Oracle Core

Тема 4

# DDL. Особенности таблиц и индексов. Сбор статистики. Словари данных

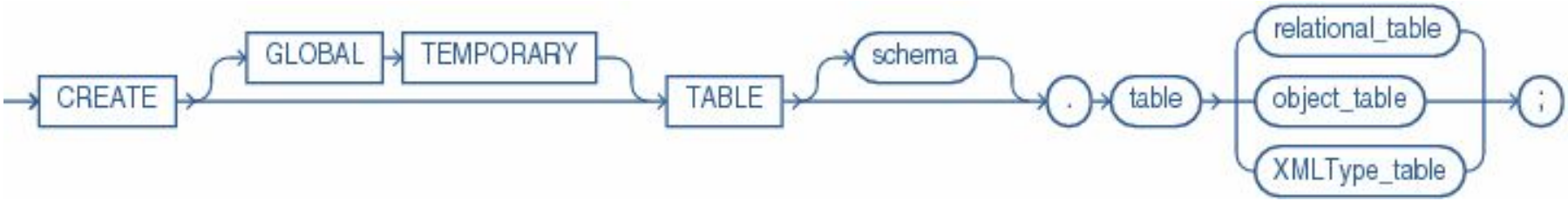
**Юрченко Игорь**

**Senior developer группы разработки Oracle**

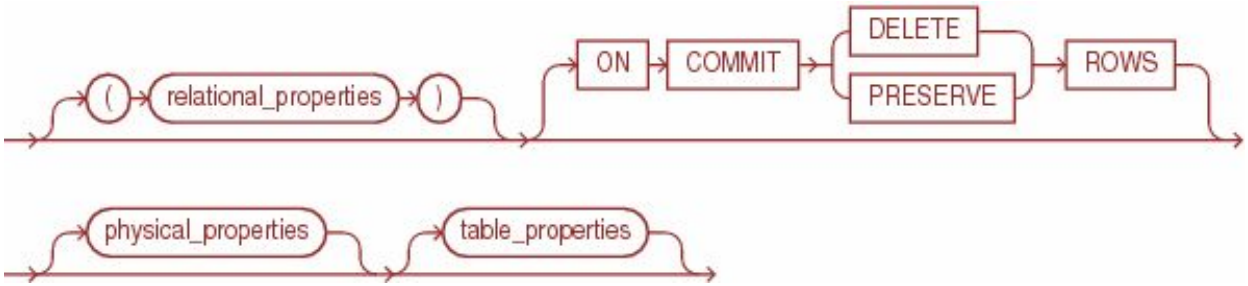
# Содержание

- DDL таблиц
- Параметры создания таблиц: pctfree и pctused; initial, next и pctincrease; minextents и maxextents; logging и nologging; initrans и maxtrans
- High water mark
- DDL индексов
- Reverse indexes, индекс на часть таблицы (по функции)
- Индексы по внешним ключам (зачем нужны, что будет если не создать)
- Причины неиспользования индексов (процент читаемых записей, неявное приведение типов, неактуальная статистика)
- Truncate table
- Неявный commit при DDL. Запрет на DDL в PL/SQL
- Словари данных (группы представлений user_, all_, dba_)
- Самые используемые словари данных (dict, dba_tables, dba_objects и т.д.)

# CREATE TABLE




**relational_table:**



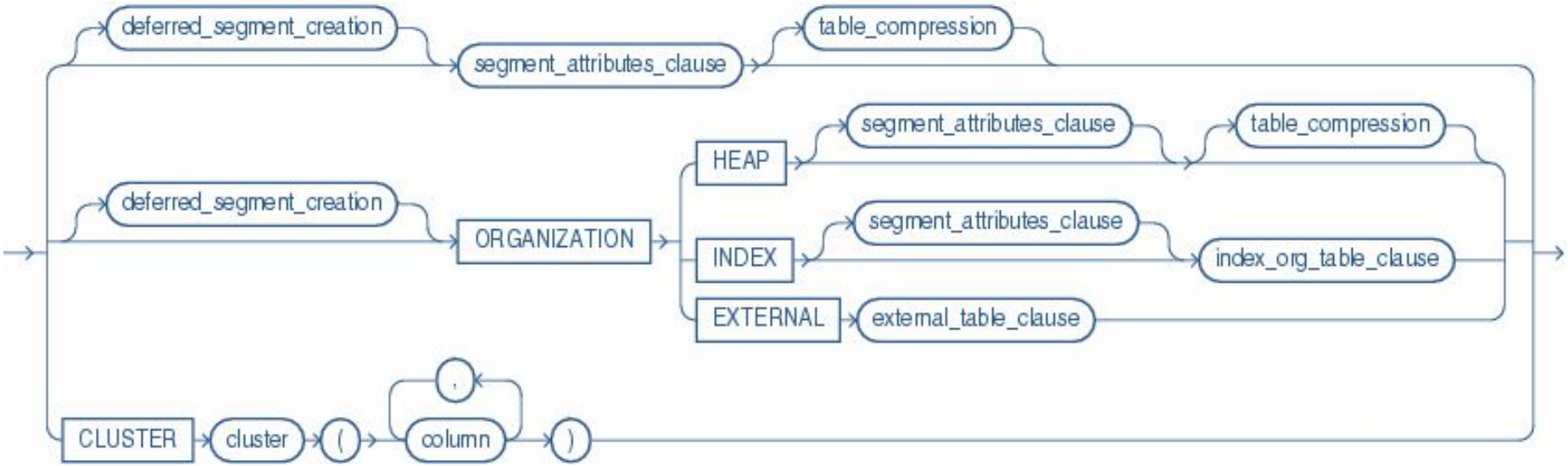

*ON COMMIT DELETE/PRESERVE ROWS – для termporary-таблиц*

*relational_properties* – описание столбцов:

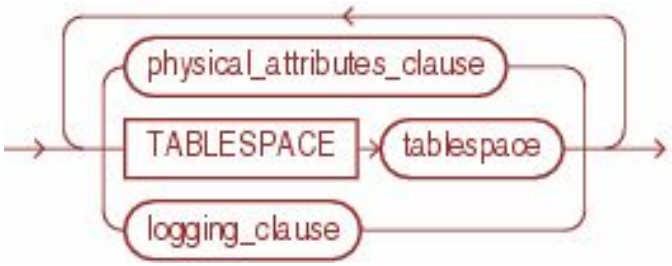
1. Имя
2. Тип данных
3. Default
4. (Not) Null
5. Constraints

```
create table scott.emp (
    id number not null,
    ...
    hiredate date default sysdate not null,
    ...
    deptno number,
    constraint pk_emp primary key (id) using index tablespace users)
```

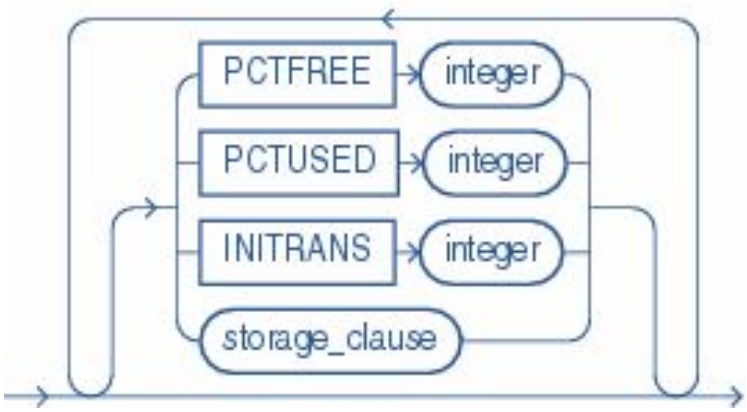
# physical_properties




**segment_attributes_clause:**




**physical_attributes_clause:**




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

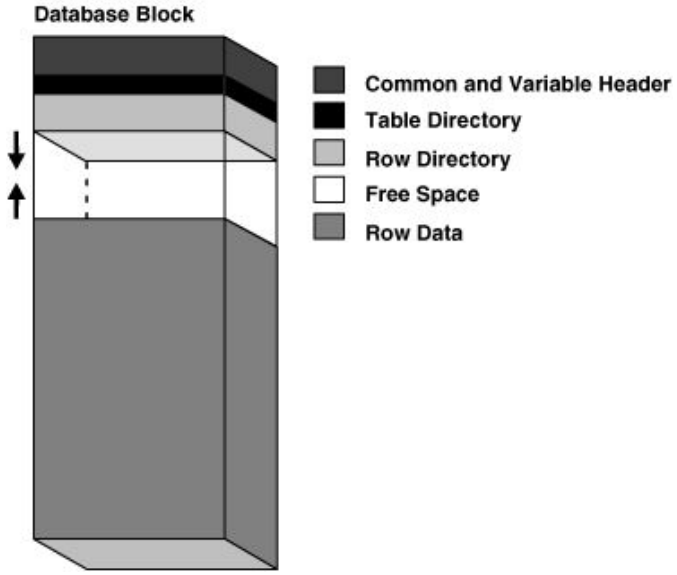
# pctfree, pctused

- 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.

<u>Как работают вместе:</u>

- в новом блоке место, доступное для INSERT, равно [размер блока – (PCTFREE + block header)]. UPDATE может использовать все доступное место, поэтому свободное может стать < PCTFREE
- когда блок заполнен до предела, заданного PCTFREE, Oracle не вставляет новые строки в него (делает только UPDATE), пока занятое место не станет меньше PCTUSED. Когда стало меньше – можно опять INSERT



# initrans, maxtrans

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.*

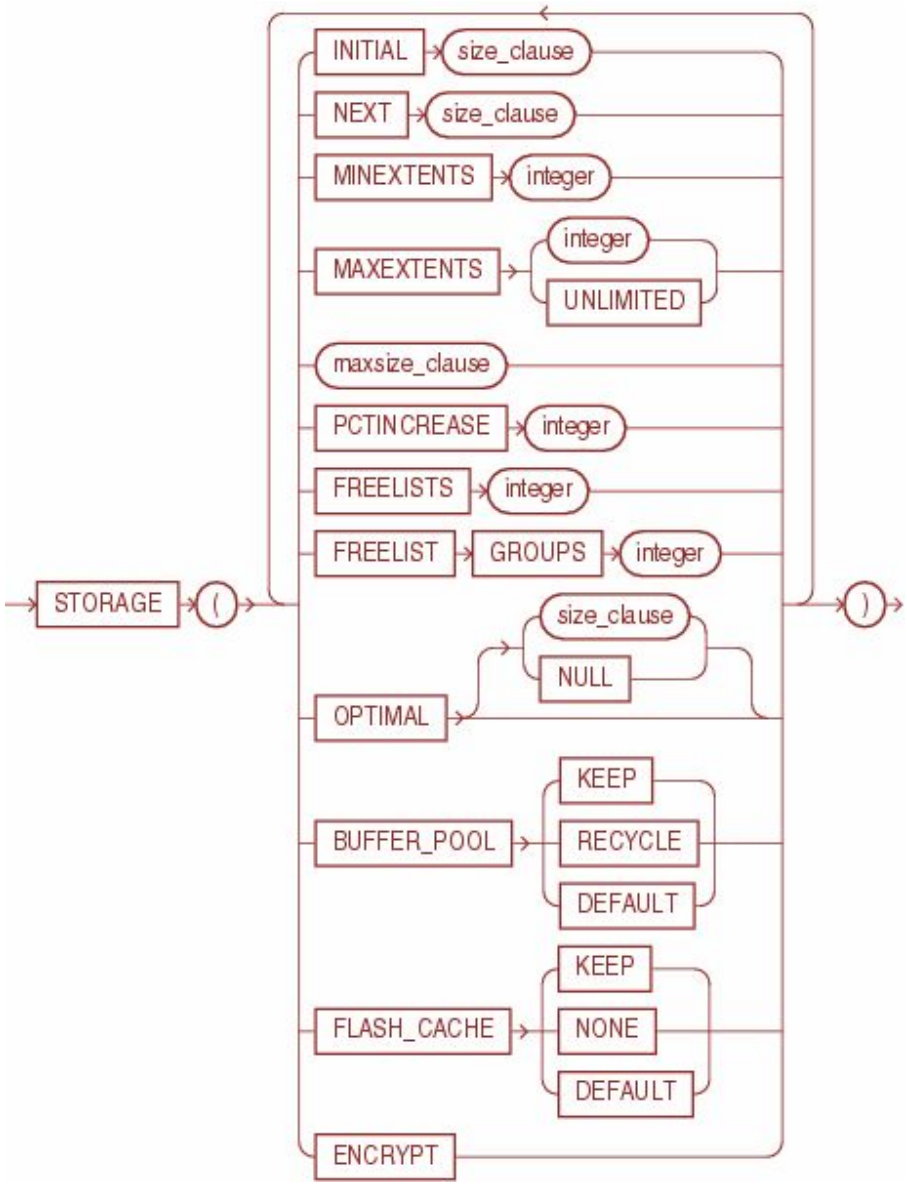
Каждая транзакция, которая выполняет UPDATE блока, занимает в блоке 1 слот. Этот параметр гарантирует, что минимальное количество транзакций могут обновлять блок.

MAXTRANS – максимальное количество слотов транзакций. В 11.2 deprecated. Для существующих остается как есть, для новых 255 (если пытаемся менять, значение игнорируется и подставляется 255).

```
create table scott.bonus_log (...) tablespace users pctfree 2;

create table scott.process_log (...) tablespace users
  pctfree 2 initrans 1 maxtrans 255 storage (
    initial 128k next 128k minextents 1 maxextents unlimited
pctincrease 0);
```

# storage_clause




*Дальше описаны параметры для типа tablespace'ов – locally managed (extent_management = LOCAL в dba_tablespaces). Другой тип tablespace'ов – dictionary managed tablespaces (устаревший, характеристики tablespace хранятся в словаре, запланирована отмена поддержки в будущих релизах).*

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

MAXSIZE – максимальный размер объекта (UNLIMITED – без ограничений)

MAXEXTENTS – используется только для dictionary-managed tablespaces

# initial, next, pctincrease, minextents

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

- 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.

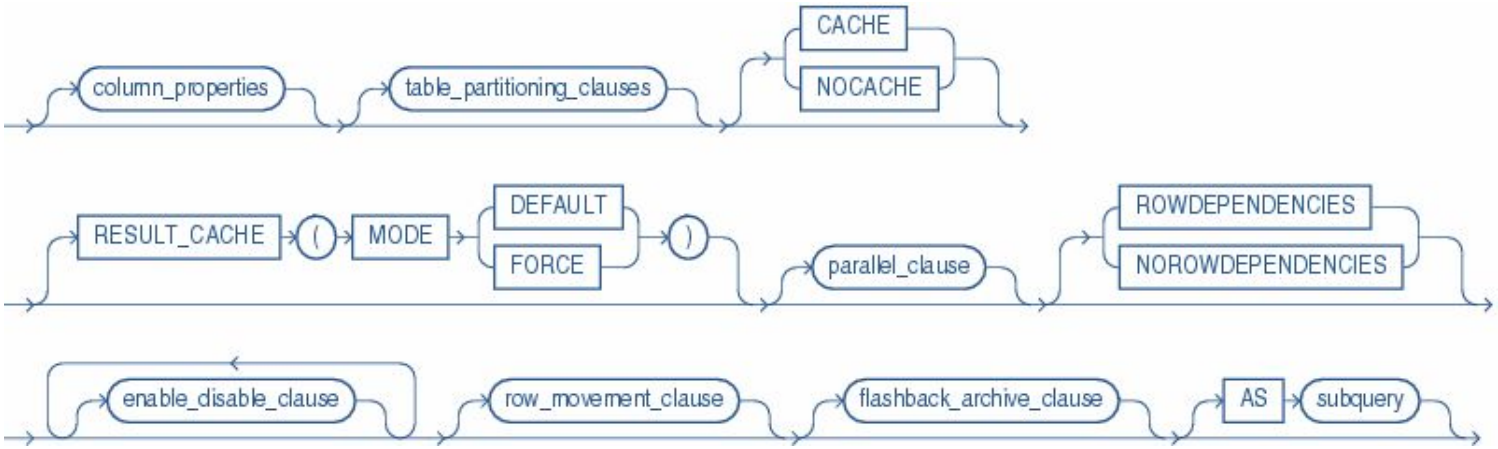
NEXT – размер следующего выделяемого экстента:

- 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).

# table_properties

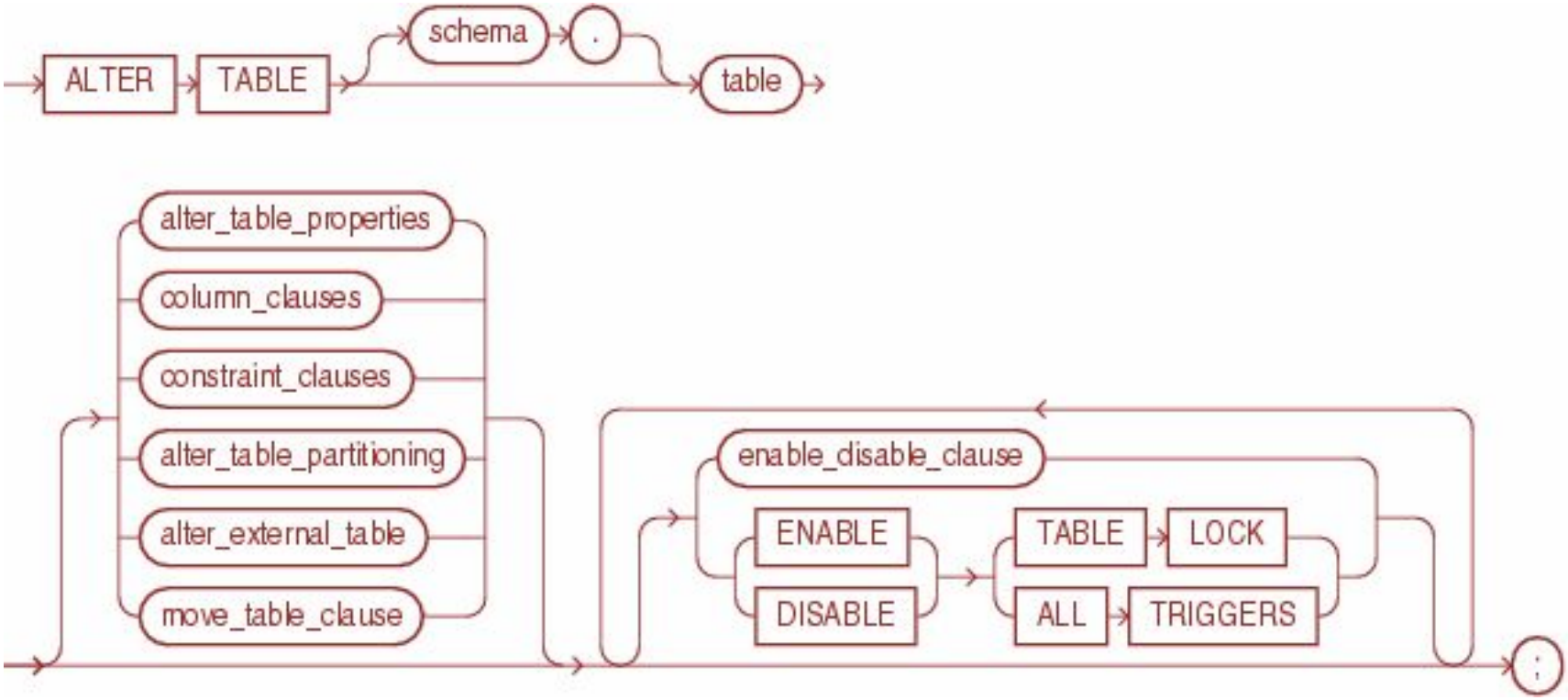



*AS subquery* – запрос, результат которого будет вставлен в таблицу сразу при создании. При таком подходе можно не указывать явно перечень полей, он сформируется из запроса (а также типы данных и их размерности).

- если выбираем поле (а не выражение) – с него переносим в новую таблицу NOT NULL constraint, если они явно заданы пользователем и имеют состояния NOT DEFERRABLE и VALIDATE
- NOT NULL constraint, созданные автоматически Oracle (например primary key), не переносятся. Также не переносятся атрибуты: primary/unique/foreign keys, check constraints, partitioning criteria (задается для новой таблицы отдельно), индексы и column default values

```
create table dual2 as select dummy || 'Y' as dummy2 from dual;
```

# 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)

# ALTER TABLE

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

```
alter table dual2 add (dummy2 number);
alter table dual2 modify (dummy2 number(3));
alter table dual2 rename column dummy2 to ololo;
alter table dual2 drop column ololo;
```

*constraint_clauses* – добавление, изменение, переименование и удаление constraints:

```
alter table dual2 add constraint pk_dual2 primary key (dummy);
alter table dual2 add constraint fk_dual2_dummy2 foreign key (dummy2)
references clients (id);
alter table dual2 rename constraint fk_dual2_dummy2 to fk_dual2_qwe;
alter table dual2 drop constraint fk_dual2_qwe;
```

*alter_table_partitioning* – изменение способа партиционирования таблицы

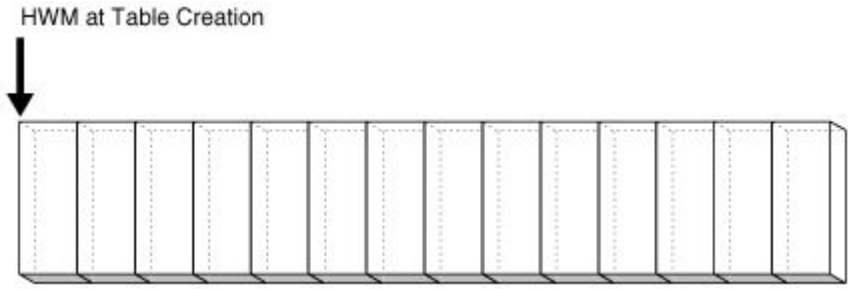
*move_table_clause* – изменение атрибутов сегмента таблицы, в том числе перемещение в другой tablespace

*enable_disable_clause* – включение/выключение constraints:

```
alter table scott.emp disable constraint fk_deptno;
alter table scott.emp enable novalidate constraint fk_deptno;
```

# High water mark

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




*ASSM (Automatic Segment Storage Management) – наиболее эффективный и дефолтный способ управления пространством в сегментах.*

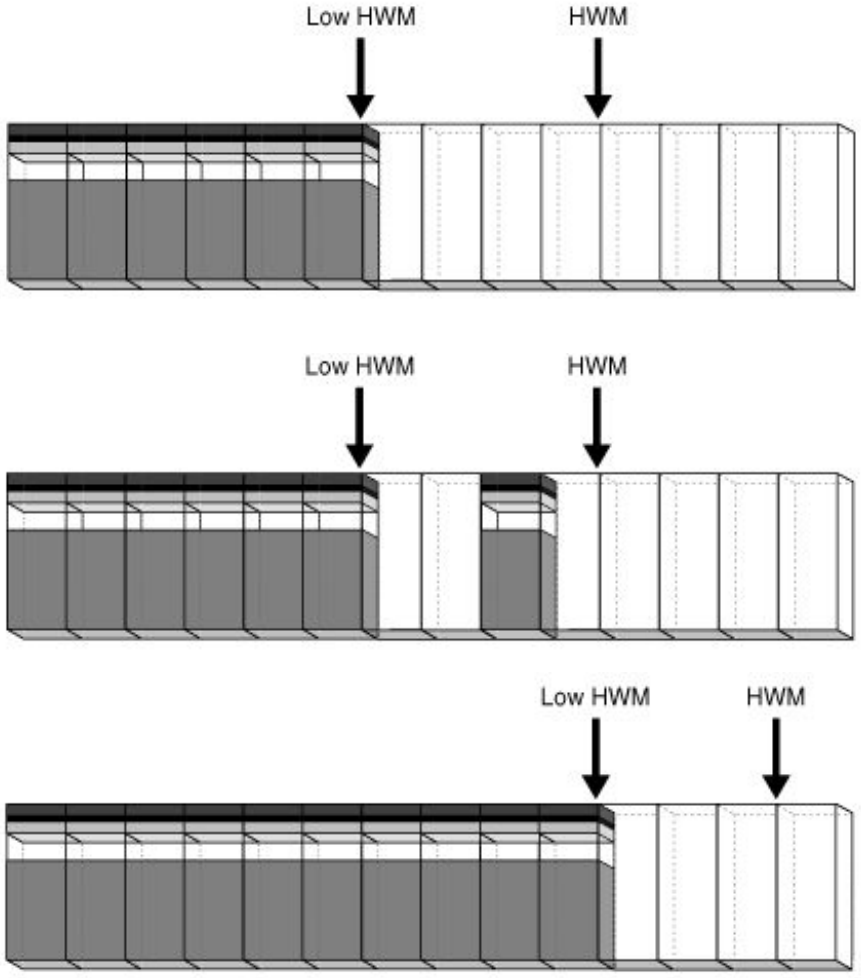
Блок может быть в одном из состояний:

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

ASSM разделяет операции вставки разных сессий между блоками, чтобы избежать конкуренции.

# High water mark

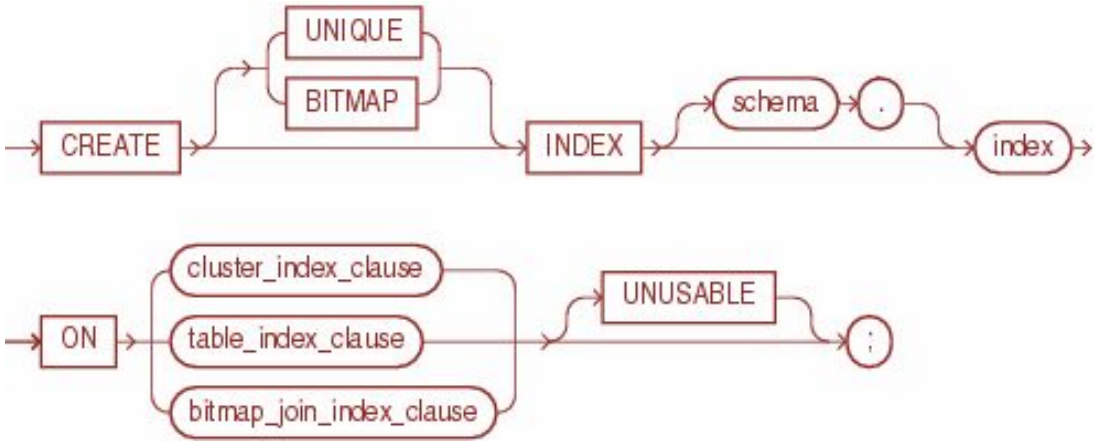
При вставке данных Oracle должен выделить группу блоков. Они располагаются ниже HWM. Oracle форматирует в этой группе bitmap block, содержащий метаданные, но не форматирует другие блоки.




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).

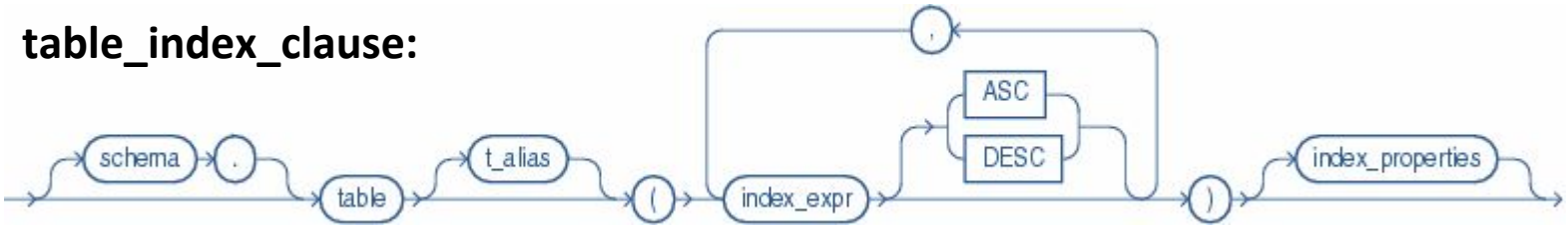
Full Table Scan: т.к. блоки ниже HWM форматируются только при использовании, некоторые могут не быть не отформатированы. Поэтому Oracle читает bitmap block и определяет low HWM, читает все блоки до low HWM (они гарантированно отформатированы), и затем читает по одному отформатированные блоки между low HWM и HWM.
Когда место между low HWM и HWM заполнено, HWM увеличивается, а low HWM становится на её старое место.

# CREATE INDEX




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 (не путать с reverse index)*

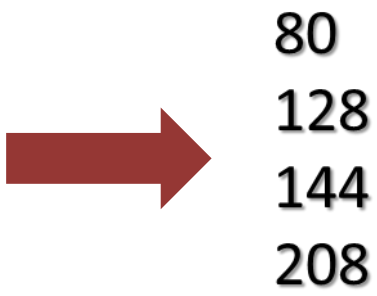
*index_properties* – физические атрибуты (*pctfree, pctused, initrans, storage*), *logging, tablespace, parallel,* (IN)VISIBLE – видимость для CBO, REVERSE и др.

# 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.

http://habrahabr.ru/post/102785/

| Поле в таблице (bin) | Ключ reverse-индекса (bin) |
|---|---|
| 1 (00000001) | 128 (10000000) |
| … | … |
| 9 (00001001) | 144 (10010000) |
| 10 (00001010) | 80 (01010000) |
| 11 (00001011) | 208 (11010000) |




Если ключи монотонно возрастают, то велика вероятность, что вставляемые строки попадут в один и тот же блок. Reverse индекс позволяет уменьшить конкуренцию за заголовок блока индекса при вставках из множества параллельных сессий, поскольку вероятность попадания ключей 144, 80 и 208 в один и тот же блок меньше, чем ключей 9, 10 и 11.

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.*

Особенности:

- Если использован *public synonym*, а затем в схеме создается объект с таким же именем, Oracle помечает индекс DISABLE. Затем, если делаем ENABLE или REBUILD, имя продолжает указывать на начальный объект (а не на созданный)
- 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
- Если создаем FBI используя некий формат даты-времени, в системных представлениях этот формат потом может оказаться другой:

```
create index scott.emp_hiredate_idx on scott.emp(nvl(hiredate,
to_date('31/12/9999','dd/mm/yyyy')));

select * from dba_ind_expressions where index_name = 'EMP_HIREDATE_IDX';
NVL("HIREDATE",TO_DATE(' 9999-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
```

# ALTER INDEX

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

```
-- пример ветного синтаксиса
create index scott.emp_ename_idx on scott.emp (ename) tablespace large_idx
pctfree 5 initrans 2 storage (initial 64k next 1m minextents 1);

-- ускорение создания индекса
create unique index scott.emp_empno_idx on scott.emp(empno) parallel 32;
alter index scott.emp_empno_idx noparallel;

-- функциональный индекс для поиска по имении в нижнем регистре
create index scott.emp_ename_lower_idx on scott.emp (lower(ename));
```

**Индекс по части таблицы:**

```
create index scott.emp_job_manager_idx on scott.emp (case when job =
'MANAGER' then 1 else null end);
```

# Индекс по внешнему ключу

- Удаление (это происходит в constraints с ON DELETE CASCADE, или может быть явно прописано в коде):

```
DELETE scott.emp
WHERE  deptno = 10;
```

```
DELETE scott.dept
WHERE  deptno = 10;
```

- Запросы PARENT -> CHILD

```
SELECT d.deptno,
       e.ename
FROM   scott.dept d
JOIN   scott.emp e ON e.deptno = d.deptno
WHERE  d.deptno = 10
```

- Блокировки

*См. далее*

Не нужен, если выполнены все условия:

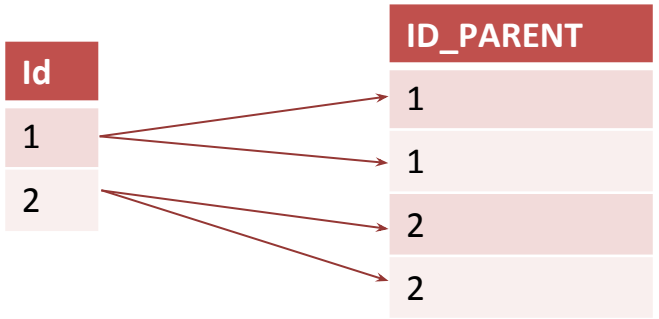
1. Не удаляются строки из PARENT-таблицы
2. Не обновляется ключ в PARENT-таблице
3. Нет соединений PARENT -> CHILD

*Например, индекс по внешнему ключу на справочник не нужен.*

# Индекс по внешнему ключу (locks)

Пример: в двух сессиях удаляем данные
по двум ID – логично ожидать, что они удалятся (разные строки)

http://docs.oracle.com/cd/E11882_01/server.112/e16508/consist.htm#CNCPT1340




| Сессия 1 | Сессия 2 | Коммент |
|---|---|---|
| drop index child_id_parent_idx | | |
| delete child where id_parent = 1 | delete child where id_parent = 2 | Ок |
| delete parent where id = 1 | delete parent where id = 2 | Сессия 2 запрашивает блокировку SSX на CHILD, но сессия 1 держит на ней же блокировку SX – сессия 2 ждет завершения транзакции 1 |
| create index child_id_parent_idx on child (id_parent) | | |
| delete child where id_parent = 1 | delete child where id_parent = 2 | Ок |
| delete parent where id = 1 | delete parent where id = 2 | Ок (при наличии индекса блокировка SSX не запрашивается) |

# Причины неиспользования индексов

- INVISIBLE или UNUSABLE

- процент читаемых записей – если 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*, например 8)

*Пример: таблица содержит 80 строк и занимает 8 блоков, индекс по ней занимает 1 блок. Нужно получить строки с 1 по 40 (50% содержимого).*
*Через FS это можно сделать за 1 операцию чтения – просто прочитать блоки таблицы.*
*Через Index Range Scan нужно прочитать 1 блок индекса и затем по ROWID оттуда 4 раза обратиться к блокам таблицы, итого 5 операций чтения.*

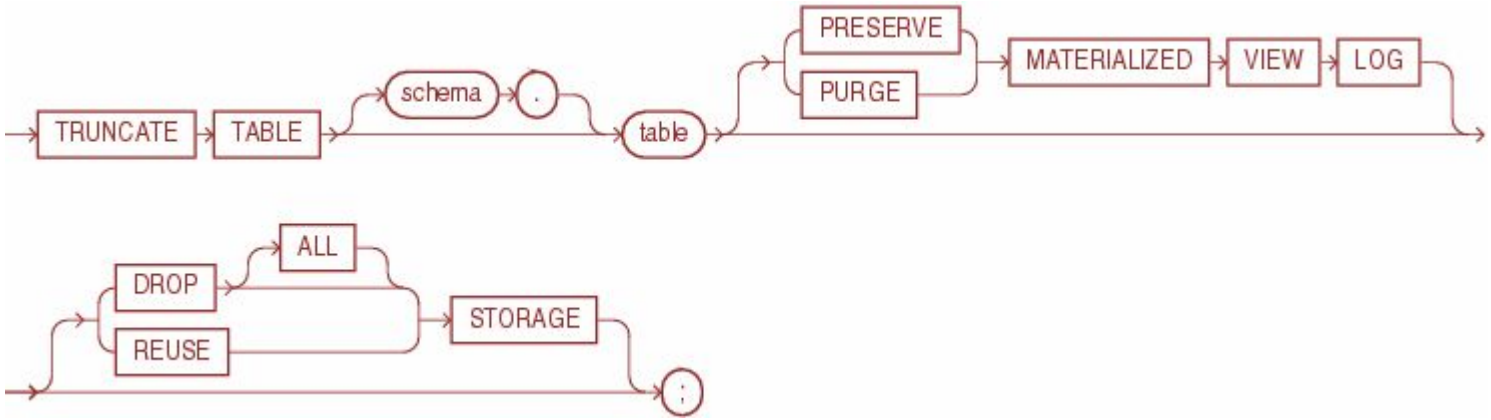
- неявное преобразование типов – индекс не используется, если тип данных столбца приводится к типу переменной (искомого значения)

*Пример: индекс построен по текстовому полю, а в запросе используем число. При неявном преобразовании текст преобразуется к числу, то есть в индексе пытаемся найти число. Но в индексе только текст – не используется, идем в таблицу и делаем full scan.*

- неактуальная статистика – оптимальный план выполнения запроса зависит в том числе и от реальных данных, лежащих в таблице. CBO судит об этих данных по статистике

*Пример: было в нашей таблице 80 строк, для запроса «where id between 1 and 70» выбирался full scan. Вставили еще 800 000 строк, статистику не обновили. Тот же запрос начинает работать медленно, т.к. по-прежнему использует full scan – CBO не знает, что мы вставили данные в таблицу.*

# Оператор 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

PRESERVE/PURGE MV LOG – сохранить (default) или удалить *materialized view log*

DROP STORAGE – освободить место (установить HWM), кроме MINEXTENTS

DROP ALL STORAGE – освободить все место, включая MINEXTENTS

REUSE STORAGE – не трогать HWM; место остается в таблице

# 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).

```
-- есть некая строка
select * from t1 where id = (select max(id) from t1);
-- удаляем её
delete t1 where id = (select max(id) from t1);
-- выполняем любой DDL, например добавляем комментарий на поле таблицы
comment on column t1.id is 'id';
-- видно в другой сессии - та строка удалена, выбирается другая
select * from t1 where id = (select max(id) from t1);
```

В 3-tier приложениях как правило управление транзакциями осуществляется в слое бизнес логики (Java). Поэтому в коде PL/SQL не используется DDL.

# Словарь данных

Data Dictionary – *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

Oracle обращается к словарю для поиска информации о пользователях, объектах БД и структурах данных, а также изменяет словарь данных каждый раз при выполнении команды DDL.

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.

*Примеры: sys.tab$, sys.seg$, sys.x$ksppcv, sys.x$ksppi, sys.deferred_stg$*

- 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, а ALL_LOCK – нет).

*Пример: dba_tables*

# Наборы представлений словаря данных

| 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 |

Список представлений словаря: `select * from dictionary`

**DUAL** – таблица в словаре. Все пользователи имеют к ней доступ. Полезна, когда нужно вернуть значение один раз. Содержит 1 столбец DUMMY типа VARCHAR2(1) и одну строку со значением “X”.

Словарь хранится в табличном пространстве SYSTEM и всегда доступен.

# Некоторые таблицы словаря данных

| | |
|---|---|
| DBA_OBJECTS | Все объекты БД (таблицы, пакеты, триггеры и т.д.) |
| DBA_ROLE_PRIVS | Роли, выданные схемам и другим ролям |
| 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_SEGMENTS | Сегменты (таблиц, разделов, индексов, отката и т.д.) |
| DBA_SEQUENCES | Список секвенсов в БД |
| DBA_SOURCE | Код всех пакетов, процедур, триггеров, Java-классов и т. д. |
| DBA_SYNONYMS | Список синонимов в БД |
| DBA_TABLES | Все таблицы и их характеристики |
| DBA_TABLESPACES | Табличные пространства в БД |
| DBA_TAB_COLUMNS | Поля (и характеристики) всех таблицы и представлений |
| DBA_TAB_COMMENTS | Комментарии на таблицы и представления |
| DBA_TAB_PRIVS | Привилегии (какая, кто выдал, кому, grantable) |
| DBA_TRIGGERS | Триггеры (характеристики, код) |
| DBA_USERS | Пользователи (схемы) БД |
| DBA_VIEWS | Представления (характеристики, текст) |
| DBA_HIST_ACTIVE_SESS | Исторические данные по статистике активных сессий |

# Summarizing

- DDL таблиц
- Параметры создания таблиц: pctfree и pctused; initial, next и pctincrease; minextents и maxextents; logging и nologging; initrans и maxtrans
- High water mark
- DDL индексов
- Reverse indexes, индекс на часть таблицы (по функции)
- Индексы по внешним ключам (зачем нужны, что будет если не создать)
- Причины неиспользования индексов (процент читаемых записей, неявное приведение типов, неактуальная статистика)
- Truncate table
- Неявный commit при DDL. Запрет на DDL в PL/SQL
- Словари данных (группы представлений user_, all_, dba_)
- Самые используемые словари данных (dict, dba_tables, dba_objects и т.д.)

# Список использованных материалов

Документация Oracle 11.2
http://docs.oracle.com/cd/E11882_01/index.htm

DDL таблиц
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_7002.htm#i2095331
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_3001.htm#CJAHHIBI

Полезная информация от Тома Кайта про pctfree, pctused
https://asktom.oracle.com/pls/apex/f?p=100:11:0::::P11_QUESTION_ID:2556828000346627752

Overview of Data Blocks
http://docs.oracle.com/cd/E11882_01/server.112/e40540/logical.htm#CNCPT302

High Water Mark
http://docs.oracle.com/cd/E11882_01/server.112/e40540/logical.htm#CNCPT89022

DDL индексов
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_5012.htm#i2062403
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_1010.htm#i2050158

# Список использованных материалов


Reverse indexes
http://www.sql.ru/forum/32684/primery-neobhodimosti-ispolzovaniya-reverse-index

Индекс по внешнему ключу
https://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:292016138754

Неявное преобразование типов
http://docs.oracle.com/cd/E11882_01/server.112/e41084/sql_elements002.htm#SQLRF00214

Truncate
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_10007.htm#i2067571

DDL и COMMIT
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_4010.htm#i2060233

DBMS_STATS
http://docs.oracle.com/cd/E11882_01/appdev.112/e40758/d_stats.htm#ARPLS059

# Список использованных материалов

Словарь данных
http://docs.oracle.com/cd/E11882_01/server.112/e40540/datadict.htm#CNCPT002