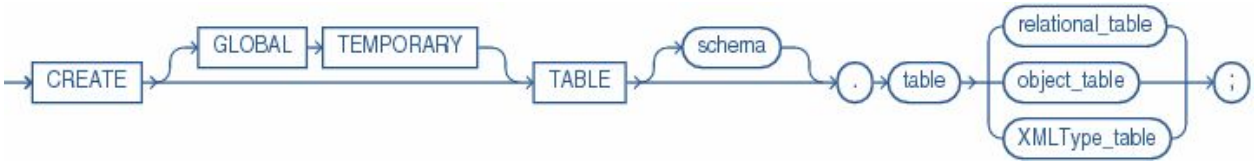
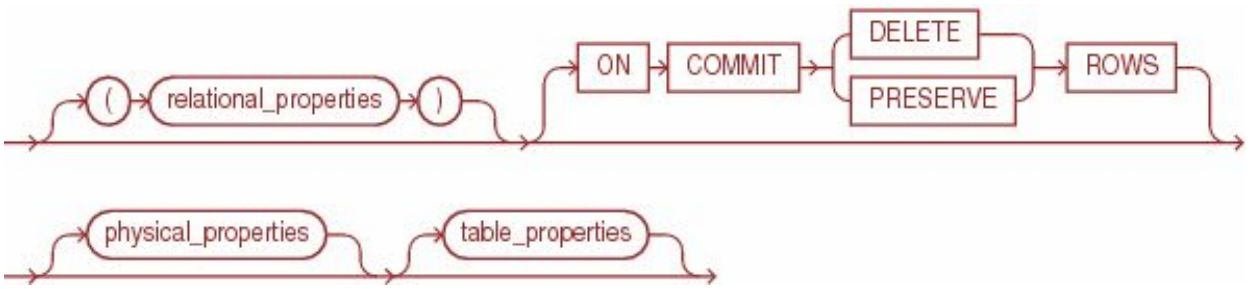
Оглавление

CREATE TABLE

Синтаксис:

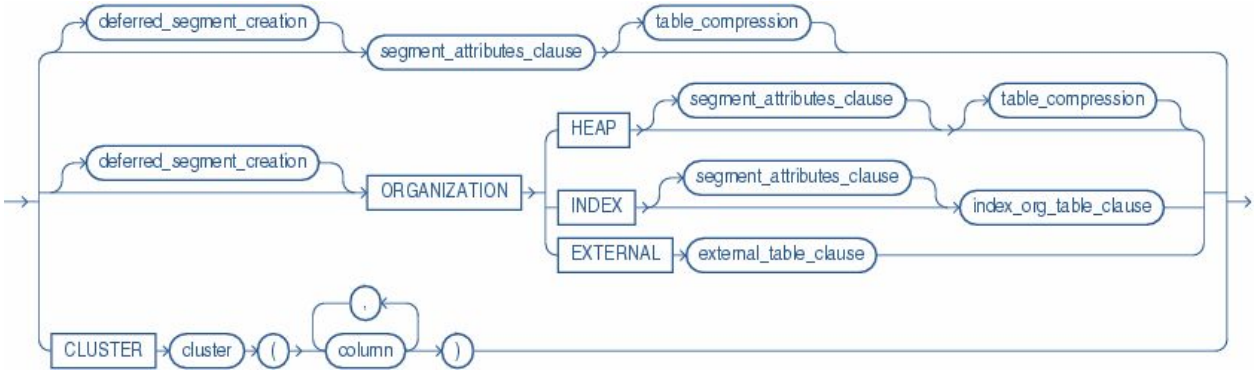


relational_table:



relational_properties – описание столбцов (Имя, Тип данных, Default, (Not) Null, Constraints).

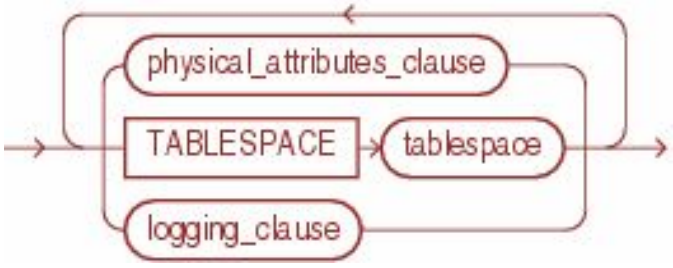
physical_properties



deferred_segment_creation – SEGMENT CREATION DEFERRED/IMMEDIATE. DEFERRED- сегмент создается при первом INSERT.

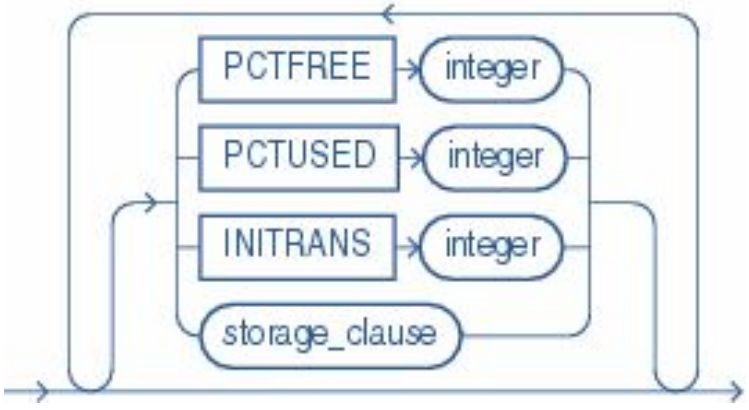
Почти все физические атрибуты для партиционированных таблиц могут быть заданы индивидуально для разделов (а могут и на уровне таблицы, тогда это влияет на все разделы).

segment_attributes_clause:



logging – будут ли операции DML записаны в Redo Log (значения: LOGGING или NOLOGGING)

physical_attributes_clause



pctfree, pctused, initrans

PCTFREE – сколько % свободного места для последующих UPDATE нужно оставить в блоке при выполнении INSERT. Значение от 0 до 99 (default 10)

PCTUSED – минимальный % используемого места в блоке. Значение от 0 до 99 (default 40). Нельзя указывать для index-organized tables

Сумма PCTFREE и PCTUSED должна быть <= 100. Вместе эти параметры используются для оптимизации использования дискового пространства.

Если задать маленькое значение PCTFREE и часто выполнять UPDATE – с большой долей вероятности строки будут перемещаться из текущего в другие блоки, а это относительно медленный процесс – быстрее просто обновить строку в блоке, чем фактически сделать DELETE (убрать её из блока) + INSERT (вставить в другой блок).

Но маленькое значение (например, PCTFREE = 2) для экономии дискового пространства имеет смысл делать для архивных таблиц, или таблиц логов – в них редко делаются UPDATE.

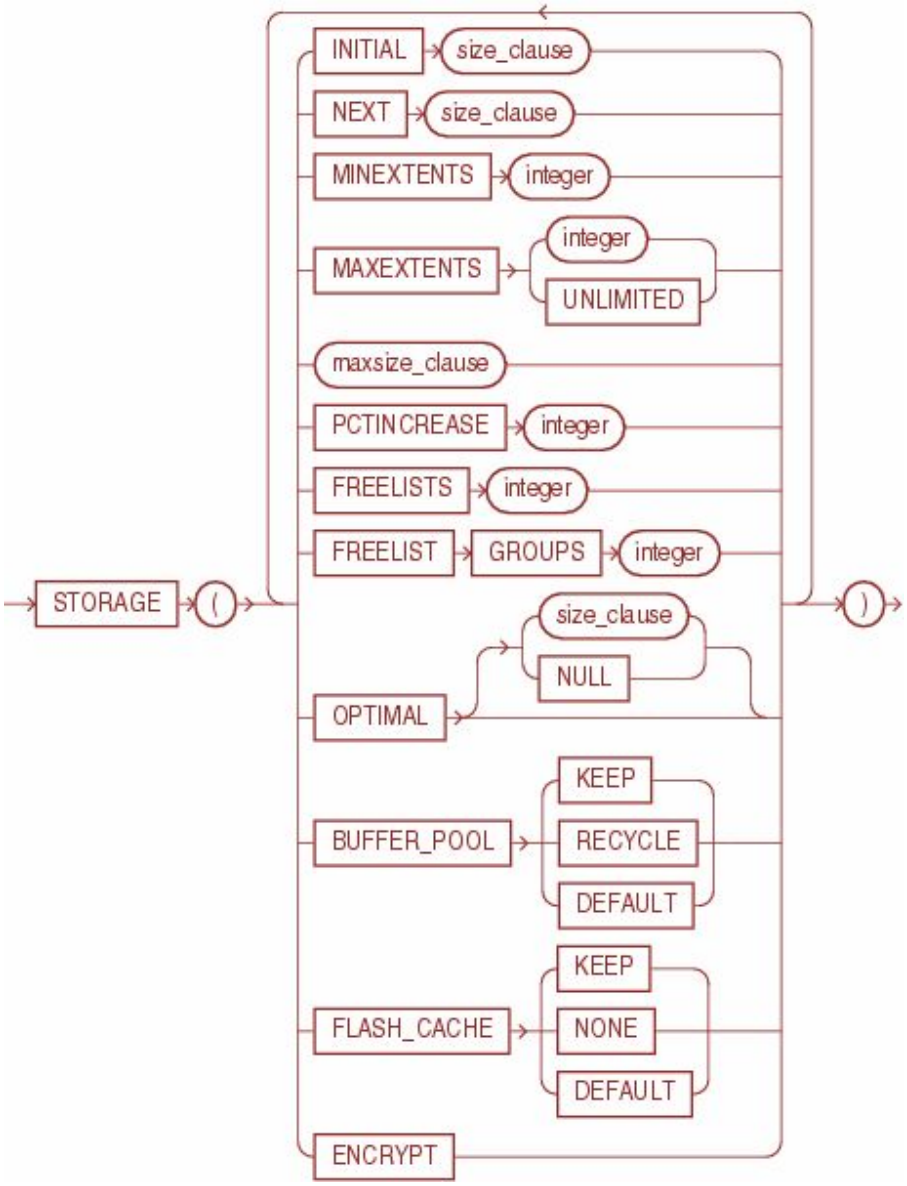
INITRANS – сколько слотов транзакций изначально выделяется в блоке. Значение от 1 до 255. По умолчанию 1, за исключением:

- для кластера таблиц – максимум из 2 и INITRANS tablespace'а, в котором находится кластер
- для индекса - 2

В общем случае не рекомендуется задавать явно, используем дефолтное значение.

Слот транзакции – это место под некий маркер, который транзакция ставит в блок для пометки того факта, что она меняет этот блок. Занимает около 23 байт.

storage_clause



MAXSIZE – максимальный размер объекта (UNLIMITED – без ограничений)

MAXEXTENTS – используется только для dictionary-managed tablespaces

При создании сегмента для вычисления его первоначального размера используются параметры INITIAL, MINEXTENTS, NEXT и PCTINCREASE. При дальнейшем выделении новых экстентов они игнорируются.

INITIAL – размер первого экстента объекта, в зависимости от *allocation_type*:

- UNIFORM – дефолтный размер экстента берется с tablespace, выделяется нужное количество
- другие – может быть выделено 64K, 1M, 8M, или 64M. Из них берем ближайшее меньшее, выделяем нужное количество таких экстентов (4M = 1M+...)

Нельзя указывать в ALTER.

NEXT – размер следующего выделяемого экстента:

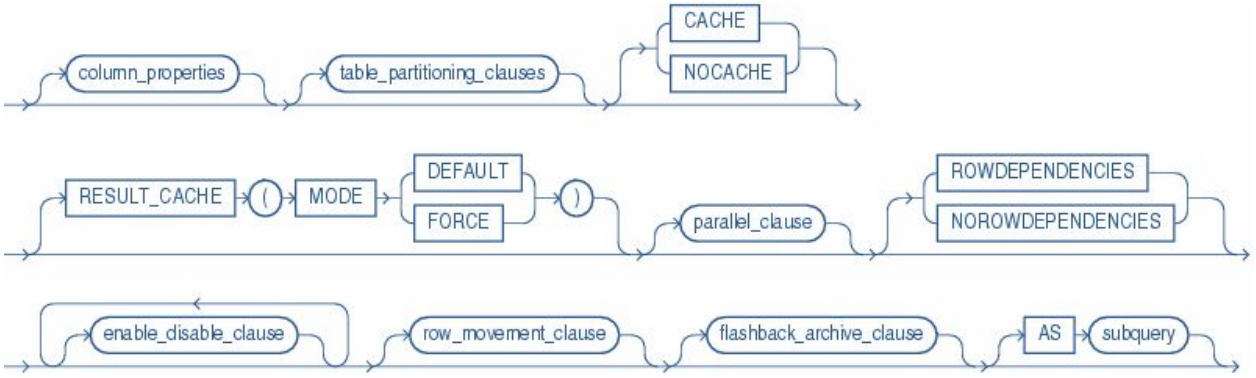
- UNIFORM – также игнорируется, значение подтягивается с tablespace
- другие – Oracle сам вычисляет (заданное пользователем игнорируется)

PCTINCREASE – на сколько % следующий экстент при выделении больше последнего (используется только для dictionary-managed tablespaces).

MINEXTENTS – минимальное количество экстентов объекта. Общее место при создании будет вычислено как INITIAL * MINEXTENTS. С помощью ALTER можно уменьшить этот параметр таблицы, но не увеличить. Это может быть полезно, например, перед использованием TRUNCATE … DROP STORAGE (чтобы сегмент занял минимальное количество экстентов после этого).

Buffer pool – в каком пуле будут лежать закэшированные блоки таблицы (DEFAULT – обычный, где горячие держатся дольше (LRU); KEEP – блоки вообще не вытесняются (пока хватает места), может быть полезно для справочников; RECYCLE – блоки вытесняются сразу после использования).

table_properties



AS subquery – запрос, результат которого будет вставлен в таблицу сразу при создании. При таком подходе можно не указывать явно перечень полей, он сформируется из запроса (а также типы данных и их размерности).

Column_properties – например, тут может быть описание полей типа LOB.

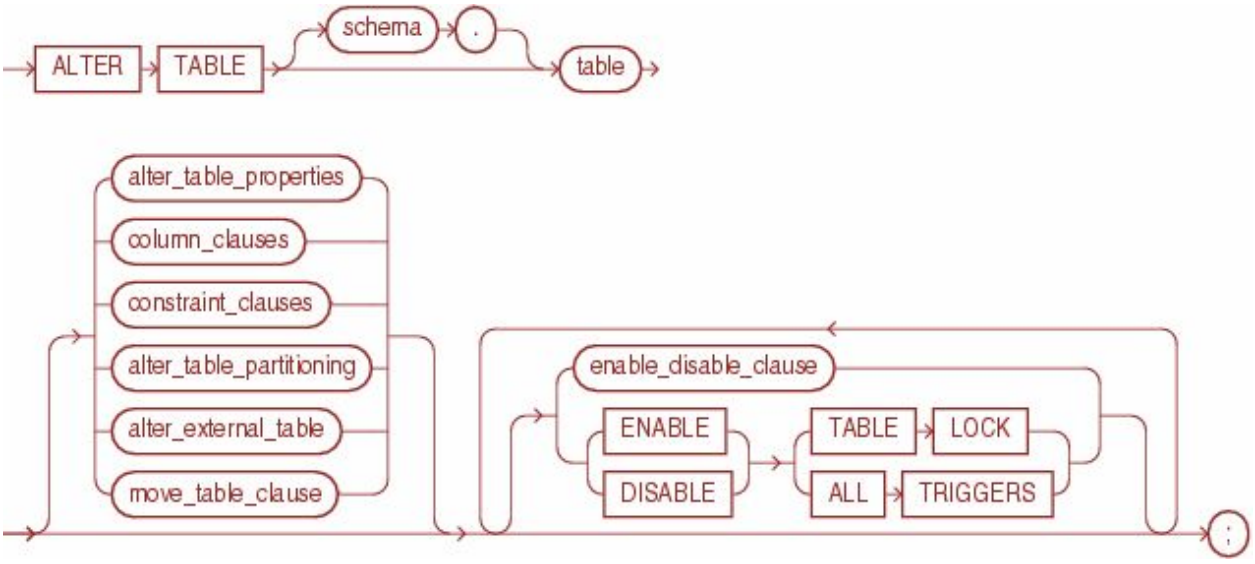
CACHE – блоки помещаются в начало LRU-листа как наиболее часто используемые (полезно для справочников). По дефолту NOCACHE.

RESULT_CACHE – описание того, как результаты запросов должны хранить в result cache.

enable_disable_clause – имеет отношение к изменению constraints (можно создать таблицу с неактивным constraint).

flashback_archive_clause – чтобы таблицу можно было восстановить после DROP.

ALTER TABLE



alter_table_properties - изменение атрибутов таблицы, аналогично созданию (*pctfee, pctused, initrans, logging, cache, result_cache, parallel, row_movement*), а также:

- RENAME TO ... – переименование таблицы
- SHRINK SPACE – уменьшение места, занятого таблицей (уменьшение HWM, см. далее)

column_clauses – добавление, удаление и изменение полей таблицы, в том числе переименование

constraint_clauses – добавление, изменение, переименование и удаление constraints

alter_table_partitioning – изменение способа партиционирования таблицы

move_table_clause – изменение атрибутов сегмента таблицы, в том числе перемещение в другой TS

enable_disable_clause – включение/выключение constraints

High water mark

High Water Mark (HWM) – точка в сегменте, после которой блоки данных не отформатированы и никогда не использовались.

Блок данных может быть в одном из состояний:

1. Выше HWM (не отформатированы, никогда не использовались)
2. Ниже HWM:
    - Выделен, но пока не отформатирован и не используется
    - Отформатирован и содержит данные
    - Отформатирован и пуст, т.к. данные были удалены

Low High Water Mark – точка, ниже которой все блоки отформатированы (содержат или содержали данные).

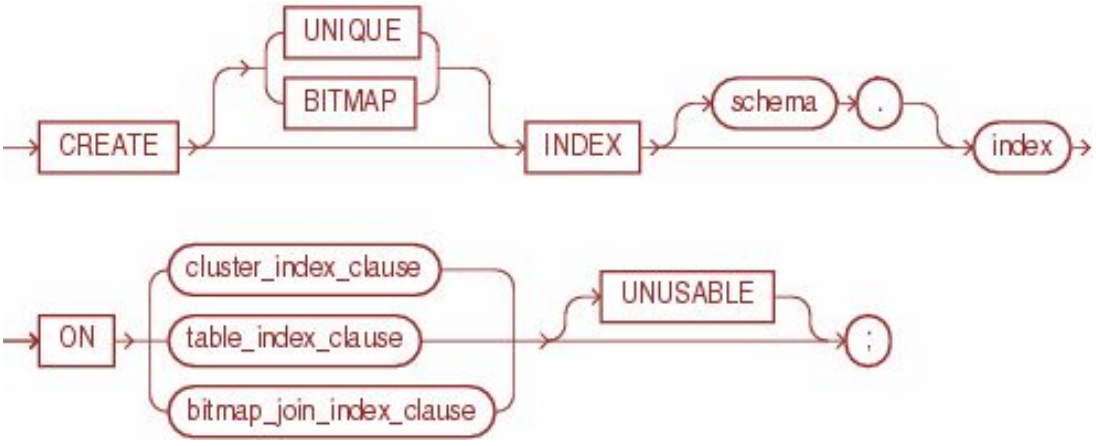
Oracle может выбрать для вставки любой блок ниже HWM, в котором достаточно места (ниже low HWM, или между ними).

Full Table Scan: т.к. блоки ниже HWM форматируются только при использовании, некоторые могут не быть не отформатированы. Поэтому Oracle читает bitmap block и определяет low HWM, читает все блоки до low HWM (они гарантированно отформатированы – можно применить «мультиблочное» чтение – за 1 операцию чтения с диска подтягивается DB_FILE_MULTIBLOCK_READ_COUNT блоков, это задано в в v$parameter), и затем читает по одному отформатированные блоки между low HWM и HWM.

Когда место между low HWM и HWM заполнено, HWM увеличивается, а low HWM становится на её старое место.

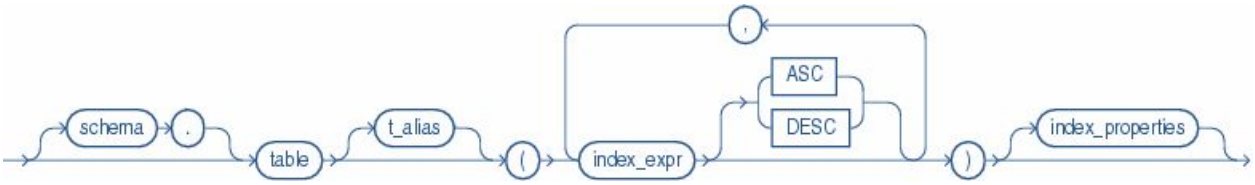
CREATE INDEX

Синтаксис:



Для UNUSABLE индекса не создается сегмент, соотв. он не может быть использован при запросе, пока не перестроен (*rebuild*), или не удален и пересоздан (*drop, create*)

table_index_clause:



index_expr – поле таблицы, или выражение (*function-based indexes - FBI*). Bitmap индекс может иметь до 30 полей, остальные до 32.

ASC/DESC – способ сортировки ключей в индексе (это не то же самое, что Reverse index!)

index_properties – физические атрибуты (*pctfree, pctused, initrans, storage*), *logging, tablespace, parallel,* (IN)VISIBLE – видимость для CBO (Cost-Based Optimizer – оптимизатор запросов), REVERSE и др.

Reverse indexes

Индексы с реверсированным ключом – байты данных ключевого столбца в блоке индекса меняют порядок на противоположный (порядок столбцов остается неизменным). Oracle не сохраняет ключи индекса друг за другом в лексикографическом порядке.

Преимущество: если ключи монотонно возрастают, то велика вероятность, что вставляемые в разных сессиях строки попадут в один и тот же блок в индексе. Reverse индекс позволяет уменьшить конкуренцию за заголовок блока индекса при вставках из множества параллельных сессий, поскольку вероятность попадания «перевернутых» ключей в один и тот же блок меньше, чем неизмененных.
Недостаток: работает только на равенство, диапазонные поиски не работают.

Function-based indexes

Создается, когда в качестве *index_expr* задано выражение по полю таблицы, константа, SQL или user-defined функция (должны быть *deterministic* – для всех входных значений всегда возвращаются одни и те же результаты).

После создания FBI рекомендуется обновить статистику по индексу и таблице, чтобы CBO мог принимать правильные решения о его использовании.

Oracle иногда не может преобразовать типы, даже если это явно задано (to_number('123 abc')). Поэтому, если индекс построен на TO_NUMBER/TO_DATE и вставляется/изменяется невалидное значение, возникает ошибка в операторе INSERT/UPDATE

Одно из применений - индекс по части таблицы: create index scott.emp_job_manager_idx on scott.emp (case when job = 'MANAGER' then 1 else null end) – если при запросах в scott.emp не участвуют строки с должностью, отличной от «MANAGER» - нет смысла держать их в индексе, экономим место. Кроме того, полное сканирование индекса быстрее работает по маленькому.

ALTER INDEX

Что можно делать (менять) этой командой:

1. Сжатие (*shrink space*)
2. Параллелизм
3. Физ. атрибуты *(pctfree, pctused, initrans)*
4. (NO)LOGGING
5. Перестроение (*rebuild*) – (NO)REVERSE, отдельные разделы (*partition*), *tablespace*
6. ENABLE/DISABLE – только для FBI
7. UNUSABLE
8. (IN)VISIBLE
9. RENAME TO

Индекс по внешнему ключу

Нужно создавать индекс по внешнему ключу, если выполнены условия:

- Происходит удаление, или перенос в исторические таблицы
- Выполняются запросы PARENT -> CHILD
- Несколько сессий одновременно удаляют данные (блокировки)

Например, индекс по внешнему ключу на справочник не нужен. А индекс по ключу на связанную бизнес-сущность нужен (например, в связке CLIENT и CLIENTS_CONTACT)


lock.sql

Пример блокировок:

Причины неиспользования индексов

- Индекс в состоянии INVISIBLE или UNUSABLE
- Если процент читаемых записей относительно велик, то дешевле может быть *full scan* (чтение выполняется по

несколько блоков за 1 операцию, как задано параметром инициализации *db_file_multiblock_read_count*)

- Неявное преобразование типов – индекс не используется, если тип данных столбца приводится к типу переменной (искомого значения)
- Неактуальная статистика – оптимальный план выполнения запроса зависит в том числе и от реальных данных, лежащих в таблице. CBO судит об этих данных по статистике


index_not_used.sql

Примеры:

Truncate

Удаление всех строк из таблицы

Не может быть отменено с помощью ROLLBACK и FLASHBACK TABLE

Быстрое (не пишется undo, не выполняются триггеры)

Индексы также транкейтятся

Освобождает место в таблице, кроме указанного в параметре MINEXTENTS

DDL и Commit

Oracle выполняет неявный COMMIT:

- Перед началом выполнения синтактически правильного DDL-выражения, даже если затем его выполнение завершится ошибкой
- После DDL, завершенного без ошибок

А также неявный COMMIT выполняется перед и после большинства процедур из стандартного пакета DBMS_STATS (просмотр и изменение статистики по объектам).

В 3-tier приложениях как правило управление транзакциями осуществляется в слое Java. Отсюда логично следует запрет на DDL в коде PL/SQL процедур (кроме джобов)

Словарь данных

Это *read-only* набор таблиц, содержащих административные метаданные о БД. Например:

- Определения всех объектов БД (включая дефолтные значения полей, ограничения целостности и т.д.)
- Выделенное и используемое объектами пространство
- Справочник пользователей, привилегий и ролей, данные аудита пользователей

Словарь данных содержит объекты двух типов:

- Базовые таблицы – содержат собственно информацию о БД. Только движку Oracle следует работать с ними. Пользователи редко это делают, т.к. большинство данных нормализованы и трудно читаемы.
- Views – преобразуют данные из базовых таблиц в полезный и читаемый вид. Как правило, организованы в наборы (DBA_, ALL_, USER_). Но не всегда (DBA_LOCK – есть, ALL_LOCK – нет).

Prefix	User Access	Contents	Notes
DBA_	Database administrators	All objects	Некоторые DBA_ представления содержат дополнительные столбцы, полезные DBA (для них и предназначены)
ALL_	All users	Objects to which user has privileges	Отражают объекты, на которые у выполняющего запрос пользователя есть права (public- или явные гранты и роли, в дополнение к объектам в своей схеме).
USER_	All users	Objects owned by user	Обычно тут нет поля OWNER, т.к. в базовых таблицах выбираем по условию OWNER = текущий пользователь

Некоторые полезные представления

DBA_SCHEDULER_JOBS	Список scheduler-джобов («новые», рекомендуемые с 11.1)
DBA_JOBS	Список обычных джобов
DBA_JOBS_RUNNING	Джобы из DBA_JOBS, которые сейчас работают
DBA_SOURCE	Код всех пакетов, процедур, триггеров, Java-классов и т.д.
DBA_TABLES	Все таблицы и их характеристики

DBA_TAB_PRIVS	Привилегии(какая,кто выдал, кому, grantable)
DBA_USERS	Пользователи(схемы) БД