основные типы данных ORACLE

Типы данных	Описание
CHAR (size)	Строка фиксированной длинны Максимум 255 символов Умолчание - 1 байт.
VARCHAR2(size)	Переменная длинна записи Максимально - 2000 байт. Не допускается для применения в параметрах функций
FLOAT (p)	Числа с плавающей точкой (Позиция точки не зафиксирована) Максимально FLOAT (126).
NUMBER (p, s)	Фиксированные с фиксированной позицией точки. Максимально 38 значащих цифр.
DATE	Формат типа дата. Хранение в Юлианском календаре(Количество дней после 1 Января 4712 До нашей эры Максимальная дата 31 Декабря 4712 Нашей эры) Формат преобразования по умолчанию определяется переменной NLS_DATE_FORMAT или ALTER SESSION параметром.

LONG	Длинные текстовые строки до 2 гигабайт Умолчание – 1 byte. Нельзя индексировать
RAW (size)	Вид хранения не определён Умолчание нет Максимально 255 символов
LONG RAW	Длинные строки до 2 гигабайт Вид хранения не определён Умолчание – 1 byte. Нельзя индексировать
ROWID	Внутренний номер строки в базе данных. Нельзя использовать для организации ключевого поля. Используется для быстрого доступа к записи таблицы
MLSLABEL	Двоичный код операционной системы Код операционной системы от 2 до 5 байт и несёт в себе информацию для автоматического преобразования данных

Совместимые типы данных

DB2 или SQL/DS типы данных	Oracle типы данных
CHARACTER (n)	CHAR (n)
VARCHAR (n)	VARCHAR2 (n)
LONG VARCHAR	LONG
DECIMAL (p,s)	NUMBER (p,s)
INTEGER, SMALLINT	NUMBER (38)
FLOAT (p)	FLOAT (p)
DATE	DATE

ANSI SQL Типы данных	Datatype Oracle Типы данных
CHARACTER (n) CHAR (n)	CHAR (n)
NUMERIC (p,s) DECIMAL (p,s) DEC (p,s)	NUMBER (p,s)
INTEGER INT SMALLINT	NUMBER (38)
FLOAT (p)	FLOAT (p)
REAL	FLOAT (63)
DOUBLE PRECISION	FLOAT (126)
CHARACTER VARYING(n), CHAR VARYING(n)	VARCHAR2 (n)

Тип данных ‘тип_данных’ может быть CHAR, NUMBER, DATE или BOOLEAN, без каких-либо ограничений.

Вариант в SQL:

```
DELETE FROM таблица_или_предст
WHERE предложение;
```

Вариант в PL/SQL:

```
DELETE FROM таблица_или_предст
WHERE CURRENT OF имя_курсора;
```

Оператор DELETE удаляет одну или несколько строк из указанной таблицы или представления данных. Предложение WHERE определяет, какие именно строки будут удалены. Если Вы хотите удалить все строки, то опустите предложение WHERE.

Оператор PL/SQL DELETE с фразой WHERE CURRENT OF имя_курсора удаляет только текущую строку последнюю, извлеченную при помощи FETCH.

Чтобы получить информацию о ходе выполнения оператора PL/SQL DELETE, можно использовать атрибуты неявного курсора SQL% и атрибуты соответствующего.

явного курсора.

LONG (тип данных)

LONG --только в SQL. В PL/SQL нельзя использовать переменные типа LONG.

Столбцы, имеющие тип данных LONG, могут хранить строки переменной длины (вплоть до 65 535 символов). В таблице может иметься только один столбец типа LONG.

Столбцы с таким типом данных могут использоваться только в списке оператора SELECT и в предложениях SET операторов UPDATE и DELETE. Столбцы типа LONG нельзя индексировать и их нельзя использовать в качестве аргументов функций. Они не могут употребляться:

- в выражениях
- в списке SELECT вложенного запроса
- в списке SELECT распределенного запроса
- в запросах с использованием преложений UNION, INTERSECT или MINUS
- в предложениях WHERE, GROUP BY, ORDER BY, CONNECT BY, и DISTINCT

LONG RAW(тип данных)

Тип данных LONG RAW в SQL предназначен для хранения двоичных данных, типа символьных строк и цепочек графических символов. ORACLE выводит данные типа RAW в шестнадцатеричном представлении.

Тип данных LONG RAW аналогичен типу LONG, за исключением того, что при передаче данных через SQL*Net может производится перекодировка символов из системы ASCII в EBCIDIC или наоборот.

NUMBER (тип данных)

NUMBER(ширина, масштаб) --ширина и масштаб необязательны

Данные типа NUMBER предназначены для хранения числовых величин. При описании столбца типа NUMBER можно указать макс. допустимое число разрядов (precision) и кол-во разрядов после десятичной точки (scale).

Если значение, заносимое в столбец, имеет больше разрядов чем указано в описании, то выдается сообщение об ошибке “exceeds precision”. Лишние разряды после дес. запятой округляются.

Если масштаб отрицателен, то заносимое значение будет округляться до указанного числа разрядов СЛЕВА от дес. точки. Если указана только шири на, а масштаб опущен, то заносимое значение будет округлено до целого числа. Если не указана ширина, то значение будет записано как есть (ограничиваясь только макс. допустимыми шириной и масштабом).

RAW(тип данных “Сырое”)

RAW(размер)

Тип данных RAW в SQL предназначен для хранения двоичных данных, типа символьных строк и цепочек графических символов. ORACLE изображает данные типа RAW в шестнадцатеричном представлении.

Максимальная ширина столбца типа RAW = 255. Тип данных RAW аналогичен типу CHAR, за исключением того, что при передаче данных через SQL*Net может производится перекодировка символов из системы ASCII в EBCIDIC или наоборот.

Указывать размер не обязательно. Если он не указан, то подразумевается размер = 1.

Команды SQL*Plus и их использование

Подстановка (амперсенд (&))

Переменными подстановки являются имена переменных, определенных польз., перед которыми стоит один или два амперсенда (&). Когда в команде есть переменная подстановки, то SQL*Plus выполняет команду так, как будто вместо ссылки на эту переменную стоит ее значение.

Например: Предположим, что переменная SORTCOL имеет значение = “JOB”, а переменная MYTABLE = “EMP”. Тогда SQL*Plus выполнит команды

```
SQL> BREAK ON &SORTCOL SQL> SELECT &SORTCOL, SAL
1  FROM &MYTABLE
2  ORDER BY &SORTCOL;
```

так, как будто было написано
```
SQL> BREAK ON JOB
SQL> SELECT JOB, SAL
2  FROM EMP
3  ORDER BY JOB;
```

Подстановку значений переменных можно применять в любом месте команд.

SQL и SQL*Plus, кроме 1-го слова, вводимого в ответ на приглашение. Если SQL*Plus встречает подстановку переменной с неопределенным значением, то он запрашивает это значение у пользователя.

Например: Предположим, что значение для переменной GIVENNAME не определено, и Вы ввели след. команду:

```
SQL> SELECT * FROM EMP WHERE ENAME = '&GIVENNAME';
```

Тогда SQL*Plus выведет на экран приглашение:

Введите значение для givenname:

В качестве значения можно ввести любую последовательность символов, включая пробелы и знаки препинания.

Если сама переменная в команде заключена в кавычки, то их вводить не надо. SQL*Plus в любом случае будет читать вводимое Вами значение с клавиатуры терминала, даже если ввод или вывод перенаправлены в файл. Если терминала нет (в случае, когда команды выполняются из файла, в пакетном режиме), то SQL*Plus использует файл, в который перенаправлен ввод.

Чтобы добавить к подстанавливаемому значению какие-нибудь символы, надо написать их сразу после имени переменной, отделив их точкой.

Например: SQL> SELECT * FROM EMP WHERE EMPNO='&E.01';

Введите значение для E: 123

воспринимается как

SQL> SELECT * FROM EMP WHERE EMPNO='12301';

символы & и && (Подстановка)

Если перед именем переменной стоит только один амперсанд, то SQL*Plus не станет описывать ее неявной командой DEFINE.

Это значит, что встретив снова подстановку этой переменной, (в другой команде или при повторном выполнении данной) SQL*Plus заново попросит ввести ее значение.

Если перед именем переменной поставить два амперсанда, то после ввода значения она будет автоматически описана командой DEFINE и повторный ввод значения не потребуется.

Например: В командном файле STATS (выполняющем некоторые статистические вычисления) записано следующее:

```
SELECT &&GROUP_COL, MAX(&&NUMBER_COL)  MAXIMUM,
MIN(&&NUMBER_COL) MINIMUM,
SUM(&&NUMBER_COL) TOTAL,
AVG(&&NUMBER_COL) AVERAGE
FROM &TABLE
GROUP BY &&GROUP_COL;
```

При запуске этого файла SQL*Plus до начала выполнения файла запросит значения переменных:

Введите значение для group_col: PROJNO

Введите значение для number_col: SAL

Введите значение для table: EMP

Затем SQL*Plus выполнит следующий запрос:

```
SELECT PROJNO, MAX(SAL)  MAXIMUM,
MIN(SAL) MINIMUM,
SUM(*SAL) TOTAL,
AVG(SAL) AVERAGE
FROM EMP
GROUP BY PROJNO;
```

Если Вы повторно запустите выполнение этого запроса, то SQL*Plus потребует ввести только значение переменной TABLE (перед ней стоит один амперсенд). Для переменных с двумя амперсендами (GROUP_COL or NUMBER_COL) ввод значений не потребуется.

Ограничения при подстановке переменных.

Подстановку переменных нельзя использовать в командах редактирования

APPEND, CHANGE, DEL и INPUT. Также подстановку нельзя использовать в командах, в которых она бессмысленна (т.е. в командах типа HELP, REMARK и TIMING). Команды APPEND, CHANGE и INPUT воспринимают текст, начинающийся с амперсанда как простой текст.

Перечисленные ниже переменные, устанавливаемые командой SET, непосредственно влияют на процесс подстановки переменных и параметров.

SET SCAN включает и выключает подстановку.

SET DEFINE устанавливает символ подстановки. Подразумеваемый символ -- &.

SET ESCAPE устанавливает символ отмены. Этот символ нужно ставить перед символом замены, если мы хотим, чтобы он был воспринят SQL*Plus как обыкновенный символ, а не как символ замены. Подразумеваемый символ отмены -- \.

SET VERIFY ON вызывает печать на экране каждой строки командного файла до и после подстановки переменных.

SET CONCAT устанавливает символ, ставящийся между именем переменной и подсоединяемой к ее значению строкой. Подразумеваемый символ -- . (точка).

CONTINUE (Продолжение)

отменяет EXIT.

Пример: Эти команды вызывают завершение SQL*Plus и печать кода ошибки, в случае, если, команда UPDATE выполниться с ошибкой. В этом случае также будет пропущена команда COPY.

```
SQL> GET RAISE
WHENEVER SQLERROR EXIT SQL.SQLCODE
UPDATE EMP SET SAL = SAL*1.1
COPY TO SCOTT/TIGER@D:BETHESDA -
REPLACE EMP -
USING SELECT * FROM EMP
WHENEVER SQLERROR CONTINUE
```

@ (знак “at”, запуск командного файла)

@ (знак “at”—коммерческое “при”)

@имя_файла[.расширение]

Команда @ служит для запуска командных файлов.

имя_файла

командный файл, который нужно выполнить. Если вы опустите расширение, SQL*PLUS по умолчанию обычно использует расширение SQL. Подразумеваемое расширение имен файлов можно изменить при помощи команды SET.

Вы можете включать в командный файл любые команды, которые вы обычно вводите интерактивно (команды SQL или SQL*PLUS).

Данная команда действует подобно START, но не позволяет передавать параметры.

Например, чтобы выполнить командный файл PRINTRPT с расширением SQL, введите:

```
SQL> @PRINTRPT
```

Чтобы выполнить командный файл WKRPT с расширением QRY, введите:

```
SQL> @WKRPT.QRY
```

/ (наклонная черта)

Ввод наклонной черты вызывает выполнение команды SQL или блок PL/SQL, находящейся (находящегося) в данный момент в буфере команд.

Вы можете вводить наклонную черту после командного приглашения или после приглашения с номером строки, выдаваемого для ввода очередной строки.

Данная команда действует подобно команде RUN, но она не выводит на ваш экран команду из буфера.

Выполнение команды SQL или блока PL/SQL, используя “/”, не изменит номер текущей строки в буфере, если команда в буфере не содержит ошибок. В случае существования ошибки SQL*PLUS поставит указатель текущей строки на строку с ошибкой.

Например, чтобы посмотреть команду, которую Вы будете выполнять, Вы можете вывести содержимое

буфера:

```
SQL> LIST
1* SELECT ENAME, JOB FROM EMP WHERE ENAME = 'JAMES'
```

Чтобы выполнить эту команду, введите "/".

```
SQL> /
```

Для вышеприведенного запроса SQL*PLUS покажет следующее:

```
ENAME     JOB
--------- ---------
AMES        CLERK
```

ACCEPT (принять)

ACC[EPT] переменная [NUM[BER] | CHAR] [PROMPT текст | NOPR[OMT]] [HIDE]

Команда ACCEPT читает строку с устройства ввода и сохраняет ее в указанной пользователем переменной.

переменная

Имя переменной, в которой вы хотите сохранить значение. Если переменная не существует, то SQL*PLUS создаст ее.

NUM[BER]

Ограничивает тип данных переменной типом NUMBER. Если ответ не соответствует типу данных, ACCEPT выдает сообщение об ошибке и завершается.

CHAR

Ограничивает тип данных переменной типом CHAR. Если ответ не соответствует типу данных, ACCEPT выдает сообщение об ошибке и завершается.

PROMPT текст

Выводит текст на экран до приема значения переменной от пользователя.

NOPR[OMPT]

Пропускает строку и ожидает ввода без вывода подсказки.

HIDE

Подавляет отображение вводимых данных.

Например: Чтобы вывести подсказку "Оклад: " и поместить ответ в переменную SALARY (с типом данных NUMBER), введите:

```
SQL> ACCEPT salary NUMBER PROMPT 'Оклад: '
```

Чтобы вывести подсказку "Пароль: " и поместить ответ в CHAR переменную PSWD и подавить показ вводимых данных, введите:

```
SQL> ACCEPT pswd CHAR PROMPT 'Пароль: ' HIDE
```

APPEND (Добавить)

A[PPEND] текст

Команда APPEND добавляет указанный текст в конец текущей строки буфера. текст

Текст, который вы хотите добавить. Если вы хотите отделить текст от предшествующих символов пробелом, введите два пробела между APPEND и текстом.

Пример: Чтобы добавить пробел и имя столбца DEPT во второй строке буфера, сначала необходимо сделать ее текущей:

```
SQL> 2
2* FROM EMP,
Затем введите APPEND:
SQL> APPEND  DEPT
SQL> 2
2* FROM EMP, DEPT
```

Обратите внимание на двойной пробел между APPEND и DEPT. Первый разделяет APPEND от текста; второй пробел является первым добавляемым символом.

Чтобы добавить точку с запятой, введите:

```
SQL> APPEND ;;
```

SQL*PLUS добавляет первую точку с запятой к текущей строке, а вторую интерпретирует как конец

команды APPEND.

BREAK (Обьявить перерыв в отчёте)

BRE[AK] [ON элемент_отчета [действие [действие]]]

Команда BREAK определяет где и как форматирование будет изменять отчет.

BREAK используется для:

- подавления вывода дублированных значений для указанных столбцов
- пропуск строк при изменении значения столбца
- печати результата промежуточных вычислений при изменении значения столбца или в конце отчета (см. команду COMPUTE)

Команда BREAK без параметров выводит перечень текущих установок BREAK.

элемент_отчета

Необходим следующий синтаксис:

{column | expr | ROW | REPORT}

действие

Необходим следующий синтаксис:

```
SKI[P] n | [SKI[P]] PAGE]
[NODUP[LICATES] | DUP[LICATES]]
```

ON столбец [действие [действие]]

Определяются действия для SQL*PLUS, которые необходимо произвести при прерывании в указанном столбце (называется столбцом прерывания). Прерывание происходит по одной из трех причин:

- изменилось значение столбца или выражения
- окончание вывода записи
- конец отчета

Когда вы опускаете действие, “BREAK ON столбец” запрещает печать дублированных значений в столбце и помечает место в отчете, в котором SQL*PLUS произведет вычисления, определенные вами в соответствующей команде COMPUTE.

Вы можете указать “ON столбец” один или несколько раз. Если вы указали фразу ON несколько раз, как в данном случае:

```
SQL> BREAK ON DEPTNO SKIP PAGE ON JOB SKIP 1 -
ON SAL SKIP 1
```

сначала фраза ON рассматривает самое внешнее прерывание (в данном случае ON DEPTNO) и в самую последнюю очередь фраза ON рассматривает самое внутреннее прерывание (в данном случае ON SAL).

SQL*PLUS ищет прерывание в каждой выводимой записи в указанных колонках, начиная с самого внешнего прерывания до самого внутреннего прерывания в заказанном вами порядке. В данном случае, SQL*PLUS ищет измененные значения в DEPTNO, затем в JOB, затем в SAL.

Затем, SQL*PLUS выполняет действия начиная с действий указанных для самого внутреннего прерывания и так обработка идет в обратном порядке к самому внешнему прерыванию (в данном случае от SKIP 1 для ON SAL к SKIP PAGE для ON DEPTNO). SQL*PLUS выполняет все действия снизу вверх, включая действия и для столбца, в котором произошло прерывание.

Если, например, в выводимой последовательности записей изменилось значение столбца JOB, но значения DEPTNO и SAL не изменились, то SQL*PLUS пропустит две строки перед тем как напечатать запись (одна строка- как результат SKIP 1 во фразе ON SAL, и другая строка как результат SKIP 1 во фразе ON JOB).

Всегда при использовании “ON столбец”, вы должны также использовать фразу ORDER BY в команде SQL SELECT. Обычно столбцы, используемые в команде BREAK, должны появляться в той же последовательности во фразе ORDER BY (хотя все столбцы, указанные в ORDER BY необязательно должны присутствовать в команде BREAK). Это предотвращает бессмысленные прерывания в вашем отчете.

Для вышеприведенной команды BREAK, следующая команда SELECT получит следующие (осмысленные) результаты:

```
SQL> SELECT DEPTNO, JOB, SAL, ENAME
FROM EMP
ORDER BY DEPTNO, JOB, SAL, ENAME;
```

Все записи с одинаковым DEPTNO напечатаются вместе на одной странице, и на этой странице все записи с одинаковым JOB напечатаются в одной группе. Внутри каждой группы профессий (job), работники с одинаковым окладом (SAL) объединяются в группы. Прерывание в колонке ENAME не приводит ни к каким действиям, т.к. этот столбец не задан в команде BREAK.

ON выражение [действие [действие]]

Определяются действия для SQL*PLUS, которые необходимо произвести при изменении значения выражения. Если вы не задали действие, “BREAK ON expr” запретит вывод дублированных значений expr и пометит место в отчете, где SQL*PLUS произведет заданные вами вычисления в соответствующей команде COMPUTE.

Вы можете использовать в выражении один или несколько столбцов таблицы или алиас, присвоенный для столбца в команде SELECT или в команде COLUMN. Если вы используете выражение (expr) в команде BREAK, вы должны ввести данное выражение точно также, как оно задано в команде SELECT. Если выражение в SELECT равно a+b, например, вы не можете использовать b+a или (a+b) в команде BREAK для ссылки на данное выражение из команды SELECT. То, что говорилось выше о “ON столбец”, также применимо к “ON expr”.

ON ROW [действие [действие]]

Определяются действия для SQL*PLUS, которые надо произвести, когда команда SQL SELECT возвращает запись. ROW-прерывание становится в конец иерархии прерываний, независимо от того, где фраза ON ROW задана в команде BREAK. Вы должны всегда задавать действие при использовании данного прерывания.

ON REPORT

Отмечает место в отчете, где SQL*PLUS произведет вычисления определенные вами в соответствующей команде COMPUTE. Используйте BREAK

ON REPORT совместно с COMPUTE для печати общих сумм.

REPORT-прерывание становится в вершину иерархии прерываний, независимо от того, где вы его указали в команде BREAK.

Теперь рассмотрим список возможных действий:

SKI[P] n

Пропускает n строк перед выводом записи при возникновении прерывания.

[SKI[P]] PAGE

Переходит к новой странице перед выводом записи при возникновении прерывания.

NODUP[LICATES]

Печатает пробелы вместо значения столбца прерывания, когда значение столбца равно значению столбца из предыдущей записи.

DUP[LICATES]

Печатает значение столбца прерывания для каждой выбранной записи. Для печати текущих описаний прерываний необходимо просто ввести

BREAK без параметров.

Каждая новая команда BREAK, которую вы введете, заменяет предшествующую.

Примеры:

Для получения отчета, который выводит дублированные значения профессий (job), выводит средний оклад (SAL) и вставляет пустую строку, когда изменяется значение JOB, и дополнительно печатает сумму окладов и вставляет еще одну пустую строку, когда изменяется значение DEPTNO, вы можете ввести следующие команды. (В примере выбираются только отделы 10 и 30 и профессии клерк и продавец).

```
BREAK ON DEPTNO SKIP 1 ON JOB SKIP 1
DUPLICATES
SQL> COMPUTE SUM OF SAL ON DEPTNO
SQL> COMPUTE AVG OF SAL ON JOB
SQL> SELECT DEPTNO, JOB, ENAME, SAL FROM EMP
WHERE JOB IN ('CLERK','SALESMAN')
AND DEPTNO IN (10,30)
ORDER BY DEPTNO, JOB;
```

CHANGE (изменить в буфере SQL)

C[HANGE] разделитель старое [разделитель [новое [разделитель]]]

Команда CHANGE изменяет текст текущей буферной строки.

разделитель

Представляет любой не алфавитно-цифровой символ, такой как "/" или "!". Используйте символ разделитель, который не появляется ни в старом ни в новом тексте. Можно опускать пробел между CHANGE и первым sepchar.

старый

Текст, который вы хотите изменить. CHANGE не различает регистр при поиске указанного текста.

Например:

CHANGE /aq/aw

найдет первое появление "aq", "AQ", "aQ", "Aq" и заменит его на "aw". SQL*PLUS вставит текст таким, каким Вы его задали в команде.

Если старому тексту предшествует "...", то команда CHANGE заменит весь текст от начала строки до искомого текста (включая сам текст).

Если "..." стоят после старого текста, то CHANGE заменит весь текст до конца строки. Если "..." является вложенным, то CHANGE изменяет часть строки между первой частью и второй часть параметра "старый". новый

Текст заменяющий "старый" текст. Если вы опустите этот параметр и второй и третий разделители, CHANGE удалит старый текст из текущей буферной строки.

CHANGE заменяет заданный Вами текст на новый в текущей буферной строке.

При выводе команды на экран текущая строка помечается звездочкой (*).

Вы также можете использовать CHANGE для изменения строки, которая привела к ORACLE-ошибке. SQL*PLUS устанавливает указатель текущей строки на строку, содержащую ошибку, поэтому вы можете проводить изменения.

Чтобы ввести строку заново, вы можете ввести номер строки ,а за ним набрать новый текст. Если вы указали номер больше, чем количество строк в буфере, то SQL*PLUS добавит новый текст в конец буфера. Если вы указали номер строк равным нулю, то текст добавится в начало буфера (данная строка получит номер 1).

Примеры: Допустим, что текущая буферная строка содержит следующий текст:

4* WHERE JOB IN

('CLERK','SECRETARY','RECEPTIONIST')

Введите следующую команду:

SQL> C /RECEPTIONIST/GUARD/

Строка изменится следующим образом:

4* WHERE JOB IN ('CLERK','SECRETARY','GUARD')

Или введите следующую команду:

SQL> C /'CLERK',.../'CLERK')/

Строка изменится следующим образом:

4* WHERE JOB IN ('CLERK')

Или введите следующую команду:

SQL> C /(...)/('COOK','BULTER')/

Строка изменится следующим образом:

4* WHERE JOB IN ('COOK','BULTER')

Вы можете замещать содержимое любой строки, используя номер

строки. Данный ввод

SQL> 2 FROM EMP e1

означает, что вторая буферная строка замещается новым значением:

FROM EMP e1

Замечание: Ввод номера строки и последующего текста будет замещать всю строку несмотря на то, какой текст вы задали. При этом

SQL> 2 c/old/new

изменит вторую буферную строку таким образом:

SQL> c/old/new

COPY (скопировать)

```
COPY [FROM] имя_польз[/пароль][@описание_БД] |
TO имя_польз[/пароль][@описание_БД]]
{APPEND | CREATE | INSERT | REPLACE} целевая_таблица
[(столбец, столбец, столбец...)] USING запрос
```

Команда COPY копирует данные из запроса в таблицу локальной или удаленной БД.

имя_польз[/пароль]

Задает имя и пароль пользователя с которым идет обмен информацией.

Фраза FROM определяет источник информации; фраза TO определяет адресата. Если вы не укажите пароль в FROM или TO, SQL*PLUS потребует их ввести. SQL*PLUS подавляет вывод вводимого пароля.

описание_БД

Состоит из имени связи с БД или SQL*NET-строки подключения. Во фразе FROM данный параметр задает БД источник; Во фразе TO данный параметр задает БД адресат. Синтаксис параметра зависит от вашего SQL

*NET-протокола связи. Для дополнительной информации об SQL*NET “Руководство пользователя SQL*NET” или обратитесь к ДБА. SQL*PLUS не выдает подсказки для ввода спецификации БД, но он использует вашу БД по умолчанию, если вы не задали данный параметр.

целевая_таблица

Задает таблицу, которую вы хотите создать или в которую вы хотите добавить данные.

(столбец, столбец, столбец...)

Задает имена столбцов в целевой таблице. Вы должны заключать имена в двойные кавычки, если они содержат символы нижнего регистра или пробелы.

Если вы задали столбцы, то номера столбцов должны соответствовать номерам выбираемого запроса. Если вы не задали столбцы, то копируемые столбцы получат те же имена, что и в таблице-источнике, если COPY создает таблицу-адресат.

USING запрос

Задает запрос (команда SELECT), в котором определяются копируемые записи и столбцы.

FROM имя_польз[/пароль][@описание_БД]

Задает имя, пароль, БД, которые содержат копируемые данные. Если вы опустили фразу FROM, по умолчанию источником является БД, к которой подключен SQL*PLUS (т.е., БД к которой адресуются все команды). Если вы желаете копировать из другой БД, вы должны обязательно задавать фразу FROM.

TO имя_польз[/пароль][@описание_БД]

Задает БД ,содержащую таблицу-адресат. Если вы опустили фразу TO, то по умолчанию подразумевается БД, к которой подключен SQL*PLUS (т.е. БД, к которой адресуются все команды). Если вы желаете копировать в другую БД, вы должны обязательно задавать фразу TO.

APPEND

Вставляет строки из запроса в таблицу-адресат, если она существует. Если таблица-адресат не существует, COPY создает ее.

CREATE

Вставляет строки из запроса в таблицу-адресат после того, как она будет создана. Если таблица-адресат уже существует, будет получена ошибка.

INSERT

Вставляет строки из запроса в таблицу-адресат, если она существует. Если таблица-адресат не существует, будет получена ошибка.

REPLACE

Замещает данные в таблице-адресате и вставляет строки из запроса в таблицу-адресат, если она существует. Если таблица-адресат не существует, COPY создаст ее.

Переменная LONG команды SET ограничивает длину копируемых LONG-столбцов.

Если некоторые LONG-столбцы содержат данные длиннее, чем значение переменной LONG, COPY обрежет эти данные. SQL*PLUS вносит изменения в БД

после каждого успешного копирования. Если вы присвоили переменной COPYCOMMIT положительное значение n (больше чем 0), изменения вносятся каждый раз после копирования n пакетов

записей. (Размер пакета определяется системной переменной ARRAYSIZE).

Примеры: Следующая команда копирует таблицу EMP из БД HQ в таблицу с именем WESTEMP в БД WEST. Если WESTEMP уже существует, SQL*PLUS заменит ее содержимое. Колонки таблиц EMP и WESTEMP имеют одинаковые имена.

```
SQL> COPY FROM SCOTT/TIGER@HQ TO JOHN/CHROME@WEST
                -
REPLACE WESTEMP -
USING SELECT * FROM EMP
```

Следующая команда копирует выбранные записи из таблицы EMP базы данных HQ в БД, к которой подключен SQL*PLUS. SQL*PLUS создает SALESMAN при копировании. SQL*PLUS копирует только столбцы EMPNO и ENAME, и данные столбцы получат имена EMPNO и SALESMAN.

```
SQL> COPY FROM SCOTT/TIGER@HQ -
CREATE SALESMAN (EMPNO,SALESMAN) -
USING SELECT EMPNO, ENAME FROM EMP -
WHERE JOB='SALESMAN'
```

DEFINE (определить)

DEF[INE] [переменная] | [переменная = текст]

Команда DEFINE определяет переменную пользователя и присваивает ей CHAR значение. Или, показывает значение и тип одной или всех переменных.

переменная

Задает пользовательскую переменную, чье значение вы хотите посмотреть или определить.

текст

Задает символьное значение, присваиваемое переменной. Если текст содержит пробелы или знаки пунктуации, заключите его в одиночные кавычки.

переменная = текст

Определяет переменную пользователя и присваивает ей символьное значение.

Чтобы посмотреть значение и тип переменной, введите DEFINE и имя переменной. Чтобы посмотреть значение и тип всех переменных, введите DEFINE без параметров.

DEFINE-переменные сохраняют свои значения, пока не произойдет одно из следующих событий:

- вы ввели новую команду DEFINE для данной переменной
- вы ввели команду UNDEFINE для данной переменной
- вы ввели команду ACCEPT для данной переменной
- вы задали данную переменную во фразе NEW_VALUE или OLD_VALUE команды COLUMN и использовали столбец в команде SELECT.
- вы вышли из SQL*PLUS

Всякий раз, когда вы выполняете сохраненный запрос или командный файл, SQL*PLUS подставляет значение переменной для каждой подстановочной переменной, ссылающейся на данную переменную (в форме &переменная или &&переменная). Вы не получите подсказку на ввод такой переменной в данном сеансе, пока вы не уничтожите ее командой UNDEFINE.

Максимальное число переменных, определенных командами DEFINE, равно 240. Заметим, что вы можете использовать DEFINE для определения переменной _EDITOR, которая задает имя системного редактора, запускаемого по команде EDIT.

Если вы размещаете значение переменной, определяемой DEFINE, на нескольких строках (используя команду SQL*PLUS символ продолжения), SQL*PLUS заменит каждый символ продолжения и возврата каретки на пробел в итоговой переменной. Например, SQL*PLUS поймет данную команду:

```
SQL> DEFINE TEXT = 'ONE-
TWO-
THREE'
```

как:

SQL> DEFINE TEXT = 'ONE TWO THREE'

Примеры: Чтобы присвоить значение MANAGER переменной POS, введите:

SQL> DEFINE POS = MANAGER

Если вы выполните команду, которая содержит ссылку на &POS, SQL *PLUS подставит значение MANAGER вместо &POS и вы не получите подсказки на ввод значения POS.

Чтобы присвоить символьное значение 20 переменной DEPTNO, введите:

SQL> DEFINE DEPTNO = 20

Хотя вы задали 20, SQL*PLUS присвоит символьное значение DEPTNO, состоящее из двух знаков 2 и 0.

Для показа описания DEPTNO, введите:

```
SQL> DEFINE DEPTNO
DEFINE DEPTNO        = “20” (CHAR)
```

DEL (Удалить буферную строку)

Команда DEL удаляет текущую буферную строку.

Текущей становится следующая строка. Для удаления нескольких последовательных строк, введите DEL несколько раз.

Примеры: Предположим, буфер SQL содержит следующий запрос:

SELECT ENAME, DEPTNO

FROM EMP

WHERE JOB = ‘SALESMAN’

ORDER BY DEPTNO

Чтобы сделать строку с WHERE текущей, вы должны ввести:

SQL> LIST 3

3* WHERE JOB = ‘SALESMAN’

Чтобы удалить фразу WHERE, введите:

SQL> DEL

Сейчас буфер SQL содержит следующие строки:

SELECT ENAME, DEPTNO

FROM EMP

ORDER BY DEPTNO

DESCRIBE (описать)-показать описание столбцов

DESC[RIBE] [польз.]объект[@имя_связи_с_БД]

Команда DESCRIBE показывает описания столбцов для указанной таблицы, представления данных, или синонимы.

польз

Задает пользователя, который владеет объектом. Если польз опущен, SQL предполагает, что вы владелец данного объекта. объект Задает таблицу, обзор, или синоним, чье описание вы хотите вывести.

имя_связи_с_БД

Состоит из имени связи БД, где существует данный объект. За информацией о связях с БД обращайтесь к “Руководство пользователя по языку SQL” и “Справочное руководство по языку SQL”. Для каждого столбца описание содержит:

- имя столбца
- разрешено ли нулевое значение (NULL или NOT NULL)
- тип данных
- точность столбца (и количество знаков после запятой, для числовых столбцов)

Примеры: Чтобы получить описание таблицы EMP, введите:

SQL> DESCRIBE EMP

DESCRIBE покажет следующую информацию:

```
Имя                   Null?          Тип данных
 ---------------------- ------------------ ------------
  EMPNO                 NOT NULL NUMBER(4)
  ENAME                          CHAR(10)
  JOB                             CHAR(9)
  MGR                             NUMBER(4)
  HIREDATE                       DATE
  SAL                             NUMBER(7,2)
  COMM                           NUMBER(7,2)
  DEPTNO                         NUMBER(2)
```

EDIT(Редактировать)

EDIT [имя_файла[.расширение]]

Команда EDIT вызывает системный текстовый редактор для редактирования указанного файла или содержимого буфера.

имя_файла[.расширение]

Задает файл, который вы хотите редактировать (обычно командный файл). Если расширение файла опущено, берется расширение .SQL . За информацией по изменению расширения по умолчанию смотрите переменную SUFFIX команды SET в данной главе.

Чтобы откорректировать содержимое SQL буфера системным текстовым редактором, введите EDIT без имени файла.

Переменная пользователя _EDITOR содержит имя текстового редактора вызываемого с помощью команды EDIT. Значение переменной _EDITOR можно изменять. Информацию по изменению пользовательских переменных смотри в DEFINE. Если переменная _EDITOR не задана, EDIT попытается вызвать системный текстовый редактор по умолчанию.

EDIT помещает содержимое буфера в файл AFIEDT с расширением BUF (в текущий директорий) и вызывает текстовый редактор для данного файла. EDIT возвращает сообщение об ошибке, если вы не задали имя файла и буфер пуст.

Примеры: Чтобы отредактировать файл REPORT с расширением SQL, используя текстовый редактор операционной системы, введите:

SQL> EDIT REPORT

EXIT (Завершить SQL*PLUS)

{EXIT | QUIT} [SUCCESS | FAILURE | WARNING | n | переменная]

Команда EXIT вносит все отложенные изменения в БД, завершает SQL*PLUS, и передает управление операционной системе.

{EXIT | QUIT}

QUIT- это синоним EXIT.

N - Задает целый код возврата.

Переменная - Задает определенную пользователем или системную переменную, такую как SQL.SQLCODE. 'EXIT переменная' выходит с кодом возврата равным значению данной переменной.

SUCCESS

Нормальный выход.

FAILURE

Выход с кодом возврата, указывающим на сбой.

WARNING

Выход с кодом возврата указывающим на предупреждение.

EXIT без параметров выходит из SQL*PLUS со значением SUCCESS

EXIT позволяет вам указать код возврата для операционной системы. Это позволяет вам выполнять командные файлы SQL*PLUS в пакетном режиме и реагировать на непредусмотренные события. Способ обнаружения события зависит от ОС.

Ключевые слова SUCCESS,FAILURE,WARNING задают значения, зависящие от типа ОС. В некоторых система FAILURE и WARNING могут не различаться.

Замечание: SUCCESS,FAILURE,WARNING не являются зарезервированными словами.

За информацией о выходе по условию обращайся к описанию WHENEVER SQLERROR.

Примеры: Ниже приводится выход с кодом ошибки последней выполненной команды или блок PL/SQL:

SQL> EXIT SQL.SQLCODE

GET (Взять)- загрузить файл в буфер SQL

GET имя_файла[.расширение] [LIS[T] | NOL[IST]]

Команда GET загружает файл операционной системы в текущий буфер.

имя_файла[.расширение]

Задает имя загружаемого файла (обычно командный файл). Если расширение файла не указано, используется расширение по умолчанию .SQL .

Информацию об изменении расширения по умолчанию смотри описание команды SET переменная SUFFIX в данной главе.

LIS[T

Выводит содержимое файла.

NOL[IST]

Запрещает вывод.

Примеры: Чтобы загрузить файл YEARENDRPT с расширением SQL в буфер, введите:

SQL> GET YEARENDRPT

HOST (Выполнить команду ОС)

HO[ST] [команда]

Команда HOST выполняет команду операционной системы без выхода из SQL*PLUS

команда

Задает команду операционной системы. Введите HOST без команды, чтобы вывести подсказку операционной системы. Затем вы можете выполнить множество команд операционной системы. За информацией по возврату обратно в SQL*PLUS обратитесь к руководству по установке ORACLE и руководству пользователя по вашей ОС.

В некоторых операционных системах можно использовать “$” или другой символ вместо HOST. Смотри руководство по установке ORACLE и руководство пользователя для вашей операционной системы.

В некоторых операционных системах к команде HOST нет доступа.

Примеры: Чтобы выполнить команду операционной системы ls *.sql, введите:

SQL> HOST ls *.sql

INPUT(Добавить в буфер SQL)

I[NPUT] [текст]

Команда INPUT добавляет одну или более строк текста после текущей буферной строки.

текст

Задает добавляемый текст. Чтобы добавить одну строку, введите текст строки после команды INPUT, отделив текст от команды пробелом. Чтобы начать строку одним или более пробелом, введите один или более пробелов между INPUT и первым непустым символом текста.

Чтобы добавить несколько строк, введите INPUT без текста. INPUT попросит вас ввести каждую строку. Выход из INPUT- это пустая введенная строка.

Если вы ввели номер строки больший ,чем количество строк в буфере и далее текст, SQL*PLUS добавит новую строку в конец буфера. Если вы задали в качестве номера строки ноль (0) и далее указали текст, то SQL*PLUS вставит новую строку в начало буфера (данная строка получит номер 1).

Примеры: Предположим буфер SQL содержит следующую команду:

```
1 SELECT ENAME, DEPTNO, SAL, COMM
2 FROM EMP
```

Чтобы добавить фразу ORDER BY к запросу, введите:

```
SQL> LIST 2
2* FROM EMP
SQL> INPUT ORDER BY ENAME
```

LIST 2 делает вторую строку текущей. INPUT добавляет новую
строку после текущей строки. Буфер SQL сейчас содержит следующие строки:

```
1   SELECT ENAME, DEPTNO, SAL, COMM
2   FROM EMP
3* ORDER BY ENAME
```

Чтобы добавить две строки с фразой WHERE, введите:

```
SQL> LIST 2
2* FROM EMP
SQL> INPUT
3   WHERE JOB = ‘SALESMAN’
4   AND COMM=500
```

5

INPUT попросит вас ввести новые строки, пока вы не введете пустой строки. Буфер SQL сейчас содержит следующие строки:

```
SELECT ENAME, DEPTNO, SAL, COMM
FROM EMP
WHERE JOB = 'SALESMAN'
AND COMM=500
ORDER BY ENAME
```

LIST (Показать строки из буфера SQL)

L[IST] [n | n m | n * | n LAST | * | * n | * LAST | LAST]

Команда LIST выводит одну или несколько строк буфера.

n Выводит n строк.

n m Выводит строки с n по m.

n * Выводит со строки n до текущей.

n LAST Выводит со строки n до последней.

- Выводит текущую строку.
- n Выводит с текущей строки до строки с номером n.
- LAST Выводит с текущей строки до последней строки.

LAST Выводит последнюю строку.

Введите LIST без параметров, чтобы вывести все строки.

Можно опускать пробел между LIST и n или *, но но не между LIST и LAST.

Последняя выведенная строка становится новой текущей строкой (помечается звездочкой).

Примеры: Чтобы вывести содержимое буфера, введите:

SQL> L

Вы, наверное, увидите следующие результаты:

```
SELECT ENAME, DEPTNO, JOB
FROM EMP
WHERE JOB = 'CLERK'
ORDER BY DEPTNO
```

Звездочка показывает, что текущей строкой стала строка с номером 4.

Чтобы вывести только вторую строку, введите:

SQL> L 2

Выведется следующее:

2* FROM EMP

Чтобы вывести с текущей строки (сейчас это строка 2) до последней, введите:

SQL> L * LAST

Затем вы увидите следующее:

2 FROM EMP

3 WHERE JOB = 'CLERK'

4* ORDER BY DEPTNO

PAUSE - (Объявить паузу)

PAU[SE] [текст]

Команда PAUSE выводит пустую строку, затем строку содержащую "текст", и ожидает нажатие клавиши [Return]. Или, выводит две пустые строки и устанавливается в режиме ожидания ввода.

текст

Представляет текст, который вы хотите вывести. Если вы хотите вы ввести две пустых строки, введите PAUSE без параметров.

Так как PAUSE всегда ожидает отклика пользователя, лучше использовать сообщение, которое явно просит нажать [Return].

PAUSE читает ввод с терминала (если он доступен) даже, когда вы переназначили источник команд ввода на файл.

За информацией по созданию пауз между страницами отчета обратитесь к описанию переменной

PAUSE команды SET.

Примеры: Чтобы вывести сообщение “Установите бумагу, нажмите Return” с ожиданием нажатия клавиши [Return], вы можете включить следующую команду PAUSE в командный файл:

```
SQL> GET MYFILE
1   SET PAUSE OFF
2   PAUSE Установите бумагу и нажмите Return
3   SELECT ...
```

PROMPT (Вывести текст на экран)

PROMPT [текст]

Команда PROMPT выводит указанное сообщение или пустую строку на экран. текст

Текст сообщения, которое вы хотите вывести. Если “текст” опущен, PROMPT выведет пустую строку на экран.

Можно использовать данную команду в командном файле для предоставления информации пользователю.

Примеры: Следующий пример демонстрирует использование PROMPT совместно с ACCEPT в командном файле ASKFORDEPT. ASKFORDEPT содержит следующие команды SQL*PLUS и SQL:

```
PROMPT
PROMPT Please enter a valid department
PROMPT For example: 10, 20, 30, 40

ACCEPT NEWDEPT NUMBER PROMPT ‘DEPT:> ‘
SELECT DNAME FROM DEPT
WHERE DEPTNO = &NEWDEPT
```

Предположим, что вы запустили данный командный файл, используя START или @:

```
SQL> @ASKFORDEPT
```

SQL*PLUS выведет следующие подсказки:

```
Please enter a valid department
For example: 10, 20, 30, 40
DEPT:>
```

Пользователь вводит номер отдела после подсказки DEPT:>.

SQL*PLUS выведет строку с &NEWDEPT до и после подстановки, и затем покажет наименование отдела, соответствующее введенному номеру после подсказки

```
DEPT:>.
REMARK
```

REM[ARK]

Команда REMARK начинает комментарий в командном файле.

Команда REMARK пишется в начале строки и комментарий заканчивается в конце строки (в строке не может быть одновременно и комментарий и команда). SQL*PLUS не интерпретирует комментарий как команду.

Примеры: Следующий командный файл содержит несколько типичных комментариев:

```
SQL> GET EMPSUM
REM COMPUTE uses BREAK ON REPORT to break on end of table
BREAK ON REPORT
COMPUTE SUM OF “DEPARTMENT 10” DEPARTMENT 20” -
DEPARTMENT 30” “TOTAL BY JOB” ON REPORT
REM Each column displays the sum of salaries by job for
REM one of the departments 10, 20, 30.
```

```
SELECT JOB,
SUM( DECODE( DEPTNO, 10, SAL, 0)) “DEPARTMENT 10”,
SUM( DECODE( DEPTNO, 20, SAL, 0)) “DEPARTMENT 20”,
SUM( DECODE( DEPTNO, 30, SAL, 0)) “DEPARTMENT 30”,
SUM(SAL) “TOTAL BY JOB”
FROM EMP
13*  GROUP BY JOB
```

RUN (Запустить)

R[UN]

Команда RUN выводит и выполняет команду SQL или блок PL/SQL запомненный в буфере SQL.

RUN приводит к тому, что последняя строка буфера SQL становится текущей.

Команда “наклонная черта” (/) аналогична RUN, но она не выводит содержимое буфера на терминал.

SAVE (Сохранить буфер SQL в файле)

SAV[E] имя_файла[.расширение] [CRE[ATE] | REP[LACE] | APP[END]]

Команда SAVE сохраняет содержимое буфера в файле операционной системы. имя_файла[.расширение]

Задает имя файла, в котором вы хотите сохранить содержимое буфера. Если Вы не указали расширение, SQL*PLUS использует расширение по умолчанию (обычно SQL).

Если вы указываете одну из фраз CREATE, REPLACE, APPEND, то необходимо явно указать расширение файла.

CRE[ATE]

Создает файл, если он не существует.

REP[LACE]

Замещает содержимое существующего файла. Если файл не существует, REPLACE создает его.

APP[END]

Добавляет содержимое буфера в конец указанного вами файла.

Когда вы сохраняете содержимое буфера SQL, SAVE добавляет строку с наклонной чертой (/) в конец файла.

Примеры: Чтобы сохранить содержимое буфера в файл DEPTSALPRT с расширением SQL, введите:

```
SQL> SAVE DEPTSALPRT
```

Чтобы сохранить содержимое буфера в файл DEPTSALPRT с расширением OLD, введите:

```
SQL> SAVE DEPTSALPRT.OLD
```

START (Старт командного файла)

STA[RT] имя_файла[.расш] [арг1 арг2 ...]

Команда START выполняет указанный командный файл.

имя_файла[.расш]

Задает командный файл, который вы хотите выполнить. Этот файл содержит любые команды, которые вы можете выполнять интерактивно.

Если вы не указали расширение, SQL*PLUS использует расширение по умолчанию (обычно SQL). За информацией по изменению расширения по умолчанию обратитесь к описанию переменной SUFFIX команды SET в данной главе.

Когда вы ввели START имя_файла.ext, SQL*PLUS будет искать данный файл в текущем директории. Если SQL*PLUS не обнаружит данный файл, то он повторит поиск в директориях, заданных в пути для операционной системы. Некоторые операционные системы могут не поддерживать возможность пути поиска.

[арг1 арг2 ...]

Определяет аргументы, передаваемые командному файлу. Если вы ввели один или более аргументов, SQL*PLUS подставит данные значения в параметры (&1,&2, ...) командного файла. Первый аргумент замещает &1,

второй замещает &2, и т.д.

Команда START присваивает параметрам значения аргументов; если вы стартуете снова этот командный файл в данном сеансе, вы можете задать новые значения или опустить аргументы для использования старых значений.

Команда @ (знак “at”) работает аналогично START, но она не позволяет передавать параметры.

Примеры: Файл с именем PROMOTE.SQL, используемый для содействия служащим, может содержать следующую команду:

```
SELECT * FROM EMP
WHERE MGR=&1 AND JOB=&2 AND SAL>&3;
```

Чтобы выполнить данный командный файл, введите:

```
SQL> START PROMOTE 7280 CLERK 950
```

Затем SQL*PLUS выполнит следующую команду:

```
SELECT * FROM EMP
WHERE MGR=7280 AND JOB='CLERK' AND SAL>950;
```

TIMING (Записать данные хронометрирования)

TIMI[NG] [START текст | SHOW | STOP]

Команда TIMING записывает данные хронометрирования об общем времени выполнения команд, выводит заголовок текущей области хронометрирования и ее данные или выводит количество активных областей хронометрирования.

START текст

Устанавливает область хронометрирования и делает текст заголовком данной области. Вы можете иметь более одной активной области хронометрирования посредством старта дополнительной области, не остановив предыдущую область хронометрирования. Последняя запущенная область хронометрирования становится текущей.

SHOW

Выводит заголовок текущей области хронометрирования и ее данные.

STOP

Выводит заголовок текущей области хронометрирования и ее данные, затем удаляет данную область. Если существуют еще активные области хронометрирования,то одна из них, созданная самой последней, еще не уничтоженной, становится текущей. Используйте фразу TIMING в команде CLEAR для удаления всех активных областей.

Для отображения количества активных областей хронометрирования используйте TIMING без параметров.

Вы можете использовать эти данные для анализа эффективности некоторых команд или блоков PL/SQL.

За подробностями обратитесь к руководству по установке ORACLE и руководству пользователя для вашей операционной системы. Смотрите команду SET TIMING ON, для отображения текущего времени после каждой выполненной команды или блока PL/SQL.

Примеры: Для создания области хронометрирования с именем SQL_AREA, введите:

```
SQL> TIMING START SQL_AREA
```

Для вывода заголовка текущей области хронометрирования и накопленных данных, введите:

```
SQL> TIMING SHOW
```

Для вывода заголовка текущей области хронометрирования и накопленных данных, и уничтожения данной области введите:

```
SQL> TIMING STOP
```

SHOW (Показать)

SHO[W] опция

Команда SHOW выводит значения системных переменных SQL*PLUS. опция есть одна из следующих фраз:

системная_переменная

ALL

BTI[TLE]

LNO

PNO
REL[EASE]
SPOO[L]
SQLCODE
TTI[TLE]
USER
ERROR

системная_переменная
Любая системная, установленная командой SET.

ALL

Выводит установки для всех опций SHOW.

BTI[TLE]

Показывает текущее определение BTITLE.

LNO

Показывает текущий номер строки (позиция на текущей странице дисплея или буферизованного вывода).

PNO

Показывает текущий номер страницы.

REL[EASE]

Показывает номер реализации ORACLE RDBMS, с которым работает SQL*PLUS.

SPOO[L]

Показывает, является ли текущий вывод буферизованным.

SQLCODE

Показывает значение SQL.SQLCODE (код возврата последней выполненной команды)

TTI[TLE]

Показывает текущее определение TTITLE.

USER

Показывает имя пользователя, с которым вы подключились к SQL*PLUS. Чтобы посмотреть текущее значение LINESIZE, введите:

SQL> SHOW LINESIZE

Если текущее значение переменной равно 80, SQL*PLUS выдаст следующее: linesize 80

SQLPLUS (Запуск программы)

SQLPLUS [[-S[ILENT]] [вход] [запуск]] | -?

Команда SALPLUS стартует SQL*PLUS (вводится после системного приглашения).

где:

вход

Требуется следующий синтаксис:

имя_пользователя[/пароль][@спецификация_базы] | / | /NOLOG

запуск

Позволяет вводить имя командного файла и аргументы. SQL*PLUS передает аргументы командному файлу аналогично команде START. Фраза “запуск” задается в следующем формате:

@имя_файла[.расш] [арг1 арг2 ...]

Если вы не указали logon (регистрацию) и задали start, SQL*PLUS подразумевает, что первая строка командного файла содержит правильный logon. Если вы не задали ни logon, ни start, SQL*PLUS выдаст подсказку для ввода имени и пароля пользователя. имя_пользователя[/пароль]

Задает имя и пароль пользователя, с которым вы хотите подключиться к ORACLE. Если вы опустили эти параметры, SQL*PLUS попросит вас ввести их. Если вы ввели наклонную черту (/) или просто нажмете [Return] на подсказке user_name, SQL*PLUS подключит вас, используя logon по умолчанию (см “/” ниже).

Если вы опустили только пароль, SQL*PLUS попросит вас ввести его.

Вводимый вами пароль после подсказки не отображается на экране.

Представляет logon по умолчанию (ops$). Вы не можете ввести спецификацию_базы, если вы используете logon по умолчанию. По умолчанию logon SQL*PLUS попытается подключить вас, используя имя пользователя OPS$имя, где “имя” есть ваше имя в операционной системе.

/NOLOG

Заставляет стартовать SQL*PLUS, но не производить вашу регистрацию в ORACLE. Перед выполнением любой команды необходимо выполнить команду CONNECT с правильным logon, чтобы подключиться к ORACLE. Спецификация_базы.

Состоит из SQL*NET-строки подключения. Синтаксис данной строки зависит от типа протокола связи вашего ORACLE. Дополнительную информацию смотрите в руководстве по SQL*NET или спросите у администратора БД.

- S[ILENT]

Подавляет все информационные и запрашивающие сообщения SQL*PLUS, включая командную подсказку ‘SQL >’ и названия программ, которые обычно отображаются во время запуска SQL*PLUS. Это используется для того, чтобы сделать вызов SQL*PLUS другими программами невидимым для пользователя.

-?

Заставляет SQL*PLUS отобразить номер текущей версии, а затем передать управление операционной системе. Оба знака нельзя разделять пробелом.

SQL*PLUS может настраиваться с помощью файла параметров (profile), создаваемого администратором базы. SQL*PLUS выполняет этот командный файл, когда пользователь запускает SQL*PLUS и устанавливает связь с ORACLE.

Данный файл параметров позволяет ДБА настраивать среду по умолчанию для всех пользователей данной системы; пользователи не имеют доступа к данному файлу параметров. Имя данного файла зависит от типа операционной системы. Подробнее о файле параметров смотри в руководстве по установке ORACLE.

SQL*PLUS также поддерживает файл параметров пользователя (User Profile), выполняемого после главного файла параметров. SQL*PLUS ищет файл с именем LOGIN.SQL в вашей текущей директории. Если SQL*PLUS не обнаружит этого файла здесь, то SQL*PLUS будет искать данный файл в директориях заданных в пути. Некоторые операционные системы не поддерживают возможность пути поиска. Если SQL*PLUS не обнаружит LOGIN-файл, он выведет предупреждающее сообщение и продолжит процесс регистрации.

Примеры: Для запуска SQL*PLUS с именем SCOTT и паролем TIGER, введите:

SQLPLUS SCOTT/TIGER

Чтобы сделать то же самое и задать базу POLICY по умолчанию, введите:

SQLPLUS SCOTT/TIGER@POLICY

Чтобы запустить командный SQL*PLUS и выполнить командный файл STARTUP.SQL, введите: SQL> CONNECT SCOTT/TIGER @STARTUP

Команды форматирования отчета

Используются для следующих целей :

- Подготовки выходного отчёта в интерактивном режиме
- Для подготовки автоматически генерируемых командных файлов с использованием параметров внутренних метофайлов

BTITLE включает печать колонтитула в нижней части каждой страницы.

Команда BTITLE имеет более ограниченные возможности форматирования, но зато она совместима с UFI (предшественник SQL*Plus).

Команда BTITLE позволяет определить нижний колонтитул как пустую строку со следующей за ней строкой с выровненным по центру текстом.

Больше подробностей можно найти в описании старой команды TTITLE.

NEWPAGE [1|n]

Новая команда: SET команда NEWPAGE

NEWPAGE пропускает в спул-файле n строк после начала следующей страницы.

TTI[TLE] текст (верхний колонтитул)

Команда TTITLE предназначена для определения верхнего колонтитула для каждой страницы отчета.

Старая команда TTITLE имеет довольно ограниченные возможности форматирования, но зато она совместима с UFI (предшественником SQL*Plus).

Старая команда BTITLE определяет верхний колонтитул как состоящий из двух строк: 1-я состоит из даты, указанного в команде текста и номера страницы; 2-я, следующая за ней, пуста. Дата в 1-й строке выравнивается по левому краю, номер страницы—по правому, а текст—по центру.

SQL*Plus выравнивает текст по центру страницы ,исходя из заданной командой SET LINESIZE ширины строки.

Вертикальная черта (|) в тексте означает переход на новую строку.

Две вертикальные черты (||) означают пропуск одной строки. Чтобы изменить символ, которым разделяются строки при выводе, следует использовать команду SET HEADSEP.

Управлять форматированием в старых TTITLE и BTITLE можно при помощи переменной _page, содержащей шаблон формата (“page &P4”). Чтобы изменить формат, нужно переопределить эту переменную с помощью команды

DEFINE, например:

SQL> SET ESCAPE /

SQL> DEFINE _page = ‘Страница /&P2’

Теперь в колонтитулах будет печататься “Страница” и номер страницы. Число 2 в данном случае устанавливает кол-во разрядов поля для номера страницы = 2. Вместо слова ‘Страница’ можно использовать любой текст. Ширину поля номера страницы тоже можно задавать произвольно. Заметьте, что перед амперсандом (&) должен стоять ESCAPE-символ, чтобы SQL*Plus не воспринял его как символ подстановки значения переменной. ESCAPE-символ задается командой SET ESCAPE.

SQL*Plus воспринимает команду TTITLE как старую если сразу после нее не следует одно из предложений, использующихся в современной команде.

Пример: Если нужно, используя старую форму TTITLE, установить верхний колонтитул как дату и номер страницы на 1-й строке,

слово MARKETING на 2-й и PERSONNEL REPORT на 3-й, то следует ввести:

SQL> TTITLE ‘MARKETING | PERSONNEL REPORT’

Функции форматирования данных

DECODE(выражение, образец1, результат1, образец2, результат2,...не_найден)	Возвращает результат1, если ‘выражение’ = ‘образец1’, результат2, если ‘выражение’ = ‘образец2’ или ‘не_найден’, если не обнаружено ни одного совпадения. Если параметра ‘не_найден’ нет, то в случае отсутствия совпадений возвращается пустое значение.
DUMP(выражение, формат, начало, длина)	Изображает выражение в заданном формате -- 8=восьмеричный, 10= дес., 16=шестнадцатеричный, 17=символьный. ‘начало’ и ‘длина’ указывают, какую часть выражения надо обработать. Что бы изобразить все выражение опустите ‘начало’ и ‘длину’.
GREATEST(выраж1, выраж2, выраж3...)	Возвращает наибольшее из перечисленных значение выражения. Возвращаемое значение будет преобразовано в такой же тип данных, как и ‘выраж1’.

LEAST(выраж1, выраж2, выраж3...)	Возвращает наименьшее из перечисленных значение выражения. Возвращаемое значение будет преобразовано в такой же тип данных, как и 'выраж1'.
NVL(выраж1, выраж2)	Возвращает 'выраж1', если его значение не пусто и 'выраж2' в противном случае. Типы данных возвращаемого значения должен соответствовать типу данных 'выраж1'.
UID	Возвращает уникальное целое число, идентифицирующее пользователя.
USER	Возвращает имя текущего пользователя.
USERENV('ENTRYID')	Возвращает доступный идентификатор ревизионной записи.
USERENV('LANGUAGE')	Возвращает язык, установленный параметром LANGUAGE в файле INIT.ORA.
USERENV('SESSIONID')	Возвращает идентификатор ревизионного сеанса пользователя.
USERENV('TERMINAL')	Возвращает идентификатор терминала в ОС.
VSIZE(выражение)	Возвращает число байтов, использованное для хранения 'выражения' во внутреннем представлении ORACLE.

Функции DUMP, GREATEST и LEAST нельзя использовать в PL/SQL.

BTITLE (Определение заголовок отчёта в конце)

BTI[TLE] [специф_формата [текст | переменная] ...] | [OFF | ON]

Команда BTITLE размещает и форматирует заданные заголовки в конце каждой страницы отчета, или показывает текущие определения BTITLE.

SQL*PLUS определяет новый формат BTITLE, если правильная спецификация формата следует непосредственно за именем команды (LEFT,SKIP,COL, и т.д.).

За информацией о печати номеров страниц в заголовках обращайтесь к описанию команды TTITLE.

Примеры: Чтобы установить нижний заголовок "CORPORATE PLANING DEPARTMENT" выровненным влево и дату справа, введите:

```
SQL> BTITLE LEFT 'CORPORATE PLANING DEPARTMENT' -
2  RIGHT '11 MAR 1988'
```

Чтобы установить нижний заголовок "CONFIDENTIAL" в колонке 50 и дату через 6 пробелов, введите:

```
SQL> BTITLE COL 50 'CONFIDENTIAL' TAB 6 '11 MAR 97'
```

CLEAR (очистить)

CL[EAR] опция (очистить установки)

Команда CLEAR сбрасывает или очищает текущее значение или установку указанных параметров.

Опция представляет одну из следующих фраз:

BRE[AKS]

BUFF[ER]

COL[UMNS]

COMP[UTES]

SCR[EEN]

SQL

TIMI[NG]

BRE[AKS]

Удаление описания прерывания, которое установлено командой BREAK.

BUFF[ER]

Очистка буфера. CLEAR BUFFER аналогична CLEAR SQL, если вы не используете несколько буферов (см SET BUFFER в приложении F).

COL[UMNS]

Сбрасывает атрибуты вывода всех столбцов, установленные командой COLUMN, к их установкам по умолчанию. Для сброса атрибутов вывода одного столбца , используйте фразу CLEAR команды COLUMN

COMP[UTES]

Удаляет все описания COMPUTE установленные командой COMPUTE.

SCR[EEN]

Очистка экрана.

SQL

Очистка буфера SQL. CLEAR SQL аналогична CLEAR BUFFER, если вы не используете несколько буферов (см SET BUFFER в приложении F).

TIMI[NG]

Удаление всех областей хронометрирования, созданных командой TIMING.

Примеры: Чтобы очистить все прерывания, введите:

```
SQL> CLEAR BREAKS
```

Чтобы сбросить все описания столбцов, введите:

```
SQL> CLEAR COLUMNS
```

COLUMN (столбец) атрибуты колонок

COL[UMN] [{столбец|expr} [опция ...]]

Команда COLUMN определяет атрибуты вывода для всех указанных столбцов, такие как:

- текст для заголовка столбца
- выравнивание заголовка столбца
- формат для данных NUMBER
- свертка данных столбца при выводе

Также показывает текущие атрибуты вывода для одного или всех столбцов.

Опция является одной из следующих фраз:

```
ALI[AS] алиас
CLE[AR] | DEF[AULT]
COLOR {цвет | переменная_цвета}
FOLD_A[FTER] n
FOLD_B[EFORE] n
FOR[MAT] формат
HEA[DING] текст
JUS[TIFY] {L[EFT] | C[ENTER] | C[ENTRE] | R[IGHT]]
LIKE {выраж | алиас}
LINEAPP {LIKE | MARK | BOTH}
NEWL[INE]
NEW_V[ALUE] переменная
NOPRI[NT] | PRI[NT]
NUL[L] символ
OLD_V[ALUE] переменная
ON | OFF
PATTERN {число_образца | переменная_образца}
WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]
```

Для показа текущих атрибутов вывода только для одного заданного столбца или выражения введите COLUMN с именем столбца или выражения. Чтобы отобразить атрибуты вывода для всех столбцов, введите

COLUMN без параметров.

Рассмотрим перечень описаний терминов или фраз:

{столбец| выраж}

Идентифицирует элемент данных (обычно, имя столбца) из команды SQL SELECT, на который

ссылается команда COLUMN.

Если вы используете выражение (expr) в команде COLUMN, вам необходимо его задавать в таком же виде, в каком оно задается в команде SELECT.

Если выражение в команде SELECT равно a+b, например, вы не можете использовать b+a или (a+b) в команде COLUMN для ссылки на выражение из команды SELECT.

Если вы выбираете столбцы с одинаковыми именами из разных таблиц, команда COLUMN воздействует на оба этих столбца. А именно, команда COLUMN для столбца ENAME воздействует на все столбцы с именами ENAME, на которые вы ссылаетесь в течение данного сеанса. COLUMN игнорирует предшествующее имя таблицы в команде SELECT.

Чтобы отформатировать столбцы по разному, присвойте уникальный алиас каждому столбцу внутри команды SELECT (не используйте фразу ALIAS команды COLUMN) и введите команду COLUMN для каждого алиаса.

ALI[AS] алиас (псевдоним столбца)

Присваивает указанный алиас столбцу, который можно использовать для ссылки на столбец в командах BREAK,COMPUTE,COLUMN.

CLE[AR]

Сбрасывает атрибуты вывода столбцов к их значениям по умолчанию.

COLOR {цвет | переменная_цвета}

FOLD_A[FTER] n

Вставляет возврат каретки после каждого заголовка столбца и после каждой записи в столбце. Вы должны ввести n; значение n не влияет на формат.

FOLD_B[EFORE] n

Вставляет возврат каретки до каждого заголовка столбца и до каждой записи в столбце.Вы должны ввести n; значение n не влияет на формат. FOLD_BEFORE n имеет такой же эффект, как и NEWLINE.

FOR[MAT] формат

Назначает формат вывода для указанного столбца. Спецификация формата д.б. текстовой константой (такой как A10 или $9,999), но не переменной.

По умолчанию ширина столбца типа CHAR является такой, как она определена в БД или равна длине заголовка столбца (выбирается максимальное). По умолчанию ширина столбца типа LONG равна значению переменной SET LONG. Чтобы изменить ширину CHAR или LONG столбцов, используйте FORMAT An. Если вы указали ширину меньшую чем заголовок, то SQL*PLUS обрежет заголовок.

SQL*PLUS форматирует CHAR-данные выравнивая их влево. Если значение не помещается в отведенную для него ширину, то SQL*PLUS сворачивает или усекает строку символов, в зависимости от установки SET WRAP.

По умолчанию ширина неотформатированных DATE-столбцов в SQL есть A9.

Чтобы изменить формат DATE-солбца, используйте SQL-функцию TO_CHAR в вашей SQL-команде SELECT.

Когда вы используете TO_CHAR, ORACLE автоматически позволяет делать очень широкий столбец, так как SQL*PLUS автоматически ширину столбца равной 80 символов. Затем Вы должны использовать команду SQL*PLUS COLUMN для восстановления ширины столбца.

Чтобы изменить ширину DATE-столбца в n, используйте FORMAT An. Если вы указали ширину меньшую чем заголовок, то SQL*PLUS обрежет заголовок.

Замечание: Другие SQL-вычисления могут привести к подобным эффектам; в таком случае используйте SQL*PLUS команду COLUMN для сброса ширины столбца.

Ширина NUMBER-столбца по умолчанию равна значению SET NUMWIDTH.

Чтобы изменить ширину, используйте FORMAT с параметром как показано в таблице

Параметры строки FORMAT

Параметр	Пример	Описание Числовые форматы
9	9999	Задает ширину вывода посредством количества введенных цифр.

0	0999	Вывод ведущих нулей.
0	9990	Вывод нуля вместо пробела,когда значение равно нулю.
$	$9999	Перед числом выводится знак доллара.
B	B9999	Показ нулевого значения пробелом.
MI	9999MI	Вывод "-" после отрицательного числа.
PR	9999PR	Вывод отрицательного числа в угловых скобках.
Запятая	9,999	Вывод запятой в указанной позиции.
Точка	99.99	Выравнивание по указанной десятичной точке.
V	999V99	Умножает число на 10n, где n- это количество "9" после "V".
EEEE	9.999EEEE	Вывод в экспоненциальном формате (формат должен содержать четыре "E")
DATE	DATE	Выводит число как дату в формате MM/DD/YY используйте данный формат для NUMBER-столбцов, хранящих Юлианские даты.

SQL*PLUS форматирует NUMBER-данные выравнивая их вправо. Ширина поля выбирается равной или ширине заголовка или длине формата плюс один пробел для знака (выбирается большее). SQL*PLUS никогда не обрезает заголовок NUMBER-столбца. Если ширины, отведенной для столбца, не хватает для вывода числа, то SQL*PLUS выводит звездочки (*), оповещая нас о переполнении.

Для всех числовых форматов SQL*PLUS округляет числа до указанного количества значащих цифр. Когда формат не задан, по умолчанию ширина числовых данных берется из переменной NUMWIDTH (см команду SET в данной главе).

HEA[DING] текст

Определяет заголовок столбца. Если вы не используете фразу HEADING, заголовком столбца по умолчанию является имя столбца или выражения.

Если текст содержит пробелы или знаки пунктуации, вы должны заключить его в одиночные или двойные кавычки. Каждый символ HEADSEP (по умолчанию '|') начинает новую строку. Например,

COLUMN ENAME HEADING 'Employee|Name'

Задает двух строковый заголовок. Смотри переменная HEADSEP команды

JUSTIFY

JUS[TIFY] {L[EFT] | C[ENTER] | C[ENTRE] | R[IGHT]}

Выравнивает заголовок. Если вы не используете фразу JUSTIFY, NUMBER -столбца выравниваются вправо, а другие- влево.

LIKE {expr | алиас}

Копирует атрибуты вывода другого столбца или выражения (атрибуты которой уже заданы другой командой COLUMN). LIKE копирует только те атрибуты, которого не заданы в текущей команде COLUMN.

NEWL[INE]

Вывод значения столбца начинает с новой строки. NEWLINE имеет такой же эффект, что и FOLD_BEFORE n.

NEW_V[ALUE] переменная

Определяет переменную, в которую помещается значение столбца. Вы можете ссылаться на эту переменную в команде TTITLE. Используйте NEW_VALUE для вывода значения столбца или даты в верхнем заголовке.

Вы должны включить данный столбец в BREAK-команду с действием SKIP PAGE. NEW_VALUE полезна при получении основных/детальных отчетов, в которых существует новая главная запись для каждой страницы. Для основных/детальных отчетов вы должны также включить данный столбец во фразу ORDER BY. Смотрите пример в конце описания данной команды.

NOPRI[NT] | PRI[NT]

Управляет печатью столбца (заголовка и всех выбранных значений).

NOPRINT выключает печать столбца. PRINT включает печать столбца.

NUL[L] символ

Управляет выводом текста для пустых значений столбцов. Если вы не используете фразу NULL в команде COLUMN, SQL*PLUS выводит пробелы или текст, который вы задали для NULL, используя команду SET- там, где встречается пустые значения для данного столбца. (SET NULL задает текст, выводимый для всех пустых значений всех столбцов, если это не отменено для конкретного столбца фразой NULL команды COLUMN.)

OLD_V[ALUE] переменная

Определяет переменную, в которую помещается значение столбца. Вы можете ссылаться на эту переменную в команде BTITLE. Используйте OLD_VALUE для вывода значения столбца или даты в нижнем заголовке.

Вы должны включить данный столбец в BREAK-команду с действием SKIP PAGE.

OLD_VALUE полезна при получении основных/детальных отчетов, в которых существует новая главная запись для каждой страницы. Для основных/детальных отчетов вы должны также включить данный столбец во фразу ORDER BY. Подробнее о выводе значений столбца в верхний заголовок смотри COLUMN NEW_VALUE. В описании TTITLE смотри о том, как надо ссылаться на переменные в заголовках.

ON | OFF

Управляет состоянием атрибутов вывода для столбца. OFF отключает атрибуты для столбца, но не влияет на описание атрибутов. ON восстанавливает атрибуты в прежнее положение.

PATTERN {число_образца | переменная_образца}

WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]

Определяет, как SQL*PLUS будет обращаться с CHAR-строками, которые слишком широки для столбца. WRAPPED сворачивает конец строки на следующую линию. WORD_WRAPPED функция аналогична WRAPPED, но при свертке слова переносятся на другую строку полностью. TRUNCATED при выводе обрезает конец строки.

Вы можете вводить любое количество команд COLUMN для одного или нескольких столбцов. Все установленные атрибуты столбцов действуют до конца сеанса, или пока вы не отключите атрибуты фразой OFF. Таким образом, команды COLUMN, которые вы ввели, управляют атрибутами столбцов для множества команд SELECT.

Для различных столбцов эта команда может быть введена любое число раз. Также, любое число команд COLUMN может быть введено для одного и того же столбца, и их фразы будут применены совместно. Если эти несколько команд COLUMN применяют одну и ту же фразу для того же столбца, управление осуществляется с помощью последней.

Примеры: Чтобы сделать столбец ENAME шириной в 20 символов и двух строчный заголовок EMPLOEYY NAME, введите:

```
SQL> COLUMN ENAME FORMAT A20 HEADING 'EMPLOYEE |NAME'
```

Чтобы отформатировать столбец SAL таким образом, чтобы он выводил миллионы долларов, округлялся до центов, использовал запятую для разделения тысяч, и выводил $0.00, когда число равно нулю, вы должны ввести:

```
SQL> COLUMN SAL FORMAT $9,999,990.99
```

Чтобы присвоить алиас NET столбцу содержащему длинное выражение, вывести результат в формате долларов, и вывести <NULL> для пустых значений, вы можете ввести:

```
SQL> COLUMN SAL+COMM+BONUS-EXPENSES-INS-TAX ALIAS NET
SQL> COLUMN NET FORMAT $9,999,999.99 NULL '<NULL>'
```

Заметим, что в данном примере спецификация столбца разделена на две команды. Первая определяет алиас NET, а вторая использует NET для задания формата. Также заметим, что в первой команде вы должны задать выражение точно так же, как вы будете использовать его в командах SELECT. Иначе, SQL*PLUS не найдет описание данного столбца.

Чтобы 'свернуть' длинное значение в столбце REMARKS, вы можете ввести:

```
SQL> COLUMN REMARKS FORMAT A20 WRAP
```

Для того, чтобы распечатать текущую дату и наименование каждой профессии в верхнем заголовке, введите следующее. (Вопрос создания переменной с текущей датой в вашем LOGIN файле рассматривался в главе "Вывод текущей даты в заголовках" темы "Определение заголовков и размерностей страниц" в главе 4.)

```
SQL> COLUMN JOB NOPRINT NEW_VALUE JOBVAR
SQL> COLUMN TODAY NOPRINT NEW_VALUE DATEVAR
```

```
SQL> BREAK ON JOB SKIP PAGE ON TODAY
SQL> TTITLE CENTER 'Job Report' RIGHT DATEVAR SKIP
2 -LEFT 'Job:    ' JOBVAR SKIP 2
SQL> SELECT TO_CHAR(SYSDATE, 'MM/DD/YY') TODAY/
2  ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO
3  FROM EMP WHERE JOB IN ('CLERK','SALESMAN')
4  ORDER BY JOB, ENAME;
```

COMPUTE (Рассчитать)

```
COMP[UTE] [функция ...
OF {expr | столбец| алиас} ...
ON {expr | столбец| алиас | REPORT | ROW}]
```

Команда COMPUTE вычисляет и печатает итоги, используя стандартные вычисления над подмножеством выбранных записей. COMPUTE без параметров показывает все определения COMPUTE.

функция для операций COMPUTE

Функция	Операция	Тип параметра функции
AVG	Среднее непустых значений	NUMBER
COU[NT]	Счетчик ненулевых значений	все типы
MAX[IMUM]	Максимальное значение	NUMBER,CHAR
MIN[IMUM]	Минимальное значение	NUMBER,CHAR
NUM[BER]	Счетчик записей	все типы
STD	Среднеквадратичное отклонение непустых значений	NUMBER
SUM	Сумма непустых значений	NUMBER
VAR[IANCE]	Дисперсия непустых значений	NUMBER

Если вы укажите больше одной функции, то разделите их пробелами.

OF {expr | столбец| алиас} ...

Определяет столбец или выражение, которое вы хотите использовать при вычислениях. Вы должны также задать данный столбец в команде SELECT, иначе SQL*PLUS проигнорирует вашу команду COMPUTE.

Если вы не хотите, чтобы вычисляемое значение для указанного столбца появлялось на экране, подавите эту печать фразой NOPRINT команды

COLUMN. Используйте пробелы для разделения нескольких выражений, столбцов или алиасов внутри фразы OF.

Чтобы сослаться в SELECT на выражение или функцию из фразы OF, заключите ссылку на выражение или функцию в кавычки. Имя столбца или алиаса заключать в кавычки не надо.

ON {выраж | столбец| алиас | REPORT | ROW}

Определяет событие, которое SQL*PLUS будет использовать как прерывание. COMPUTE выводит вычисляемое значение и начинает повторное вычисление, когда происходит данное событие (также, когда изменяется значение выражения, после выборки новой записи, или заканчивается отчет).

Если несколько команд COMPUTE ссылаются на один и тот же столбец во фразе ON, то действует только последняя команда COMPUTE.

Чтобы сослаться в SELECT на выражение или функцию из фразы ON, заключите ссылку на выражение или функцию в кавычки. Имя столбца или алиаса заключать в кавычки не надо.

Чтобы вывести все описания COMPUTE, введите COMPUTE без параметров.

Для проведения вычислений необходимо, чтобы выполнялись следующие условия:

- Выражение, столбец, или алиас столбца, на который вы ссылаетесь во фразе ON, должны быть заданы в команде SELECT.
- Выражение, столбец, или алиас столбца, на который вы ссылаетесь во фразе ON, должны быть заданы в последней команде BREAK.
- Если вы задаете ROW или REPORT во фразе ON, также необходимо задать ROW или REPORT в последней команде BREAK.
- Одно или более выражений, столбцов, или алиасов столбцов, которые вы задаете во фразе OF, необходимо задать в команде SELECT.

Примеры: Чтобы вычислить промежуточные суммы окладов для профессий

“clerk”,”analyst”,”salesman”, введите:

```
SQL> BREAK ON JOB SKIP 1
SQL> COMPUTE SUM OF SAL ON JOB
SQL> SELECT JOB, ENAME, SAL
     2   FROM EMP
     3   WHERE JOB IN (‘CLERK’,’ANALYST’,’SALESMA’)
     4....OREDER BY JOB, SAL;
```

Чтобы вычислить средний и максимальный оклады в отделах продавцов, введите:

```
SQL> BREAK ON DNAME SKIP 1
SQL> COMPUTE AVG MAX OF SAL ON DNAME
SQL> SELECT DNAME, ENAME, SAL
2  FROM DEPT, EMP
3  WHERE DEPT.DEPTNO=EMP.DEPTNO
4  AND DNAME IN (‘ACCOUNTING’,’SALESMAN’)
5  ORDER BY DNAME;
```

SPOOL (Установить протоколирование работы)

SPO[OL] [имя_файла[.расширение] | OFF | OUT

Команда SPOOL сохраняет результаты запросов в файле операционной системы и, обычно, посылает данный файл на принтер. Также показывает текущее состояние буферизации.

имя_файла[.расширение]

Задает имя файла, в который будет производиться буферизация отображаемых выходных данных. Если вы не указали расширение, SPOOL использует расширение по умолчанию (LST или LIS в большинстве систем).

OFF

Заставляет SQL остановить буферизацию.

OUT

Останавливает буферизацию и посылает файл на принтер. Введите SPOOL без параметров, чтобы вывести текущее состояние буферизации.

Чтобы буферировать вывод генерируемый командами из командного файла без отображения его на экране, используйте SET TERMOUT OFF. SET TERMOUT OFF не влияет на вывод команд, введенных интерактивно.

Примеры: Чтобы начать запись отображаемых данных в файл с именем DIARY, введите:

SQL> SPOOL DIARY

Чтобы закрыть и распечатать файл, введите:

SQL> SPOOL OUT

TTITLE (Создать заголовок отчёта)

TTI[TLE] [формат [текст | переменная] ...] |

Команда TTITLE помещает и форматирует указанный заголовок вверху каждой страницы отчета, или выводит текущее определение TTITLE.

Замечание: Описание старого формата TTITLE, совместимого с UFI (предшественник SQL*PLUS), смотри TTITLE(старый формат) .

[OFF | ON]

где формат состоит из одной или нескольких следующих фраз, используемых для размещения и форматирования текста:

COL n

S[KIP] [n]

TAB n

LE[FT]

CE[NTER]

R[IGHT]

BOLD

FORMAT симв

Если вы не указали фразу printspec перед текстом, TTITLE выровнит текст влево. Для вывода текущего определения TTITLE, введите TTITLE без параметров.

Описание данных терминов и фраз действительно и для команды BTITLE.текст

Задает текст заголовка. Текст необходимо поместить в одиночные кавычки, если в нем больше одного слова. Переменная.

Задает пользовательскую переменную или одну из следующих системных переменных:

- SQL.LNO (текущий номер строки)
- SQL.PNO (текущий номер страницы)
- SQL.RELEASE (текущий номер реализации ORACLE)
- SQL.SQLCODE (текущий код ошибки)
- SQL.USER (текущее имя пользователя)
-

Чтобы вывести одно из этих значений, используйте соответствующее имя в заголовке. Можно форматировать переменную фразой FORMAT.

OFF

Выключает заголовок (запрещает его вывод) не влияя на его описание.

ON

Включает заголовок (разрешает его вывод). Когда вы описали верхний заголовок, SQL*PLUS автоматически устанавливает TTITLE в ON.

COL n

Перемещение в столбец n текущей строки. “Колонка” в данном контексте означает позицию печати, но не столбец таблицы.

S[KIP] [n]

Пропуск n строк; если вы опустили n, пропускается одна строка; если n=0, то возврат в начало текущей строки.

TAB n

Пропуск n столбцов (перемещение назад, если вы задали отрицательное число). “Колонка” в данном контексте означает позицию печати,но не столбец таблицы.

LE[FT]

Выравнивание влево

CE[NTER]

Выравнивание по центру

R[IGHT]

Выравнивание вправо

LEFT, CENTER и RIGHT используют значение SET LINESIZE для вычисления позиции, в которую необходимо поместить данные.

BOLD

Печатает данные полужирным шрифтом. SQL*PLUS показывает полужирную печать на терминале повторением данных на трех последовательных строках.

FORMAT симв

Задает модель формата, которая определяет формат для последующих элементов данных. Модель формата должна быть символьной константой, такой как A10 или $999- но не переменной.

Если тип данных модели формата не соответствует типу элемента данных, фраза FORMAT не оказывает влияния на данный элемент.

Если данное несоответствие обнаружено, то SQL*PLUS печатает числовые значения, используя формат по умолчанию для числовых значений или формат заданный фразой SET NUMFORMAT. Данные типа DATE выводятся в формате по умолчанию.

SQL*PLUS интерпретирует TTITLE в новом формате, если допустимая фраза printspec (LEFT, SKIP, COL, и т.д.) следует непосредственно за именем команды. Смотри описание COLUMN NEW_VALUE для печати столбцов и дат в верхнем заголовке.

Вы можете использовать любое количество констант и переменных в printspec. SQL*PLUS будет выводить константы и переменные в указанном вами порядке, позиционируя и форматируя каждую

константу и переменную в соответствии с предшествующей фразой printspec.

Примеры: Чтобы определить верхний заголовок: “Данные месяца” выровненным влево, дату по центру и справа номер страницы, и вывести “В тысячах” полужирным шрифтом в центре следующей строки, введите:

```
SQL> TTITLE LEFT ‘Данные месяца’ CENTER ‘11 Mar  88’ -
  RIGHT ‘Стр:’ FORMAT 999 SQL.PNO SKIP CENTER
  BOLD -‘В тысячах’
```

В результате будет следующий заголовок:

Данные месяца 11 Mar 88

Стр: 1

В тысячах

Чтобы подавить вывод заголовка без изменения его описания, введите: SQL> TTITLE OFF

UNDEF[INE] переменная

Команда UNDEFINE уничтожает переменную пользователя, которую вы создали явно (командой DEFINE) или неявно (с помощью аргументов команды START).Переменная имя переменной пользователя, которую вы хотите удалить.Примеры: Для удаление переменной с именем POS, введите: SQL> UNDEFINE POS

Функции преобразования

TO_CHAR(выражение, формат)	Превращает численное выражение или выражение с использованием дат в символьное представление по образцу, заданному форматом.
TO_DATE(char, формат)	Преобразует дату из char в значение типа DATE, используя указанный формат.
TO_NUMBER(char)	Преобразует число из CHAR в значение типа NUMBER.
CHARTOROWID(char)	Преобразует символьные значения в значения для ROWID.
ROWIDTOCHAR(ROWID)	Преобразует значение ROWID в символьное.
CONVERT(char, new, source)	Переводит char из исходной системы кодировки символов в другую систему кодировки.
HEXTORAW(char)	Преобразует шестнадцатеричные числа в двоичные бесформатные значения (тип данных RAW).
RAWTOHEX(raw)	Преобразует бесформатные (двоичные) данные в символьное (шестнадцатеричное) представление.

Построчные символьные функции.

ASCII(строка)	Вычисляет код в 1-ом символе строки по таблице сортирующей последовательности.
CHR(n)	Возвращает символ, имеющий код (ASCII или EBCDIC) равный n.
INITCAP(симв)	Делает 1-ую букву каждого слова в строке прописной, а остальные буквы—строчными.
INSTR (симв1,симв2,m,n)	Возвращает позицию n-ного вхождения подстроки симв2 в строку симв1. Поиск подстроки симв2 начинается с поз. m. Пустые n и m задаются как 0.
LENGTH(симв)	Возвращает длину строки симв.
LOWER(симв)	Возвращает строку симв, преобразовав все буквы в строчные.
LPAD (симв1,n,симв2)	Добавляет n строк симв2 перед строкой симв1.
LTRIM(симв, s)	Удаляет начальные символы из строки симв вплоть до 1-го символа, не принадлежащего множеству s. Пустое s задается как ‘’.
NLSSORT(симв)	Выдает сортирующую последовательность для симв.
REPLACE (симв,find, new)	Заменяет каждое вхождение find на new. Чтобы удалить find, опустите new. Для одиночных символов используйте функцию TRANSLATE.
RPAD (симв1,n,симв2)	Добавляет n раз симв2 после симв1.
RTRIM(симв,set)	Удаляет символы в конце симв., начиная с символа, не принадлежащего множеству set. Пустое множество set задается как ‘’.
SOUNDEX (симв)	Возвращает строку, фонетически эквивалентную симв.
SUBSTR (char , c, n)	Возвращает n символов строки char, начиная с символа c.
TRANSLATE (char, find, new)	Заменяет в char каждый find на new.
UPPER(char)	Возвращает char, преобразовав все буквы в прописные.
ABS(n)	Возвращает абсолютное значение(n).

CEIL(n)	Возвращает наименьшее целое число, большее или равное n.
FLOOR(n)	Возвращает наибольшее целое число, меньшее или равное n.
MOD(m,n)	Возвращает результат деления m на n. Если n=0, то возвращает m.
POWER(m,n)	Возвращает m в n-ной степени. n должно быть целым числом.
ROUND(m,n)	Возвращает m, округленное до n (необязательный. параметр) десятичных разрядов.
SIGN(n)	Возвращает: -1 если n отрицательно. 1, если n положительно или 0, если n = 0.
SQRT(n)	Возвращает квадратный корень n.
TRUNC(m,n)	“Сжимает” m до n (необязательный. параметр) разрядов. -n добавляет к целому числу n нулей.

SET (Установить значения системных переменных)

SET системная_переменная значения

Команда SET настраивает окружение SQL*PLUS для текущего сеанса:

- устанавливает ширину вывода данных NUMBER
- устанавливает ширину вывода данных LONG
- разрешает/запрещает печать заголовков столбцов
- устанавливает количество строк на странице

системная_переменная является одним из следующих значений:

```
ARRAY[SIZE] {20 | n}
AUTO[COMMIT] {OFF | ON | IMM[EDIATE]}
BLO[CKTERMINATOR] {. | c}
CMDS[EP] {; | c | OFF | ON}
COM[PATIBILITY] {V5 | V6}
CON[CAT] {. | c | OFF | ON}
COPYC[OMMIT] {0 | n}
CRT crt
DEF[INE] {& | c | OFF | ON}
ECHO {OFF | ON}
EMBEDDED {OFF | ON}
ESC[APE] {\ | c | OFF | ON}
FEED[BACK] {6 | n | OFF | ON}
FLU[SH] {OFF | ON}
HEA[DING] {OFF | ON}
HEADS[EP] {| | c | OFF | ON}
LIN[ESIZE] {80 | n}
LONG {80 | n}
MAXD[ATA] n
NEWP[AGE] {1 | n}
NULL текст
NUMF[ORMAT] формат
NUM[WIDTH] {10 | n}
PAGES[IZE] {14 | n}
PAU[SE] {OFF | ON | текст}
RECSEP {WR[APPED] | EA[CH] | OFF}
RECSEPCHAR {_ | c}
SCAN {OFF | ON}
SHOW[MODE] {OFF | ON}
SPA[CE] {1 | n}
SQLC[ASE] {MIX[ED] | LO[WER] | UP[PER]}
SQLCO[NTINUE] {>  | текст}
SQLN[UMBER] {OFF | ON}
SQLPRE[FIX] {# | c}
SQLP[ROMPT] {SQL> | текст}
SQLT[ERMINATOR] {; | c | OFF | ON}
SUF[FIX] {SQL | текст}
TAB {OFF | ON}
TERM[OUT] {OFF | ON}
TI[ME] {OFF | ON}
TIMI[NG] {OFF | ON}
TRIM[OUT] {OFF | ON}
UND[ERLINE] {- | c | OFF | ON}
```

VER[IFY] {OFF | ON}
WRA[P] {OFF | ON}

ARRAY[SIZE] {20 | n}

Устанавливает количество строк <называется пакетом>, которые SQL*PLUS выбирает из БД за один раз. Допустимое значение от 1 до 5000. Большее число повышает эффективность запросов и подзапросов, выбирающих много записей, но требует больше памяти в операционной системе. ARRAYSIZE не влияет на выполнение операций SQL*PLUS.

AUTO[COMMIT] {OFF | ON | IMM[EDIATE]}

Задает время проведения отложенных изменений в базе данных. ON заставляет SQL вносить произведенные изменения в БД сразу же после выполнения любой команды SQL или блока PL/SQL. OFF отменяет автоматическое сохранение, и вам придется самостоятельно сохранять изменения (например, командой SQL COMMIT). IMMEDIATE- эквивалентна опции ON.

BLO[CKTERMINATOR] {. | c}

Устанавливает не алфавитно-цифровой символ, используемый в конце блоков PL/SQL. Чтобы выполнить блок, вы должны использовать команды

RUN или /.

CMDS[EP] {; | c | OFF | ON}

Устанавливает не алфавитно-цифровой символ, используемый для разделения нескольких команд SQL*PLUS, вводимых на одной строке. ON/OFF разрешает/запрещает использование нескольких команд на одной строке; ON автоматически устанавливает точку с запятой (;) в качестве символа разделителя команд.

COM[PATIBILITY] {V5 | V6}

Задает, сохранять или нет SQL*PLUSy команды COMMIT и ROLLBACK в буфере SQL. V5 не сохраняет COMMIT и ROLLBACK в буфере SQL; V6наоборот; Установите COMPATIBILITY в V5, если вы хотите выполнить командный файл, разработанный для пятой версии ORACLE; ... V6- для шестой версии ORACLE. Смотрите описание команд COMMIT и ROLLBACK в “Справочное руководство по языку SQL”.

CON[CAT] {. | c | OFF | ON}

Устанавливает символ, который вы можете использовать для разделения имени подстановочной переменной от другого текста, если он примыкает вплотную к данной переменной. Если вы задали COMCAT ON, то SQL*PLUS восстанавливает значение символа по умолчанию, т.е. точку.

COPYC[OMMIT] {0 | n}

Задает количество пакетов, после которых команда COPY производит изменения в БД. Команда COPY производит изменения строк в целевой БД каждый раз, когда скопирует n пакетов строк. Допустимое значение от 0 до 5000. Длина пакета задается переменной ARRAYSIZE. Если вы задали COPYCOMMIT в 0, то COPY производит изменения только после окончания операции копирования.

CRT crt

Изменяет CRT-файл по умолчанию, используемый командой SQL*PLUS RUNFORM. Чтобы восстановить первоначальное имя crt, задайте SET CRT “”. Вы можете задавать имя-crt при вызове формы:

SQL> RUNFORM -c NEW имя_формы

или SQL> SET CRT NEW

SQL> RUNFORM имя_формы

При втором способе, SQL*PLUS запоминает имя-crt,. поэтому, при вызове формы не надо задавать имя-crt в течение данного сеанса.

DEF[INE] {& | c | OFF | ON}

Устанавливает символ “c”, используемый как префикс подстановочной переменной. ON/OFF включает/выключает поиск и замену подстановочной переменной ее значением. Установка DEFINE в ON

или OFF переопределяет значение установки переменной SCAN.

ECHO {OFF | ON}

Задает или запрещает вывод выполняемых команд из командного файла, запущенного командой START. ON разрешает вывод команд; OFF-запрещает.

EMBEDDED {OFF | ON}

OFF означает, что каждый отчет будет начинаться от начала новой страницы. При ON- будет начинаться с любого места страницы.

ESC[APE] {\ | c | OFF | ON}

“c” определяет ESCAPE-символ. OFF отменяет этот символ. ON разрешает ESCAPE-символ.

Если ESCAPE-символ определен, то его можно использовать, например, перед символом, объявленным в команде DEFINE, для обозначения того, что такой символ является обычным символом.

FEED[BACK] {6 | n | OFF | ON}

Выводит количество записей выбранных при запросе в момент выбора n записей. ON/OFF включает/выключает эту команду. При установке ON n устанавливается в 1. Установка n в 0 равнозначна установке OFF.

FLU[SH] {OFF | ON}

Задает, когда посылать результаты на устройство вывода пользователя. OFF разрешает операционной системе буферный вывод. ON запрещает буферизацию. OFF следует применять для командных файлов работающих не интерактивно (т.е., когда нет необходимости в выводимых сообщениях пока не закончится командный файл). Чем меньше операций ввода/ вывода, тем быстрее работает программа.

HEA[DING] {OFF | ON}

Управляет печатью заголовков столбцов. ON заставляет SQL печатать заголовки в отчетах; OFF-запрещает;

HEADS[EP] {| | c | OFF | ON}

Устанавливает символ- разделитель заголовка. Можно использовать данный символ в команде COLUMN или в старых формах BTITLE и TTITLE для разделения заголовков на несколько строк. ON/OFF включают/выключают разделитель. Когда разделитель заголовков выключен, SQL*PLUS воспринимает символ-разделитель как обычный символ.

LIN[ESIZE] {80 | n}

Устанавливает длину строки. Также контролирует позицию центрирования и выравнивания текста вправо в BTITLE и TTITLE. Допустимое значение от 1 до 500.

LONG {80|n}

Устанавливает максимальную длину LONG-значений для вывода и копирования. Допустимые значения от 1 до 32767. Значение переменной LONG должно быть меньше, чем MAXDATA.

MAXD[ATA] n

Устанавливает максимальную ширину строки, которую SQL*PLUS может обрабатывать. Максимальное и значение по умолчанию для n зависит от типа операционной системы. Смотри руководство по установке ORACLE и руководство пользователя для вашей ОС, или обратитесь к администратору БД.

NEWP[AGE] {1 | n}

Устанавливает количество пустых строк, печатаемых между началом каждой страницы и верхним заголовком. Значение равное нулю заставляет SQL*PLUS выводить между страницами символ перевода листа и очищать экран на большинстве типов терминалов.

NULL текст

Устанавливает текст, который SQL*PLUS отображает для представления пустой величины. NULL без указания “текста” побуждает SQL*PLUS выводить пробелы (по умолчанию). Используйте фразу NULL команды COLUMN для замены данного значения для некоторого столбца.

NUMF[ORMAT] формат

Устанавливает формат, используемый по умолчанию для вывода на экран числовых элементов данных. Укажите числовой формат для “формат”. Описание форматов смотри в описании фразы FORMAT команды COLUMN.

NUM[WIDTH] {10 | n}

Устанавливает ширину, используемую по умолчанию для вывода чисел.

PAGES[IZE] {14 | n}

Устанавливает количество строк между заголовком и концом страницы.

Для листа в 11 дюймов задают 54 (плюс NEWPAGE=6). Вы можете установить PAGESIZE в 0, чтобы подавить все заголовки, прерывания страниц, начальные пустые строки и другую информацию для форматирования.

PAU[SE] {OFF | ON | текст}

Позволяет управлять скролированием на терминале во время выполнения отчета. Вы должны нажать [Return] или [Clear] после каждой паузы.

ON заставляет SQL*PLUS создавать паузу перед выводом каждой страницы отчета. Текст определяет сообщение SQL*PLUSa во время паузы.

Текст с пробелами необходимо заключать в одиночные кавычки. Можно использовать терминальные escape последовательности в команде PAUSE; т.е. можно вывести сообщение в инверсном режиме и т.д.

RECSEP {WR[APPED] | EA[CH] | OFF}

и RECSEPCHAR { |c}

Выводит или печатает разделитель записей. Разделитель записей состоит из одной строки, в которой символ “c” повторяется LINESIZE раз. RECSEPCHAR описывает символ разделения записей. По умолчанию это пробел.

RECSEP сообщает SQL*PLUS, где необходимо отображать разделение записей. Например, если RECSEP WRAPPED, то SQL*PLUS выведет разделитель записей только после свернутой строки; если RECSEP EACH, то разделитель будет выводиться после каждой записи. Если вы установите RECSEP в OFF, то SQL*PLUS не будет выводить символ- разделитель записей.

SCAN {OFF | ON}

OFF запрещает обработку подстановки переменных и параметров; ON разрешает.

SHOW[MODE] {OFF | ON}

Управляет выводом старого и нового значения системных переменных SQL*PLUS, когда вы их изменяете командой SET.

SPA[CE] {1 | n}

Устанавливает количество пробелов между столбцами при выводе. Максимальное значение n равно 10.

SQLC[ASE] {MIX[ED] | LO[WER] | UP[PER]}

Переводит регистр команд SQL или блоков PL/SQL перед выполнением. SQL*PLUS переводит весь текст команды, включая литералы в кавычках, следующим образом:

- верхний регистр, если SQLCASE=UPPER
- нижний регистр, если SQLCASE=LOWER
- без изменения, если SQLCASE=MIXED

SQLCASE не изменяет содержимое буфера SQL.

SQLCO[NTINUE] {> | текст}

Устанавливает последовательность символов, выводящихся в качестве подсказки для продолжения командной строки SQL*PLUS (для продолжения используют символ переноса “-“).

SQLN[UMBER] {OFF | ON}

ON устанавливает, что для второй и последующих строк команды SQL подсказкой будет номер строки. OFF позволяет изменять подсказку через SQLPROMPT.

SQLPRE[FIX] {# | c}

Устанавливает символ префикса SQL. В то время, как вы выводите команду SQL*PLUS за подсказкой SQL>, вы можете на отдельной строке, которой предшествует символ префикса SQL, ввести команду SQL. SQL будет выполнять команду SQL, независимо от того, что введена команда SQL*PLUS.

SQLP[ROMPT] {SQL> | текст}

Устанавливает командную подсказку SQL*PLUS.

SQLT[ERMINATOR] {; | c | OFF | ON}

Устанавливает символ, использующийся для завершения команды SQL. OFF означает, что отменяются все символы окончания команды SQL; пользователь заканчивает команду SQL вводом пустой строки. ON устанавливает символ окончания в значение по умолчанию (;).

SUF[FIX] {SQL | текст}

Устанавливает использующееся по умолчанию расширение имени файла, которое SQL*PLUS использует в командах, ссылающихся на командные файлы. SUFFIX не влияет на расширение файла для вывода (при буферизации).

TAB {OFF | ON}

OFF заставляет SQL*PLUS использовать символ табуляции для форматирования абзаца в выходных данных. ON заставляет SQL*PLUS использовать для этого пробелы. Значение для TAB по умолчанию является системно-зависимым.

Введите SHOW TAB для того, чтобы посмотреть значение по умолчанию для вашей системы.

TERM[OUT] {OFF | ON}

OFF запрещает отображение выходных данных, генерируемых командами, выполняющимися из файла, в результате чего вывод может быть буферизован без вывода экран. ON разрешает отображение выходных данных.

TERMOUT OFF не влияет на вывод команд введенных интерактивно.

TI[ME] {OFF | ON}

ON заставляет SQL*PLUS отображать текущее время перед каждой командной подсказкой. OFF запрещает вывод времени.

TIMI[NG] {OFF | ON}

Управляет выводом временной статистики. ON выводит временную статистику по каждой выполненной команде SQL или блоку PL/SQL. OFF запрещает вывод. Информацию о SET TIMING ON смотри в руководстве по установке пользователя и в руководстве пользователя по вашей операционной системе.

BUF[FER] {буфер|SQL}

Команда SET BUFFER делает указанный буфер для хранения команд активным. SQL*Plus использует только один буфер—т.н. SQL-буфер.

В начале сеанса активным будет именно он. Как правило, одного буфера вполне хватает.

Если Вы введете имя несуществующего буфера, то он будет автоматически создан и ему будет

присвоено указанное имя. Дополнительные буферы исчезают при выходе из SQL*Plus.

Запуск выполнения запроса автоматически делает SQL-буфер активным. Для копирования текста из буфера в буфер можно использовать команды GET и SAVE. Команда CLEAR BUFFER уничтожает текст в активном буфере.

Команда CLEAR SQL уничтожает текст в SQL-буфере, даже если в данный момент активен другой буфер.

DOC[UMENT] {OFF|ON}

При помощи команда SET DOCUMENT можно включать (ON) и отключать (OFF) возможность ввода комментариев после команды DOCUMENT.

После выполнения SET DOCUMENT OFF (отключение) все строки, введенные после команды DOCUMENT будут восприниматься как команды, а не как комментарии. Подробности можно найти в описании команды DOCUMENT.

TRU[NCATE] {OFF|ON}

Команда SET TRUNCATE позволяет указывать, что данные, которые при выводе результатов не умещаются на одной строке, следует отбрасывать (ON), а не переносить (OFF). Следует заметить, что результаты команд

SET TRUNCATE ON и SET WRAP OFF аналогичны. Рекомендуется, однако, пользоваться командой SET WRAP, т.к. команда SHOW не “знает” параметра TRUNCATE.

TRIM[OUT] {OFF | ON}

Определяет, как выводит SQL*PLUS конечные пробелы в конце каждой строки. ON удаляет пробелы в конце каждой строки, улучшая характеристики при взаимодействии с ORACLE через медленные устройства. OFF разрешает отображение завершающих пробелов. TRIMOUT ON не привязан к буферному выводу; он игнорируется, если не установлен SET TAB ON.

UND[ERLINE] {- | c | OFF | ON}

“c” устанавливает символ, использующийся для подчеркивания заголовков столбцов в отчетах SQL*PLUS. ON/OFF включает/выключает подчеркивание, независимо от значения “c”.

VER[IFY] {OFF | ON}

ON заставляет SQL*PLUS выводить текст командной строки до и после замещения указателей подставляемых переменных действительными величинами. OFF отменяет вывод.

WRA[P] {OFF | ON}

OFF отсекает отображаемые элементы данных, если их длина больше ширины текущей строки. ON разрешает элементам данных “переноситься” на следующую строку.

Используйте фразы WRAPPED и TRUNCATED команды COLUMN, чтобы отменить установки WRAP для описанных столбцов.

SQL*PLUS поддерживает системные переменные (их также называют переменными команды SET), которые позволяют настраивать окружение SQL*PLUS в данном сеансе. Можно изменять их значения командой SET и смотреть их текущие значения командой SHOW.

Замечание: SET TRANSACTION READ ONLY является командой SQL; смотри справочное руководство по языку SQL.

Примеры: Рассмотрим несколько примеров использования переменных команды

COMPATIBILITY:

Чтобы выполнить командный файл, SALARY.SQL, созданный для версии 5 ORACLE, введите:

SQL> SET COMPATIBILITY V5

SQL> START SALARY

После выполнения файла, восстановим режим эмуляции в V6:

SQL> SET COMPATIBILITY V6

Вы можете добавить команду SET COMPATIBILITY V5 в начало командного файла и восстановить эмуляцию в конце командного файла SET COMPATIBILITY V6.

ESCAPE:

Если вы задали escape-символ равным “!”, то

SQL> ACCEPT v1 PROMPT ‘Enter !&1:’

выведет следующую подсказку:

Enter &1:

HEADING:

Для подавления вывода заголовков столбцов в отчете, введите:

SQL> SET HEADING OFF

Если вы после этого выполните SQL команду SELECT,

SQL> SELECT ENAME, SAL FROM EMP

2 WHERE JOB = ‘CLERK’;

то получите следующие результаты:

ADAMS 1100
JAMES 950
MILLER 1300

LONG:

Чтобы установить максимальную ширину выводимых и копируемых LONG-значений равной 500, введите:

SQL> SET LONG 500

LONG-данные будут свернуты при выводе на терминал; SQL*PLUS будет обрезать данные длиннее 500 символов.

SQLCONTINUE:

Чтобы установить подсказку продолжения команд SQL*PLUS равной восклицательному знаку и пробелу, введите:

SQL> SET SQLCONTINUE ‘! ‘

SQL*PLUS будет выдавать новую подсказку для строк-продолжений:

SQL> TTITLE ‘YEARLY INCOME’ -

! RIGHT SQL.PNO SKIP 2 -

! CENTER ‘PC DIVISION’

SQL>

SUFFIX:

Чтобы установить расширение для командных файлов по умолчанию равным UFI, введите:

SQL> SET SUFFIX UFI

Если вы введете

SQL> GET EXAMPLE

То SQL*PLUS будет искать файл EXAMPLE.UFI вместо EXAMPLE.SQL

Примеры

CONNECT BY

Для получения списка служащих со структурой управляющий/служащий и соответствующими такой структуре отступами в тексте используйте “древовидный” запрос типа

```
SELECT LPAD(‘ ‘, 2*LEVEL)||ename organization,
level, empno, mgr
FROM emp
CONNECT BY PRIOR empno = mgr
START WITH ename = ‘KING’;
```

Функция LPAD может использоваться для форматирования переменной при выводе данных

FOR с курсором

```
DECLARE
CURSOR emp_cursor(dnum NUMBER)
IS
SELECT sal, comm
FROM emp
WHERE deptno=dnum;
high_paid  NUMBER(4)  := 0;
BEGIN
FOR emp_record IN emp_cursor(20)
LOOP
emp_record.comm := NVL(emp_record.comm, 0);
total_wages := total_wages + emp_record.sal +
emp_record.comm;
IF emp_record.sal > 2000.00
THEN high_paid := high_paid + 1;
END IF;
END LOOP;
INSERT INTO temp VALUES (high_paid, higher_comm,
‘          Сумма выплат: ‘ || TO_CHAR(total_wages));
COMMIT;
END;
```

функции преобразования

Вычисление кол-ва оставшихся дней в году после 31 Мая 1989:

```
SELECT SYSDATE, TO_DATE(‘01-JAN-90’) - ROUND(SYSDATE) “Осталось”
FROM dummy;
SYSDATE    Осталось
31-May-89          214
```

Подсчет кол-ва служащих, нанятых в разные годы.

Нанято в COUNT(*)

```
SELECT TO_CHAR(hiredate,’YYYY’) “Нанято в”, COUNT(*)
FROM emp
GROUP BY TO_CHAR(hiredate, ‘YYYY’);
```

соотносящцйся подзапрос

Данный соотносящийся подзапрос извлекает всех служащих, чья зарплата превышает среднюю в их же отделе.

```
SELECT ename, deptno, sal
FROM emp emp_twin
WHERE sal > (SELECT AVG(sal)
FROM emp
WHERE emp.deptno = emp_twin.deptno)
ORDER BY deptno;
```

В пример ROWID с DELETE продемонстрирован другой вариант соотносящегося подзапроса.

курсор

```
DECLARE
CURSOR c1 IS
SELECT ename, sal FROM emp
```

```
ORDER BY sal DESC;
  name      emp.ename%TYPE;
  salary    emp.sal%TYPE;
BEGIN
    OPEN c1;
FOR i IN 1..5 LOOP
FETCH c1 INTO name, salary;
EXIT WHEN c1%NOTFOUND;
INSERT INTO tmp VALUES(name, salary);
COMMIT;
END LOOP;
CLOSE c1;
END;
```

двойной курсор

В процедуре может использоваться произвольное количество курсоров. При этом следует учитывать что операция SELECT курсора выполняется только в момент открытия. Поэтому в выражении WHERE может использоваться любая переменная , находящееся в области видимости

```
DECLARE
CURSOR my_cursor IS
SELECT * FROM ALL_TABLES
WHERE OWNER='&NEW_OWNER';
multiplied_sal my_cursor%ROWTYPE;
CURSOR MY_COLUMN IS
SELECT * FROM ALL_TAB_COLUMNS
WHERE OWNER=multiplied_sal.OWNER AND
TABLE_NAME=multiplied_sal.TABLE_NAME;
mul_COL MY_COLUMN%ROWTYPE;
BEGIN
OPEN my_cursor;
LOOP
FETCH my_cursor INTO multiplied_sal;
EXIT WHEN my_cursor%NOTFOUND;
DBMS_OUTPUT.ENABLE(10000);
DBMS_OUTPUT.PUT_LINE(multiplied_sal.TABLE_NAME);
DBMS_OUTPUT.PUT_LINE('----------------------');
DBMS_OUTPUT.NEW_LINE;
OPEN MY_COLUMN;
LOOP
FETCH MY_COLUMN INTO mul_COL;
EXIT WHEN MY_COLUMN%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('.....'||
mul_COL.COLUMN_NAME);
DBMS_OUTPUT.NEW_LINE;
END LOOP;
CLOSE MY_COLUMN;
END LOOP;
END;
/
```

TRIGGER (Процедура автозапуска)

Под триггером понимается процедура с автоматическим запуском по событиям над таблицей. Достаточно триггер создать, и он активизируется каждый раз при изменении в таблице.

В триггере переменная :NEW и переменная OLD есть переменные строкового типа (%ROWTYPE) и которые содержат поля как в записи. Для изменении данных в записи до изменения (BEFORE) достаточно в поле :NEW.<имя поля записи> присвоить требуемое значение.

```
CREATE OR REPLACE TRIGGER MyEdit
•  после ( до ) каких операций запускать триггер
AFTER INSERT OR UPDATE OR DELETE
--  на какие поля наложить триггер.
--  Не обязательно указывать
OF my_field
ON AutoPark – на какую таблицу наложить триггер
FOR EACH ROW – запускать для каждой строки
BEGIN
•  если были произведены вставка или изменение
IF INSERTING OR UPDATING
THEN
INSERT INTO Direction VALUES(
:NEW.CarNumber, :NEW.CarType, :NEW.DriverName,
:NEW.XCurrent, :NEW.YCurrent, :NEW.LastDate,
:NEW.LastTime, :NEW.PassQuant, :NEW.Parol);
END IF;
•  если была произведена операция удаления
IF DELETING
THEN
DBMS_OUTPUT.PUT_LINE('Car N '||:OLD.CarNumber||
' there is no in the park!!!');
END IF;
END;
```

создание пакета

Пакет - набор процедур , функций и глобальных переменных пакета, объединенных с целью их совместного использования в приложении.

При создании пакета создаётся интерфейсная часть и часть пакета называемая “Телом”

- создание интерфейсной части пакета
- Данная часть определяет какие из процедур и функций пакета будут доступны пользователям

```
CREATE OR REPLACE PACKAGE Pack1 AS
PROCEDURE Proc1(Enter_Owner IN CHAR,
Enter_Table IN CHAR);
PROCEDURE Proc2(Enter_Owner IN CHAR);
END Pack1;
/
```

- Создание тело пакета

```
CREATE OR REPLACE PACKAGE BODY Pack1 AS
PROCEDURE Proc1(Enter_Owner IN CHAR,
Enter_Table IN CHAR) IS
BEGIN
NULL; -- пустой оператор  возможен в любом месте вместо SQL директивы
END;
PROCEDURE Proc2(Enter_Owner IN CHAR) IS
BEGIN
NULL;
END;
END Pack1;
/
```

- назначение возможности доступа через синоним для

- других пользователей

```
drop public synonym pack1;
/
create public synonym packALL for pack1;
/
grant execute on packALL to public;
```

Процедуры в данном пакте - пустые. Пример ничего полезного не делает ,а служит примером построения пакета.

DELETE

В данном примере оператора DELETE внутри соотносящегося запроса используется ROWID, позволяющий избежать дублирования строк в таблице DEPT.

```
DELETE FROM dept x
WHERE ROWID > (SELECT MIN(ROWID)
FROM dept y
WHERE x.deptno = y.deptno
AND x.dname  = y.dname
AND x.loc    = y.loc);
```

Использование соотносящегося подзапроса в операторе SELECT показано в примере соотносящегося подзапроса.

типы данных

Из командного файла, создающего таблицы для примеров:

```
DROP TABLE accounts;
CREATE TABLE accounts(
account_id  NUMBER(4) NOT NULL,
```

- до 4 разрядов; не может быть пустым

```
bal             NUMBER(11, 2));
```

- до 11 разрядов, 2 из которых

```
DROP TABLE action;                          --находятся после дес. точки.
CREATE TABLE action(
   account_id    NUMBER(4) NOT NULL,
   oper_type     CHAR(1) NOT NULL,   --только один символ
   new_value      NUMBER(11, 2),
   status           CHAR(45),
   time_tag       DATE NOT NULL);     --должно быть датой;
```

- не может быть пустым

функциц обработки дат

Функция обработки дат используется для получения списка служащих, нанятых с 1997 г.

```
SELECT ename, hiredate
FROM emp
WHERE hiredate BETWEEN '1-JAN-97' AND SYSDATE
ORDER BY hiredate;

SELECT ename, hiredate,
ADD_MONTHS(hiredate, 6) "1-я Атт."
FROM emp
ORDER BY ename;
```

обработка ошибок

```
DECLARE
bad_employee_num    EXCEPTION;      --определено польз.
bad_acct_num          EXCEPTION;      --определено польз.
. . .                                          --прочие определения
BEGIN
. . .                          --Оператор PL/SQL
COMMIT;
EXCEPTION
```

```
WHEN bad_employee_num OR bad_acct_num
THEN ROLLBACK;
WHEN ZERO_DIVIDE
THEN                --особая ситуация PL/SQL
INSERT INTO inventory_table VALUES (product_name, quantity);
COMMIT;
END;
```

форматы

Посмотрите, как форматы дат и чисел влияют на получаемые результаты.

```
SELECT
TO_CHAR(sal, '$9999') monthly, ename
TO_CHAR(hiredate, 'MM-DD-YY') hired,
FROM emp
WHERE sal >= 2000
ORDER BY sal;
```

```
HIRED    MONTHLY  ENAME
06-09-96   $2450      CLARK
05-01-95   $2850      BLAKE
04-02-97   $2975      JONES
01-05-94   $5000       SCOTT
12-03-90   $7000       FORD
11-17-92   $8000        KING
```

Функция LPAD может использоваться для форматирования переменной при выводе данных

GROUP BY

Вывод самой маленькой и самой большой зарплаты, получаемой клерками в каждом отделе, записанном в таблице emp.

```
DEPTNO  MIN(SAL)  MAX(SAL)
SELECT deptno, MIN(sal), MAX(sal)
FROM emp WHERE job = 'CLERK'
GROUP BY deptno;
```

Если нужно получать информацию, только если самая низкая зарплата в отделе меньше 1000:

```
SELECT deptno, MIN(sal), MAX(sal)
FROM emp WHERE job = 'CLERK'
GROUP BY deptno
HAVING MIN(sal) < 1000;
```

групповые функции

Вот несколько примеров использования групповых функций:

```
SELECT 12 * AVG(sal)          Вычисляет сред-
  FROM emp                      нюю годовую зарпла-
  WHERE job = 'CLERK';        ту секретарей.

SELECT COUNT(comm) eligible    Подсчитывает кол-во    ELIGIBLE
  FROM emp;                    служащих, заслужива-   --------
                               ющих премии.             4

SELECT COUNT(*)                Подсчитывает кол-во    COUNT(*)
  FROM emp                     служащих в отд. #20.   --------
```

WHERE deptno = 20; Ф-ция COUNT(*) считает 5 только строки, удолвлетворяющие условие WHERE.

INSERT

Добавление информации о новом служащем в таблицу emp:

INSERT INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)

VALUES(7979, 'POWERS', 'ANALYST', 7839, '1-JUN-89', 3600, NULL, 10);

Чтобы добавить информацию из другой таблицы, следует заменить подзапросом список VALUES, например:

```
INSERT INTO bonus (ename, job, sal, comm)
SELECT ename, job, sal, comm
FROM emp
WHERE ename = 'KING';
```

LIKE

Нахождение служащих с двойной фамилией (т.е. с двумя фамилиями, разделенными дефисом):

```
SELECT ename
FROM emp
WHERE ename LIKE '%-%'
ORDER BY ename;
```

Нахождение служащих с фамилией, начинающейся с буквы 'M':

```
SELECT ename, job, deptno
FROM emp
WHERE ename LIKE 'M%';
```

LOCK TABLE

В данном примере оператор LOCK TABLE запрещает выполнять все действия ,кроме запросов, с таблицей emp. Слово NOWAIT говорит о том, что если таблица заблокирована другим пользователем, то следует продолжить выполнение программы, не дожидаясь снятия блокировки.

```
LOCK TABLE emp IN EXCLUSIVE MODE NOWAIT;
ост_часть_транзакции
COMMIT;
```

В следующем примере таблица accounts блокируется в разделенном режиме (SHARED), который позволяет выполнять запросы, но запрещает добавления, обновления и стирания.

```
LOCK TABLE accounts IN SHARE MODE;
ост_часть_транзакции
COMMIT;
```

NEXTVAL

Предположим, что существует последовательность по имени custnoseq, описанная след. образом

CREATE SEQUENCE custnoseq INCREMENT BY 10;

и таблица customer (клиент) со столбцами custno (номер), fname (имя) и lname (фамилия). Значение custnoseq.NEXTVAL используется для создания уникального номера нового клиента, добавляемого в таблицу. Этот номер будет на 10 больше, чем наибольший существующий номер клиента.

INSERT INTO cust VALUES (custnoseq.NEXTVAL, 'Stanley', 'Daleson');

Предполагая, что существует таблица custbal со столбцами custno и bal, можно выполнить нижеприведенный оператор для добавления записи о балансе неуплаты нового клиента, только что внесенного в таблицу customer.

INSERT INTO custbal VALUES (custnoseq.CURRVAL, 144.12);

пустые значения

Служащие, недостойные премирования, отмечены пустым значением в столбце comm. Чтобы посмотреть перечень достойных премии служащих, выполняем след. оператор:

```
SELECT ename, comm, sal, deptno
FROM emp
WHERE comm IS NOT NULL
ORDER BY comm DESC;
```

Заметьте, что всегда следует использовать IS или IS NOT. Операции сравнения <> или = со значением NULL всегда вырабатывают FALSE (ложь).

Чтобы записать в столбец пустое значение используйте два следующих друг за другом апострофа, вот так:

INSERT INTO emp VALUES (8905, 'POST', '', 7698,...);

численный цикл FOR

```
DECLARE
   x  NUMBER := 100;
 BEGIN
 FOR i in 1..4 LOOP
      IF MOD(i,2) = 0  THEN
INSERT INTO temp VALUES (i, x, 'четный индекс');
ELSE
INSERT INTO temp VALUES (i, x, 'нечетный индекс');
END IF;
x := x + 25;
END LOOP;
COMMIT;
END;
SQL>  SELECT * FROM temp
ORDER BY col1;
```

избыточное соединение

Соединить таблицы EMP и DEPT (избыточным соединением) и извлечь перечень отделов 30 и 40, со служащими или без них.

```
SELECT ename, deptno, dname
FROM emp, dept
WHERE dept.deptno = emp.deptno
AND dept.deptno IN (30, 40)
ORDER BY ename;
```

запросы

Найти управляющих и служащих с окладом $5000 или выше и вывести их список в порядке уменьшения оклада.

```
SELECT ename, sal, job          ENAME      SAL    JOB
FROM emp                        ----------------------------------------
WHERE job = 'MANAGER'       KING     5000  PRESIDENT
OR sal >= 5000                JONES    2975  MANAGER
ORDER BY sal DESC;            BLAKE    2850  MANAGER
CLARK    2450  MANAGER
```

оператор ROLLBACK

При обновлении информации о заработной плате Вы можете установить несколько точек сохранения, присваивая каждой уникальное имя. Чтобы отменить действия, произведенные в базе данных после некоторой точки сохранения, надо выполнить оператор ROLLBACK с указанием имени данной точки. Например:

```
UPDATE emp SET sal = 2000 WHERE ename = 'BLAKE';
SAVEPOINT blakesal;
UPDATE emp SET sal = 1500 WHERE ename = 'CLARK';
SELECT SUM(sal) FROM emp;
ROLLBACK TO SAVEPOINT blakesal;      --отменяет изменения информации
COMMIT;                                --о зарплате служащего Clark.
```

SELECT INTO

```
DECLARE
```

```
    dept_rec      dept%ROWTYPE;
    name          emp.ename%TYPE;
    job_title     empjob%TYPE;
    wages         emp.sal%TYPE;
BEGIN
SELECT * INTO dept_rec
FROM dept
WHERE deptno = 20;
SELECT ename, job, sal INTO name, job_title, wages
FROM emp
WHERE empno = 1440;
...         --остальная часть блока
END;
```

SET TRANSACTION

Некая компания в конце каждого месяца проверяет кол-во имеющихся у нее кораблей и контейнеров. Для этого используется такая последовательность операторов:

```
COMMIT;
SET TRANSACTION READ ONLY;
SELECT COUNT(*) FROM ship;
SELECT COUNT(*) FROM container;
COMMIT;
```

Последний оператор COMMIT служит для завершения транзакции типа READ ONLY.

Каждый оператор SELECT во время транзакции типа READ ONLY (только чтение) выбирает записи из “моментального снимка” базы данных, не замечая изменений, вносимых другими пользователями.

операциц с множествами

Предположим, есть три таблицы типа emp и нужно найти в таблице emp служащих, имеющих такой же оклад, как у Зэйва из Тель-Авива Nat из Ашкелона, Оранит из Рэховота

```
SELECT ename, job, sal
FROM emp
WHERE sal IN
(SELECT sal
FROM tel_aviv
WHERE ename = ‘Ze’ev’
UNION
SELECT sal
FROM Ashqelon
WHERE ename = ‘Nat’
UNION
SELECT sal
FROM Rehovot
WHERE ename = ‘Oranit’);
```

Запросы, использующие операторы множеств UNION, INTERSECT и MINUS должны извлекать одинаковое кол-во столбцов. Типы данных соотв. столбцов в разных запросах должны совпадать.

простое соединение

Для получения списка сотрудников, получающих $3000 или больше, с указанием оклада, отдела и его местонахождения, используем простое соединение.

```
SELECT ename, sal, dname, loc
FROM emp, dept
WHERE emp.deptno = dept.deptno
AND sal >= 3000;
```

Чтобы найти ранг оклада каждого служащего, делаем:

```
SELECT ename, job, grade, sal
FROM emp, salgrade
WHERE sal BETWEEN losal AND hisal
ORDER BY ename
```

подзапрос

Этот подзапрос позволяет повысить оклад каждому зав отделом в Далласе до самой высокой ставки в компании.

```
  UPDATE emp
  SET sal = (SELECT MAX(sal) FROM emp)
WHERE JOB = 'MANAGER'
AND DEPTNO IN (SELECT deptno FROM dept WHERE loc = 'DALLAS');
```

Если подзапрос извлекает только одно значение, то для сравнения с ним используется знак '='. Если же подзапрос извлекает более одного значения, то вместо '=' следует писать слово IN.

Чтобы посмотреть список поощренных сотрудников, неменявших место работы, вводим:

```
SELECT empno, ename
FROM emp
WHERE (job, ename) = (SELECT job, ename FROM bonus);
```

WHERE CURRENT OF

Использование внутри процедуры изменение данных по курсору является предпочтительным в связи с тем, что изменения ключевых полей другим пользователем не приведёт к ошибке при работе программы.

При работе с большими базами данных по курсору следует учитывать размер ROLLBACK сегмента. При необходимости внутри процедуры может использоваться оператор COMMIT для утверждения произведённых изменений и очистки сегмента отката.

Приведенная здесь процедура на языке PL/SQL делает отметки в таблице today, устанавливая для закончившихся операций поле status = DONE (сделано). Подобные действия могут выполняться, к примеру, над таблицей счетовых балансов в конце каждого дня.

```
DECLARE
CURSOR c1 IS SELECT action, acct FROM today
FOR UPDATE OF status;
        the_action   today.act%TYPE;
        acct_num     today.acct%TYPE;
BEGIN
OPEN c1;
LOOP
FETCH c1 INTO the_action, acct_num;
EXIT WHEN c1%NOTFOUND;
...      --операторы, выполняющие какие-то действия
UPDATE today SET status = 'done'
WHERE CURRENT OF c1;
END LOOP;
CLOSE c1;
COMMIT;
END;
```

WHILE

Предположим, что в фонде заработной платы ваших служащих имеется свыше $50 000 и Вы хотите пропорционально увеличить оклад каждого из них. Приведенный в данном примере цикл последовательно увеличивает каждый оклад на 10% до тех пор, пока общая сумма окладов не станет равна или не превысит $50 000.

```
DECLARE
   total_sal    NUMBER(5);
BEGIN
    SELECT SUM(sal) INTO total_sal FROM emp;
    WHILE total_sal < 50000 LOOP
       UPDATE emp SET sal = sal * 1.10;
          SELECT SUM(sal) INTO total_sal FROM emp;
     END LOOP;
```

END;

Приложение

Ключевые слова

Ключевые слова имеют особые значения в SQL и PL/SQL. Они не могут быть использованы в качестве имен идентификаторов (если они не заключены в кавычки).

Слово	Перевод
ADMIN	АДМИНИСТРАЦИЯ
AFTER	ПОСЛЕ
ALLOCATE	РАСПРЕДЕЛИТЕ
ANALYZE	АНАЛИЗИРУЙТЕ
ARCHIVE	АРХИВ
ARCHIVELOG	АРХИВ РЕГИСТРАЦИИ
AUTHORIZATION	РАЗРЕШЕНИЕ
AVG	В СРЕДНЕМ
BACKUP	КОПИЯ
BEGIN	НАЧНИТЕ
BECOME	СТАНЕМ
BEFORE	ПЕРЕД (ПРЕЖДЕ)
BLOK	Блок
BODY	ТЕЛО
CACHE	КЭШ
CANCEL	ОТМЕНА
CASCADE	КАСКАД
CHANGE	ИЗМЕНЕНИЕ
CHARACTER	СИМВОЛ
CHECKPOINT	КОНТРОЛЬНАЯ ТОЧКА
CLOSE	ЗАВЕРШЕНИЕ
COMMIT	ПЕРЕДАЙТЕ(СОВЕРШИТЕ)
COMPILE	КОМПИЛИРУЙТЕ
CONSTRAINT	ОГРАНИЧЕНИЕ
CONTENTS	СОДЕРЖАНИЕ
CONTINUE	ПРОДОЛЖИТЕ
CONTROLFILE	Файл контроля
COUNT	СЧЕТ
CURSOR	КУРСОР
CYCLE	ЦИКЛ
DATABASE	БАЗА ДАННЫХ
DATAFILE	ФАЙЛ ДАННЫХ
DECLARE	ОБЪЯВИТЬ
DISABLE	ОТКЛЮЧИТЬ
DISMOUNT	ДЕМОНТИРОВАТЬ
DOUBLE	ДВОЙНОЕ
EACH	КАЖДЫЙ
ENABLE	ДОПУСТИТЬ
END	КОНЕЦ
ESCAPE	УСКОЛЬЗАТЬ
EVENTS	СОБЫТИЯ
EXCEPT	ЗА ИСКЛЮЧЕНИЕМ
EXCEPTIONS	ИСКЛЮЧЕНИЯ
EXEC	ВЫПОЛНИТЬ (сокращение)
EXPLAIN	ОБЪЯСНИТЬ
EXECUTE	ВЫПОЛНИТЬ
EXTENT	ПРОТЯЖЕННОСТЬ
EXTERNALLY	ВНЕШНЕЕ
FETCH	ВЫБОРКА
FLUSH	ПОТОК
FREE LIST	СВОБОДНЫЙ СПИСОК

FORCE	СИЛА
FOREIGN	ИНОСТРАННОЕ
FOUND	НАЙДЕННЫЙ
GOTO	ПЕРЕЙТИ
GROUPS	ГРУППЫ
INCLUDING	ВКЛЮЧЕНИЕ
INDICATOR	ИНДИКАТОР
INI TRANS	НАЧАЛЬНАЯ СДЕЛКА
INSTANCE	ОБРАЗЕЦ
KEY	КЛАВИША(КЛЮЧ)
LANGUAGE	ЯЗЫК
LAYER	УРОВЕНЬ
LINK	СВЯЗЬ
LISTS	СПИСКИ
LOGFILE	LOGIN FILE Регистрационный файл
MANUAL	РУКОВОДСТВО
MAXDATAFILES	MAXIMUM DATA FILES Максимальное количество файлов данных
MAXINISTANCES	MAXimum Instances МАКСИМАЛЬНОЕ КОЛИЧЕСТВО ИНСТАНЦИЙ ОБРАЩЕНИЙ (Клиентных программ, обращающихся к серверу)
NONE	НИ ОДИН
NOORDER	НЕТ ПОРЯДКА
NORESETLOGS	NORESETLOGS
NORMAL	НОРМАЛЬНОЕ
NOSORT	НЕ Сортированное
NUMERIC	ЧИСЛОВОЕ
OFF	ОТКЛЮЧИТЬ
OLD	СТАРОЕ
ONLY	ТОЛЬКО
OPTIMAL	ОПТИМАЛЬНО
OPEN	ОТКРЫТО
OWN	СОБСТВЕННО
PACKAGE	ПАКЕТ
PARALLEL	ПАРАЛЛЕЛЬНО
PLAN	ПЛАН
PRECISION	ТОЧНОСТЬ
PRIMARY	ПЕРВИЧНОЕ
PRIVATE	ЧАСТНОЕ
PROFILE	ПРОФИЛЬ
QUOTA	КВОТА
READ	ЧИТАЙТЕ
REAL	РЕАЛЬНОЕ
RECOVER	ВОССТАНОВЛЕНИЕ
REFERENCES	ССЫЛКИ
REFERENCING	ССЫЛКА НА
RESETLOGS	RESET LOGS Переустановка регистраций
RESTRICTED	ОГРАНИЧЕННЫЙ
REUSE	ПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ
ROLE	РОЛЬ
ROLLBACK	ОБРАТНАЯ ПЕРЕМОТКА (Отказ от действий)
SAVEPOINT	ТОЧКА СОХРАНЕНИЯ
SCHEMA	СХЕМА ОБЪЕКТОВ
SECTION	РАЗДЕЛ
SEGMENT	СЕГМЕНТ
SEQUENCE	ПОСЛЕДОВАТЕЛЬНОСТЬ
SHARED	ОБЩЕДОСТУПННО
SNAPSHOT	КАДР

SOME	НЕКОТОРЫЕ
SORT	СОРТИРОВКА
SQLERROR	SQL ОШИБКА
STATEMENT_ID	STATEMENT IDentify Утверждение Указатель (Указатель утверждения)
STATISTICS	СТАТИСТИКА
STOP	ОСТАНОВ
STORAGE	ПАМЯТЬ(ХРАНЕНИЕ)
SUM	СУММА
SWITCH	ПЕРЕКЛЮЧАТЕЛЬ
SYSTEM	СИСТЕМА
TABLES	ТАБЛИЦЫ
TABLESPACE	ТАБЛИЧНАЯ ОБЛАСТЬ
TEMPORARY	ВРЕМЕННО
THREAD	НИТЬ
TIME	ВРЕМЯ
TRACING	РАССМОТРЕНИЕ
TRANSACTION	ТРАНЗАКЦИЯ
TRIGGERS	ВЫЗЫВАЕТ
TRUNCATE	УСЕКИТЕ
UNDER	ПОД
UNLIMITED	НЕОГРАНИЧЕННО
UNTIL	ДО
USE	ИСПОЛЬЗОВАТЬ
WHEN	КОГДА
WRITE	ПИШИТЕ
WORK	РАБОТА

Представления словаря данных.

Словарь данных предназначен для хранения информации об объектах и событиях в СУРБД ORACLE, которая доступна пользователям через множество представлений данных. Для получения полного перечня таблиц и представлений словаря данных, выполните запрос в DICTIONARY.

Имя представления	Описание
ACCESSIBLE_COLUMNS	столбцы всех таблиц, представлений и кластеров
ACCESSIBLE_TABLES	таблицы и представления, доступные пользователю
AUDIT_ACTIONS	перекодировка номеров действий в их наименования
ALL_CATALOG	доступные таблицы, представления, синонимы и последовательности.
ALL_COL_COMMENTS	комментарии к столбцам доступных таблиц и представлений.
ALL_COL_GRANTS	права на столбцы, к которым польз. имеет отношение как давший, получивший или владелец , или же получившим является PUBLIC
ALL_COL_GRANTS_MADE	права на столбцы, к которым польз. Имеет отношение как владелец или как давший право
ALL_COL_GRANTS_RECD	права на столбцы, полученные польз. или всеми
ALL_DB_LINKS	доступные пользователю связи с удаленными БД
ALL_DEF_AUDIT_OPTS	подразумеваемые парам. ревизии для таблиц и системы
ALL_INDEXES	описание индексов к доступным таблицам
ALL_IND_COLUMNS	столбцы, включающие индексы к доступным таблицам
ALL_OBJECTS	объекты, доступные пользователю
ALL_SEQUENCES	описания принадлежащих польз. последовательностей
ALL_SYNONYMS	синонимы, доступные пользователю
ALL_TABLES	описания таблиц, доступных пользователю
ALL_TAB_AUDIT_OPTS	парам. ревизии для доступных таблиц и представлений
ALL_TAB_COLUMNS	столбцы всех таблиц, представлений и кластеров
ALL_TAB_COMMENTS	комментарии к доступным пользователю таблицам и представлениям данных.
ALL_TAB_GRANTS	права на объекты, к которым польз. имеет отношение как давший, получивший или владелец , или же получившим является PUBLIC
ALL_TAB_GRANTS_MADE	права польз. и права на его объекты
ALL_TAB_GRANTS_RECD	права на объекты, полученные польз. или всеми

ALL_USERS	информация обо всех пользователях базы данных
ALL_VIEWS	тексты доступных польз. представлений данных
AUDIT_ACTIONS	коды действий и их описание
COLUMN_PRIVILEGES	то же, что и ALL_COL_GRANTS
CONSTRAINT_COLUMNS	доступные столбцы в описаниях ограничений
CONSTRAINT_DEFS	описания ограничений к доступным таблицам
DICTIONARY	описание таблиц и представлений словаря данных
DICT_COLUMNS	столбцы таблиц и представлений словаря данных
TABLE_PRIVILEGES	то же, что и ALL_TAB_GRANTS
USER_AUDIT_CONNECT	ревизионные записи о пользовательских подключениях к БД и отключенных от нее
USER_AUDIT_TRAIL	ревизионные записи,относящиеся к пользователю
USER_CATALOG	доступные таблицы, представления данных, синонимы и последовательности.
USER_CLUSTERS	описания принадлежащих польз. кластеров
USER_CLU_COLUMNS	соответствие между столбцами таблицы и столбцами кластера
USER_COL_COMMENTS	комментарии к столбцам таблиц и представления. данных пользователя
USER_COL_GRANTS	права на столбцы, где польз. является владельцем, давшим право, или же получившим его
USER_COL_GRANTS_MADE	права на столбцы объектов, принадл. польз.
USER_COL_GRANTS_RECD	права на столбцы, предоставленные польз
USER_CROSS_REF	перекрестные ссылки на представления. данных, синонимы и ограничения пользователя
USER_DB_LINKS	принадлежащие польз. связи с удаленными БД
USER_EXTENTS	экстенты, содержащие принадлежащие. польз. сегменты
USER_FREE_SPACE	свободные экстенты в принадлежащей польз. Области хранения
USER_INDEXES	описания принадлежащих пользователю индексов
USER_IND_COLUMNS	индексированные столбцы таблиц польз.
USER_OBJECTS	объекты, принадлежащие пользователю
USER_SEGMENTS	пространство, занятое всеми сегментами БД
USER_SEQUENCES	описания принадлежащих польз. последовательностей
USER_SYNONYMS	частные синонимы пользователя
USER_TABLES	описания принадлежащих польз. таблиц
USER_TABLESPACES	описания доступных областей хранения
USER_TAB_AUDIT_OPTS	парам. ревизии для таблицы и предст.пользователя
USER_TAB_COLUMNS	столбцы таблиц, представлений данных и кластеров польз.
USER_TAB_COMMENTS	комментарии к принадлежащие. польз. таблицам и представлений.
USER_TAB_GRANTS	права на объекты, к которым польз. имеет отношение как давший, получивший или владелец, или же получившим является PUBLIC
USER_TAB_GRANTS_MADE	права на объекты пользователя
USER_TAB_GRANTS_RECD	права на объекты, полученные польз.
USER_TS_QUOTAS	квоты области хранения для польз.
USER_USERS	информация о текущем польз.
USER_VIEWS	тексты принадлежащих польз. представлений данных

Функции обработки дат

Почти все ниже перечисленные ф-ции возвращают значение типа DATE

MONTHS_BETWEEN	возвращает численное значение.
ADD_MONTHS (d, n)	Прибавляет к дате d n месяцев (n = целое).
LAST_DAY(d)	Возвращает последнее число месяца в заданной дате
MONTHS_BETWEEN (d, e)	Возвращает кол-во месяцев между датами d и e.
NEW_TIME(d, a, b)	По заданной дате d во временном поясе a вычисляет сотоветствующее. им дату/время во временном поясе . Для обозначения врем. поясов используются сокращения
NEXT_DAY(d, day)	Вычисляет дату, выпадающую на ближайший день недели (по имени), наступающий после даты d.
SYSDATE	Возвращает текущую дату и время.
ROUND(d, формат)	Округляет дату d в соответствии с указанным форматом

TRUNC(d, формат)	“Обрезает” дату d в соответствии с указанным форматом
TRUNC без формата	“отрезает” от даты время.

Форматирование даты ROUND или TRUNC

CC	век
SCC	век до н.э.
YYYY/YEAR/YYY	год
Q	квартал (округляется по 16-тым и 2-м частям мес.)
MONTH/MON/MM	месяц
WW	неделя года
W	неделя мес.
DDD/DD/J	число
DAY/DY/D	ближайшее воскресенье
HH/HH12/HH24	час
MI	минута

Наименование	Windows 95 Значение по умолчанию	Windows 95 Максимальное значение
ARRAYSIZE	20 строк	100 строк
CHARWIDTH	80 символов	65,535 символов
DATEWIDTH	9 символов	65,535 символов
LONGWIDTH	80 символов	65,535 символов
MAXDATA	20 килобайт	65,535 символов
NUMWIDTH	10 точек	65,535 точек

Встраиваемые функции

Error (Ошибки)	Number (Числовые)	Character (Символьные)
SQLCODE	ABS	ASCII
SQLERRM	CEIL	CHR
COS	CONCAT	HEXTORAW
COSH	INITCAP	RAWTOHEX
EXP	INSTR	ROWIDTOCHAR
FLOOR	INSTRB	TO_CHAR
LN	LENGTH	TO_DATE
LOG	LENGTHB	TO_LABEL
MOD	LOWER	TO_MULTI_BYTE
POWER	LPAD	TO_NUMBER
ROUND	LTRIM	TO_SINGLE_BYTE
SIGN	NLS_INITCAP	
SIN	NLS_LOWER	
SINH	NLS_UPPER	
SQRT	NLSSORT	
TAN	REPLACE	
TANH	RPAD	
TRUNC	RTRIM	
SOUNDEX		
SUBSTR		
SUBSTRB		
TRANSLATE		
UPPER		

Conversion Преобразования	Date Даты	Misc Дополнитель.
CHARTOROWID	ADD_MONTHS	DECODE
CONVERT	LAST_DAY	DUMP
MONTHS_BETWEEN	GREATEST	
NEW_TIME	GREATEST_LB	
NEXT_DAY	LEAST	
ROUND	LEAST_LB	
SYSDATE	NVL	
TRUNC	UID	
	USER	
	USERENV	
	VSIZE	

Групповые функции

Групповые функции языка SQL производят вычисления с группами строк. Их можно использовать только в запросах и подзапросах.

AVG(D|A выражение)	Вычисляет среднее арифметическое значение выражения по строкам.
COUNT(D|A выражение)	Вычисляет кол-ва строк, в которых 'выражение' не пусто.
COUNT(*)	Вычисляет полное кол-во строк, включая пустые значения.
MAX(D|A выражение)	Находит максимальное знач. выраж. по строкам.
MIN(D|A выражение)	Находит минимальное знач. выраж. по строкам.
STDDEV(D|A выражение)	Вычисляет стандартное отклонения.
SUM(D|A выражение)	Вычисляет сумму выражений по строкам.
VARIANCE(D|A выражение)	Вычисляет дисперсию выражения по строкам D|A = DISTINCT или ALL (подразумевается ALL).

Пустые значения (NULL) при вычислениях игнорируются (кроме функции COUNT(*)).

Дополнительные параметры настройки

Имя параметра	Значение по умолчанию	Границы значений
BACKGROUND_DUMP_DEST	%RDBMSnn%\TRACE\	Любая существующая директория
COMMIT_POINT_STRENGTH	1	0 - 255
CONTROL_FILES	%ORACLE_HOME%\DATABASE\CTL1%ORACLE_SID%.ORA	Любая существующая директория и файл
DB_BLOCK_SIZE	2048	512 - 8192
DB_FILES	32	254, не превышающие DB_BLOCK_BUFFERS
DB_FILE_MULTIBLOCK_READ_COUNT	4	1 - 32
DB_FILE_SIMULTANEOUS_WRITES	4	1 - 24
DISTRIBUTED_TRANSACTIONS	25 * TRANSACTIONS	0 - TRANSACTIONS
LOG_ARCHIVE_BUFFER_SIZE	127	1 - 127
LOG_ARCHIVE_BUFFERS	4	1 - 8
LOG_ARCHIVE_DEST	%RDBMSnn%\	Любая существующая директория
LOG_ARCHIVE_FORMAT	ARC%S.%T	Действительный формат файла
LOG_CHECKPOINT_INTERVAL	8000	2 - Не ограничено
LOG_FILES	255	2 - 255
LOG_SMALL_ENTRY_MAX_SIZE	800	0 - Не ограничено
NLS_LANGUAGE	AMERICAN	Любой существующий язык
NLS_SORT	(Language dependent)	(Language dependent)
NLS_TERRITORY	AMERICA	Любая существующая территория

OPEN_CURSORS	50	1 - Не ограничено
OS_AUTHENT_PREFIX	OPS$	Любая последовательность символов
PROCESSES	25	3 - Не ограничено
REMOTE_LOGIN_PASSWORDFILE	shared	shared, exclusive, none
SHARED_POOL_SIZE	3,500,000	300K - Не ограничено
SORT_AREA_SIZE	65536	0 - Не ограничено
SORT_READ_FAC	20	0 - Не ограничено
SORT_SPACEMAP_SIZE	512	0 - Не ограничено
TEMPORARY_TABLE_LOCKS	SESSIONS	0 - Не ограничено
TRANSACTIONS_PER_ROLLBACK_SEGMENT	30	1 - 255
USER_DUMP_DEST	%RDBMSnn%\TRACE	Любая существующая директория

Обозначение ограничений в ALL_CONSTRAINTS

Тип ограничения	Буква
PRIMARY KEY	P
UNIQUE KEY	U
FOREIGN KEY	R
CHECK, NOT NULL	C

Привилегии

Наименование привилегии	Перевод на русский	Выполняемая операция
ALTER ANY CLUSTER	ИЗМЕНИТЕ ЛЮБОЙ КЛАСТЕР	Разрешает изменять любой кластер в любой схеме объектов.
ALTER ANY INDEX	ИЗМЕНИТЕ ЛЮБОЙ ИНДЕКС	Разрешает изменять любой индекс в любой схеме объектов
ALTER ANY PROCEDURE	ИЗМЕНИТЕ ЛЮБУЮ ПРОЦЕДУРУ	Разрешает изменять любую сохраненную процедуру, функцию, или пакет в любой схеме объектов.
ALTER ANY ROLE	ИЗМЕНИТЕ ЛЮБУЮ РОЛЬ	Разрешает изменять любую роль в базе данных.
ALTER ANY SEQUENCE	ИЗМЕНИТЕ ЛЮБУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ	Разрешает изменять любую последовательность в базе данных.
ALTER ANY SNAPSHOT	ИЗМЕНИТЕ ЛЮБОЙ КАДР	Разрешает изменять любой кадр в базе данных.
ALTER ANY TABLE	ИЗМЕНИТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает изменять любую таблицу или представление в схеме объектов.
ALTER ANY TRIGGER	ИЗМЕНИТЕ ЛЮБОЙ TRIGGER	Разрешает , отключать, или компилировать любой триггер в любой схеме объектов.
ALTER DATABASE	ИЗМЕНИТЕ БАЗУ ДАННЫХ	Разрешает изменять базу данных.
ALTER PROFILE	ИЗМЕНИТЕ ПРОФИЛЬ	Разрешает изменять профили.
ALTER RESOURCE COST	ИЗМЕНИТЕ СТОИМОСТЬ РЕСУРСА	Разрешает устанавливать ограничения для ресурсов сеанса.
ALTER ROLLBACK SEGMENT	ИЗМЕНИТЕ СЕГМЕНТ ОБРАТНОЙ ПЕРЕМОТКИ	Разрешает изменять сегменты отказа от выполняемых транзакций
ALTER SESSION	ИЗМЕНИТЕ СЕАНС	Разрешает выдавать утверждения изменение текущей сессии.
ALTER SYSTEM	ИЗМЕНИТЕ СИСТЕМУ	Разрешает выдавать утверждения ALTER SYSTEM.
ALTER TABLESPACE	ИЗМЕНИТЕ ОБЛАСТЬ ХРАНЕНИЯ	Разрешает изменять область хранения таблиц.

ALTER USER	ИЗМЕНИТЕ ПОЛЬЗОВАТЕЛЯ	Разрешает изменять параметры для любого пользователя. Эта привилегия разрешает изменять пароль другого пользователя или опознавательный метод, назначать квоты на любой табличной области, устанавливать значение по умолчанию и временную табличную область, и назначать профиль и заданные по умолчанию роли.
ANALYZE ANY	АНАЛИЗИРУЙТЕ ЛЮБОГО	Разрешает анализировать любую таблицу, кластер, или индекс в любой схеме объектов.
AUDIT ANY	РЕВИЗИЯ ЛЮБОЙ	Разрешает ревизовать любой объект в любой схеме объектов.
AUDIT SYSTEM	КОНТРОЛЬНАЯ СИСТЕМА	Разрешает выдавать РЕВИЗИЮ.
BACKUP ANY TABLE	РЕЗЕРВИРУЙТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает использовать Экспорт данных.
BECOME USER	СТАЛ ПОЛЬЗОВАТЕЛЕМ	Разрешает стать другим пользователем.
COMMENT ANY TABLE	КОММЕНТАРИИ ДЛЯ ЛЮБОЙ ТАБЛИЦЫ	Разрешает устанавливать комментарии для любой таблицы, представления, или столбца в любой схеме объектов.
CREATE ANY CLUSTER	СОЗДАЙТЕ ЛЮБОЙ КЛАСТЕР	Разрешает создавать кластер в любой схеме объектов..
CREATE ANY INDEX	СОЗДАЙТЕ ЛЮБОЙ ИНДЕКС	Разрешает создавать индекс в любой схеме объектов на любой таблице в любой схеме объектов.
CREATE ANY PROCEDURE	СОЗДАЙТЕ ЛЮБУЮ ПРОЦЕДУРУ	Разрешает создавать сохраненные процедуры, функции, и пакеты в любой схеме объектов.
CREATE ANY SEQUENCE	СОЗДАЙТЕ ЛЮБУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ	Разрешает создавать последовательность в любой схеме объектов.
CREATE ANY SNAPSHOT	СОЗДАЙТЕ ЛЮБОЙ КАДР	Разрешает создавать кадры в любой схеме объектов.
CREATE ANY SYNONYM	СОЗДАЙТЕ ЛЮБОЙ СИНОНИМ	Разрешает создавать частные синонимы в любой схеме объектов.
CREATE ANY TABLE	СОЗДАЙТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает создавать таблицы в любой схеме объектов. Владелец схемы объектов, содержащей таблицу, должен иметь квоту пустой области в табличной области.
CREATE ANY TRIGGER	СОЗДАЙТЕ ЛЮБОЙ ВЫЗЫВАЕТ	Разрешает создавать триггер базы данных в любой схеме объектов, связанных с любой таблицей.
CREATE ANY VIEW	СОЗДАЙТЕ ЛЮБОЕ ПРЕДСТАВЛЕНИЕ	Разрешает создавать представления в любой схеме объектов.
CREATE CLUSTER	СОЗДАЙТЕ КЛАСТЕР	Разрешает создавать кластеры в собственной схеме объектов.
CREATE DATABASE LINK	СОЗДАЙТЕ СВЯЗЬ БАЗЫ ДАННЫХ	Разрешает создавать частные связи базы данных в собственной схеме объектов.
CREATE PROCEDURE	СОЗДАЙТЕ ПРОЦЕДУРУ	Разрешает создавать сохраненные процедуры, функции, и пакеты в собственной схеме объектов.
CREATE PROFILE	СОЗДАЙТЕ ПРОФИЛЬ	Разрешает создавать профили (настройку конфигурации пользователя).
CREATE PUBLIC DATABASE LINK	СОЗДАЙТЕ ОБЩУЮ СВЯЗЬ БАЗЫ ДАННЫХ	Разрешает создавать общие связи базы данных.
CREATE PUBLIC SYNONYM	СОЗДАЙТЕ ОБЩИЙ СИНОНИМ	Разрешает создавать общие синонимы.
CREATE ROLE	СОЗДАЙТЕ РОЛЬ	Разрешает создавать роли.

CREATE ROLLBACK SEGMENT	СОЗДАЙТЕ СЕГМЕНТ ОБРАТНОЙ ПЕРЕМОТКИ	Разрешает создавать сегменты отказа от действий пользователя.
CREATE SEQUENCE	СОЗДАЙТЕ ПОСЛЕДОВАТЕЛЬНОСТЬ	Разрешает создавать последовательности в собственной схеме объектов.
CREATE SESSION	СОЗДАЙТЕ СЕАНС	Разрешает соединяться с базой данных.
CREATE SNAPSHOT	СОЗДАЙТЕ КАДР	Разрешает создавать кадры в собственной схеме объектов.
CREATE SYNONYM	СОЗДАЙТЕ СИНОНИМ	Разрешает создавать синонимы в собственной схеме объектов.
CREATE TABLE	СОЗДАЙТЕ ТАБЛИЦУ	Разрешает создавать таблицы в собственной схеме объектов. Чтобы создавать таблицу необходимо иметь квоту пустого места в текущей табличной области.
CREATE TABLESPACE	СОЗДАЙТЕ TABLESPACE	Разрешает создавать табличную область. Табличная область - место хранения информации приложения
CREATE TRIGGER	СОЗДАЙТЕ ВЫЗЫВАЮТ	Разрешает создавать процедуры с автозапуском (триггеры) в собственной схеме объектов.
CREATE USER	СОЗДАЙТЕ ПОЛЬЗОВАТЕЛЯ	Разрешает создавать пользователей. Эта привилегия также Разрешает создателю назначать квоты использования табличных областей , устанавливать значение по умолчанию и временную табличную область , и назначать профиль как часть утверждения CREATE USER.
CREATE VIEW	СОЗДАЙТЕ ПРЕДСТАВЛЕНИЕ	Разрешает создавать представления в собственной схеме объектов.
DELETE ANY TABLE	УДАЛИТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает удалять строки из таблиц или представлений в любой схеме объектов. А так же очищать таблицы в любой схеме объектов.
DROP ANY CLUSTER	ВЫБРОСИТЕ ЛЮБОЙ КЛАСТЕР	Разрешает уничтожать кластеры в любой схеме объектов.
DROP ANY INDEX	ВЫБРОСИТЕ ЛЮБОЙ ИНДЕКС	Разрешает удалять индексы в любой схеме объектов.
DROP ANY PROCEDURE	ВЫБРОСИТЕ ЛЮБУЮ ПРОЦЕДУРУ	Разрешает удалять сохраненные процедуры, функции, или пакеты в любой схеме объектов.
DROP ANY ROLE	ВЫБРОСИТЕ ЛЮБУЮ РОЛЬ	Разрешает удалить роли.
DROP ANY SEQUENCE	ВЫБРОСИТЕ ЛЮБУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ	Разрешает удалить последовательности в любой схеме объектов.
DROP ANY SNAPSHOT	ВЫБРОСИТЕ ЛЮБОЙ КАДР	Разрешает удалить кадры в любой схеме объектов.
DROP ANY SYNONYM	ВЫБРОСИТЕ ЛЮБОЙ СИНОНИМ	Разрешает удалить частные синонимы в любой схеме объектов.
DROP ANY TABLE	ВЫБРОСИТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает удалить таблицы в любой схеме объектов.
DROP ANY TRIGGER	СНИЖЕНИЕ ЛЮБОЙ ВЫЗЫВАЕТ	Разрешает удалять триггер в любой схеме объектов.
DROP ANY VIEW	ВЫБРОСИТЕ ЛЮБОЕ ПРЕДСТАВЛЕНИЕ	Разрешает удалить представления в любой схеме объектов
DROP PROFILE	ВЫБРОСИТЕ ПРОФИЛЬ	Разрешает удалять профили.
DROP PUBLIC DATABASE LINK	ВЫБРОСИТЕ ОБЩУЮ СВЯЗЬ БАЗЫ ДАННЫХ	Разрешает удалять общие связи базы данных.
DROP PUBLIC SYNONYM	ВЫБРОСИТЕ ОБЩИЙ СИНОНИМ	Разрешает удалять общие синонимы.

DROP ROLLBACK SEGMENT	ВЫБРОСИТЕ СЕГМЕНТ ОБРАТНОЙ ПЕРЕМОТКИ	Разрешает удалять сегменты обратной перемотки.
DROP TABLESPACE	ВЫБРОСИТЕ ТАБЛИЧНУЮ ОБЛАСТЬ	Разрешает удалять основную область хранения таблиц.
DROP USER	ПОЛЬЗОВАТЕЛЬ СНИЖЕНИЯ	Разрешает удалять пользователей.
EXECUTE ANY PROCEDURE	ВЫПОЛНИТЕ ЛЮБУЮ ПРОЦЕДУРУ	Разрешает выполнять процедуры или функции или ссылаться на общие переменные пакета в любой схеме объектов.
FORCE ANY TRANSACTION	ВЫНУДИТЕ ЛЮБУЮ ТРАНЗАКЦИЮ	Разрешает устанавливать приоритет выполнения транзакции в любой схеме обьектов. Применяется при использовании распределённых транзакций и работе с совокупностью взаимосвязанных серверов
FORCE TRANSACTION	ТРАНЗАКЦИЯ СИЛЫ	Разрешает устанавливать приоритет выполнения транзакции в локальной схеме обьектов.
GRANT ANY PRIVILEGE	ПРЕДОСТАВЬТЕ ЛЮБУЮ ПРИВИЛЕГИЮ	Разрешает предоставлять любую привилегию системы.
GRANT ANY ROLE	ПРЕДОСТАВЬТЕ ЛЮБУЮ РОЛЬ	Разрешает предоставлять любую роль в базе данных.
INSERT ANY TABLE	ВСТАВЬТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает вставлять строки в таблицы и представления в любой схеме объектов.
LOCK ANY TABLE	БЛОКИРУЙТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает блокировать таблицы и представления в любой схеме объектов.
MANAGE TABLESPACE	УПРАВЛЯЙТЕ TABLESPACE	Разрешает брать табличную область автономно и интерактивно и начинать и устанавливать копии табличных областей.
READUP	READUP	Разрешает сделать запрос, данные, имеющие доступ классифицируют выше чем метка сеансов. Эта привилегия только доступна в доверительных системах Oracle7.
RESTRICTED SESSION	ОГРАНИЧЕННЫЙ СЕАНС	Разрешает обеспечивать вход в систему после того, как основная база запущена , используя команду STARTUP RESTRICT администраторной станции.
SELECT ANY SEQUENCE	ВЫБЕРИТЕ ЛЮБУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ	Разрешает ссылаться на последовательности в любой схеме объектов.
SELECT ANY TABLE	ВЫБЕРИТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает сделать запрос таблиц, представлений(видов), или кадров в любой схеме объектов.
UNLIMITED TABLESPACE	НЕОГРАНИЧЕННЫЙ TABLESPACE	Разрешает использовать неограниченное количество пространства в табличной области . Эта привилегия отменяет любые специфические назначенные квоты.. Запрещается предоставлять эту привилегию системы ролям.
UPDATE ANY TABLE	МОДИФИЦИРУЙТЕ ЛЮБУЮ ТАБЛИЦУ	Разрешает модифицировать строки в таблицах и представлениях в любой схеме объектов.
WRITEDOWN	WRITEDOWN	Разрешает создавать, изменять, и удалять объекты схеме объектов, и вставлять, модифицировать, и удалять строки, имеющие доступ классифицирует ниже чем метка сеансов. Эта привилегия только доступна в доверительных системах Oracle7.

WRITEUP	WRITEUP	Разрешает создавать, изменять, и удалять объекты схеме объектов, и вставлять, модифицировать, и удалять строки, имеющие доступ классифицирует выше чем метка сеансов. Эта привилегия только доступна в доверительных системах Oracle7.

Дополнительные примечания

Ниже приведены наиболее характерные ошибки, встречаемые при работе с ORACLE у неподготовленного пользователя.

- Не вставляйте в имена объектов пробелы. Если надо разделить можно использовать символ нижнего подчёркивания (имена вида MM_KKK_CCC)
- Сохранённые процедуры SQL файла запускать только по команде @<имя файла SQL>
- Не вводите несколько команд SQL в один буфер SQL
- Активнее используйте параметры командного файла (&1 , &2 .... и запуск файла вида @abc.sql ffff hhhhh)
- Для устранения выдачи сообщений на экран вводить директиву SET TERMOUT OFF;
- Для устранения выдачи на экран режима проверки автоподстановки (&<Имя переменной>) необходимо использовать директиву SET VERIFY OFF;
- Для сохранения текущего буфера на диск использовать директиву SAVE <Имя файла SQL> При этом данная команда обеспечивает как создание нового файла ,так и дополнение текущей командой. Однако для добавления необходимо использовать ключ /APP
- Активнее используйте режим добавления буфера SQL в файл данных (Опция SAVE в меню SQL*PLUS)
- Доступ к функциям осуществляется только в режиме PL/SQL или по директиве EXECUTE <Имя функции>
- Примечания указываются символами – при этом все символы после – игнорируются
- Команды SQL, введенные Вами, хранятся в буфере SQL.
- Команда может располагаться как на одной, так и на нескольких строках.
- Чтобы закончить ввод и выполнить команду надо ввести точку с запятой (;), пустую строку или косую черту (/) в отдельной строке.
- Перед каждой дополнительной строкой SQL*Plus высвечивает соответствующий ей номер.
- Чтобы записать содержимое буфера в файл, введите команду SAVE <Имя файла>
- Чтобы выполнить команду, находящуюся в буфере SQL следует (в ответ на приглашение SQL*Plus-а) ввести команду RUN или косую черту (/).
- Программы (блоки) на PL/SQL также хранятся в буфере SQL. Чтобы закончить ввод блока PL/SQL следует ввести на отдельной строке точку (.).
- Выполнить находящийся в буфере блок можно так же, как и команду SQL—введя RUN или косую черту (/).
- В тексте блоков PL/SQL и команд SQL можно вставлять комментарии в виде /* комментарий */ или при помощи оператора REMARK. (В Personal Oracle-7 не рекомендуется)
- В блоках PL/SQL можно начать комментарий двумя минусами (--). Такой комментарий заканчивается вместе со строкой.
- Команды SQL*Plus можно заканчивать символом ; (точка с запятой), но это не обязательно.
- Команды SQL*Plus не сохраняются в буфере SQL.
- Для создания файлов, содержащих команды SQL*Plus, используйте команду EDIT. При этом будет вызван текстовый редактор Notepad. (Для изменения применяемого текстового редактора - смотри команду SET)
- Окончание редактирования буфера SQL по команде Exit в редакторе Notepad
- Не рекомендуется сохранять из текстового редактора Notepad , (Вызванного по команде EDIT SQL*Plus) в файл.

- Файл команд запускается командой START или @.
- Oracle-7 не поддерживает использование длинных имен файлов, поэтому нельзя использовать длинные имена файлов и директорий (имя файла - 8 знаков до точки и 3 знака на тип файла)
- Все имена объектов в Oracle-7 должны вводиться на английском языке без пробелов. Допускается использование нижнего подчеркивания.
- Можно также использовать комментарии типа /*комментарий */, но они не могут быть расположены в одной строке с командой SQL*Plus—они долѓны стоять на отдельных строках. (В Personal Oracle-7 использовать не рекомендуется)
- Почти везде, где идёт вопрос о создании объекта ORACLE можно добавить ключевое слово OR REPLACE. При этом объект будет заменён.
- Длинные команды SQL*Plus можно разбить на несколько строк, используя символ продолжения - (минус) в конце каждой не последней строки. Строка, не законченная символом - заканчивает команду и вызывает ее выполнение.
- В процедурах и функциях необходимо использовать ROLLBACK в обработке исключительных ситуаций.
- При использовании Директивы CURRENT OF имя_курсора в условиях отбора WHERE операторов UPDATE или DELETE необходимо в процедуре в операторе DECLARE объявить курсор с ключом FOR UPDATE; Для изменения данных в процедуре по мере возможности рекомендуется использовать обновление данных по курсору.
- В процедуре и функции необходимо в обязательном порядке использовать оператор COMMIT и ROLLBACK . Причем ROLLBACK необходимо применять в EXCEPTION блоке для всех аварийных ситуаций
- Для издания команд администратора базы данных (DBA) необходимо использовать SQLDBA.EXE. или SQLDBA72.72 (Директория ORAWIN95\BIN) В режиме SQL*Plus 3.2 команды администратора базы данных не работают.(не хватает привилегий)
- При использовании в процедуре или функции административной утилиты DBMS_OUTPUT.PUT_LINE(‘текст’) необходимо использовать процедуру DBMS_OUTPUT.ENABLE(size) для установки размера буфера промежуточного хранения данных. Данная команда не явным образом очищает содержимое буфера вывода на экран. Без её применения возможна ошибка переполнения буфера вывода.(максимальный размер 1 000 000 байт)
- В данных типа DATE (Дата) можно хранить не только саму дату но и время. По умолчанию сохранённое, время не отображается . Для её получения необходимо явным образом преобразовать с помощью функции TO_CHAR пример: TO_CHAR(SYSDATE, ‘MM-DD-YYYY HH24:MI:SS’) Для преобразования из формата внешнего представления даты и времени в формат внутреннего хранения необходимо применять функцию TO_DATE
- Для редактирования имени объекта связи (SID) для WINDOWS-95 используется программа REGEDIT имя которой нужно набрать в меню RAN. Имя объекта связи по умолчанию - ORCL
- При назначении ролей у пользователя необходимо использовать защищённые роли. Роли должны быть назначены.
- Используйте при создании пользователя назначение ему табличной области по умолчанию. (DEFAULT TABLESPACE). После этого создание любого обьекта будет осуществлено в указанной табличной области.
- Для выравнивания колонки при применении оператора SELECT с колонкой VARCHAR2 используйте RPAD
- Для обеспечения корректной работы с базой данных в ASP ADO необходимо устанавливать TAG HTML:

<!-- METADATA NAME="Microsoft ActiveX Data Objects 2.5 Library" TYPE="TypeLib" UUID="{00000205-0000-0010-8000-00AA006D2EA4} - ->

Данное издание будет отредактировано по Вашим вопросам. Направляйте их: webmaster@deltacom.co.il На наиболее интересные вопросы автор ответит Вам на e-Mail

Данное издание не касается особенностям применения Oracle-Server в ADO и ASP. Этой теме будет

посвещено отдельное справочное пособие.

Содержание: