Oracle Database 12c: podstawy administracji

Maciej Zakrzewicz
ExplainIT.pl

ROZDZIAŁ 7
ZARZĄDZANIE TABELAMI

PLAN ROZDZIAŁU

zarządzanie tabelami bazy danych

- Rodzaje tabel
- Segmenty, ekstenty, bloki
- ROWID
- Wskaźnik wysokiej wody
- Tworzenie podstawowych typów tabel:
  - tabele zwykłe
  - tabele partycjonowane
  - tabele kompresowane
  - tabele tymczasowe
  - tabele szyfrowane
- Ograniczenia integralnościowe
- Operacje administracyjne

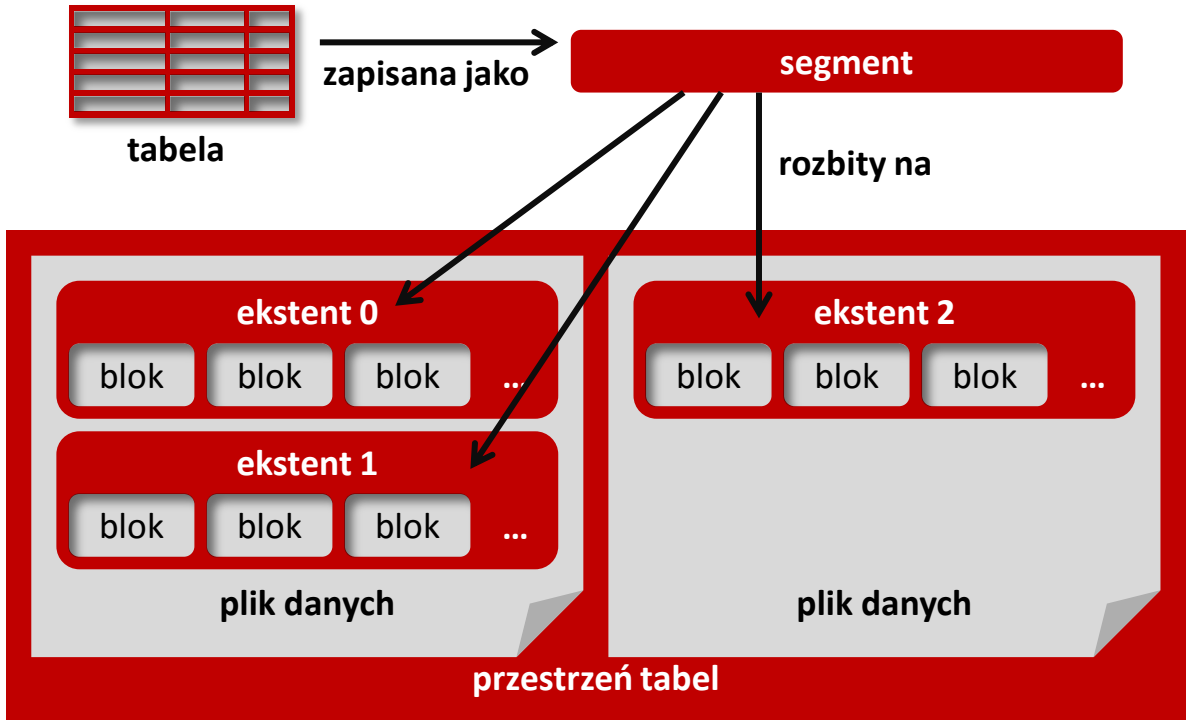
Tabele

rodzaje tabel bazy danych

- Podział logiczny
  - relacyjne
  - obiektowe
- Podział fizyczny
  - sterta (heap)
  - tabela-indeks (index-organized table)
  - klaster indeksowy
  - klaster haszowy
  - partycjonowana
  - kompresowana
  - zewnętrzna
- Podział wg trwałości danych
  - trwałe
  - tymczasowe
- Tabele szyfrowane

JAK TABELE SĄ ZAPISANE W PLIKACH DANYCH

struktury fizycznej reprezentacji tabeli w plikach danych



JAKIE EKSTENTY MA MOJA TABELA

perspektywa dba_extents

```
SQL> select segment_name, extents
     from dba_segments where segment_name='DEMO' and owner='SCOTT';

SEGMENT_NAME  EXTENTS
------------- -------
DEMO                6

SQL> select extent_id, file_id, block_id, bytes
     from dba_extents where segment_name='DEMO' and owner='SCOTT'

 EXTENT_ID    FILE_ID   BLOCK_ID      BYTES
---------- ---------- ---------- ----------
         0          6        216      65536
         1          6        224      65536
         2          6        232      65536
         3          6        240      65536
         4          6        248      65536
         5          6        256      65536
```

JAK WYMIAROWANE SĄ EKSTENTY

zasady przydziału ekstentów dla tabeli

- Tabela automatycznie otrzymuje od serwera nowy ekstent gdy brakuje miejsca w dotychczasowych ekstentach
- Rozmiar przydzielanego ekstentu zależy od ustawień przestrzeni tabel:
  - EXTENT MANAGEMENT LOCAL UNIFORM SIZE x
    - każdy ekstent ma rozmiar X
    - efektywne dla dużych tabel (można uzyskać niską fragmentację)
  - EXTENT MANAGEMENT LOCAL AUTOALLOCATE
    - 16 pierwszych ekstentów tabeli ma rozmiar 8 bloków (np. 64KB)
    - kolejne ekstenty mają rozmiar 128 bloków (np. 1MB)
    - w rezultacie szybko rosnące tabele otrzymują coraz większe ekstenty

JAK ZBUDOWANY JEST BLOK DANYCH

wewnętrzna budowa bloku danych

- Rozmiar bloku jest jednakowy w całej przestrzeni tabel (zwykle też w całej bazie danych)
  - domyślnie 8KB
- Każdy blok należy do dokładnie jednego ekstentu
- Wstawiane rekordy stopniowo wypełniają wolne miejsce w bloku
- Po zapełnieniu bloku, kolejne rekordy są zapisywane do następnego bloku

JAK BLOK JEST WYPEŁNIANY REKORDAMI

rezerwowanie wolnej przestrzeni dla przyszłych modyfikacji

- Nowe rekordy są umieszczane w bloku tak długo, aż ilość wolnego miejsca spadnie poniżej parametru PCTFREE
- Pozostawiane wolne miejsce posłuży w przyszłości do rozszerzania rekordów w wyniku ich modyfikacji
- Domyślne PCTFREE = 10%
- Zalecane PCTFREE = 0 dla tabel, które:
  - nigdy nie podlegają modyfikacji
  - posiadają rekordy stałej długości
- Brak wolnego miejsca na modyfikację spowoduje migrację rekordu do innego bloku
  - w pierwotnym bloku pozostanie wskaźnik przekierowania – gorsza wydajność

JAK ODNALEŹĆ ZMIGROWANE REKORDY

wyszukanie i naprawienie zmigrowanych rekordów

```
SQL> analyze table demo compute statistics;

SQL> select num_rows, chain_cnt
     from dba_tables where table_name='DEMO' and owner='SCOTT';

  NUM_ROWS  CHAIN_CNT
---------- ----------
      8192     40

SQL> @?/rdbms/admin/utlchain.sql

SQL> analyze table demo list chained rows;

SQL> select head_rowid from chained_rows;

HEAD_ROWID
------------------
AAAWxFAAGAAAAJAACH
AAAWxFAAGAAAAJAACI
...
```

- Naprawienie zmigrowanych rekordów polega na ich usunięciu i ponownym umieszczeniu w tabeli
- W przypadku małych tabel:
  - alter table DEMO move;
  - expdp/impdp
- Jednocześnie warto skorygować PCTFREE na przyszłość

Co to jest ROWID

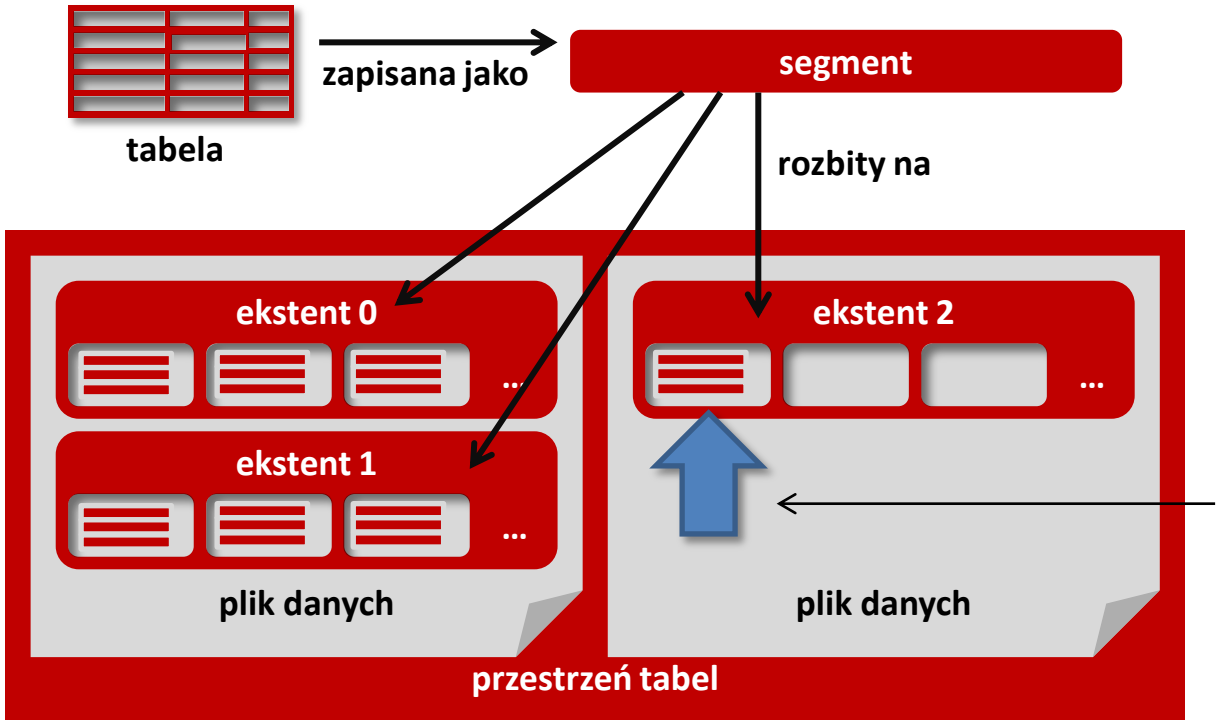
struktura identyfikatora rekordu

- Każdy rekord posiada niepowtarzalny adres fizyczny – pseudokolumnę ROWID
- ROWID ma rozmiar 10 bajtów:
  - bity 1 do 32 (bajty 1 do 4): identyfikator obiektu (0-4294967295)
  - bity 33 do 44 (bajt 5 i pół bajtu 6): numer pliku (0-4095)
  - bity 45 do 64 (pół bajtu 6 i bajty 7 i 8): numer bloku w pliku (0-1048575)
  - bity 65 do 80 (bajty 9 i 10): numer rekordu w bloku (0-65535)
- Podczas wyświetlania (np. `select rowid from emp`) stosowane jest kodowanie radix 64 (A-Za-z0-9+/): OOOOOOFFFBBBBBBRRR
- W przestrzeniach tabel BIGFILE, numer pliku i numer bloku są łączone w numer bloku
- Pakiet DBMS_ROWID umożliwia dekodowanie ROWID:

```
SQL> select DBMS_ROWID.ROWID_OBJECT(rowid) "OBJECT",
   DBMS_ROWID.ROWID_RELATIVE_FNO(rowid) "FILE",
   DBMS_ROWID.ROWID_BLOCK_NUMBER(rowid) "BLOCK",
   DBMS_ROWID.ROWID_ROW_NUMBER(rowid) "ROW"
   from demo
```

Wskaźnik wysokiej wody

High Water Mark (HWM)



Wskaźnik wysokiej wody (HWM) wskazuje ostatni blok tabeli kiedykolwiek użyty do zapisu rekordów. Przesuwa się w przód automatycznie. Nie cofa się przy DELETE! Używany do zatrzymania pełnego odczytu tabeli.

JAK PRZESUWA SIĘ HWM

podczas wstawiania i usuwania rekordów - demonstracja

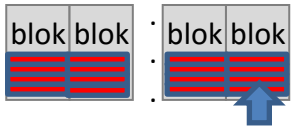
```
CREATE TABLE test (
  id NUMBER(10),
  desc VARCHAR2(100))
PCTFREE 10;
```

```
ANALYZE TABLE test COMPUTE STATISTICS;
SELECT BLOCKS, EMPTY_BLOCKS FROM
USER_TABLES WHERE TABLE_NAME='TEST';
```

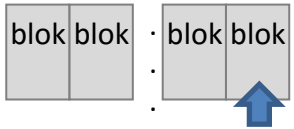
HWM

```
DECLARE
  i NUMBER(10);
BEGIN
 FOR i IN 1..10000 LOOP
  INSERT INTO test VALUES (i,'X');
 END LOOP;
END;
```

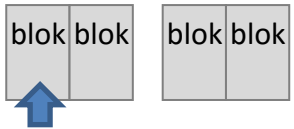
→ HWM = 20



DELETE test; → HWM = 20



TRUNCATE TABLE test; → HWM = 0



Jak przesuwa się HWM

ALTER TABLE SHRINK SPACE

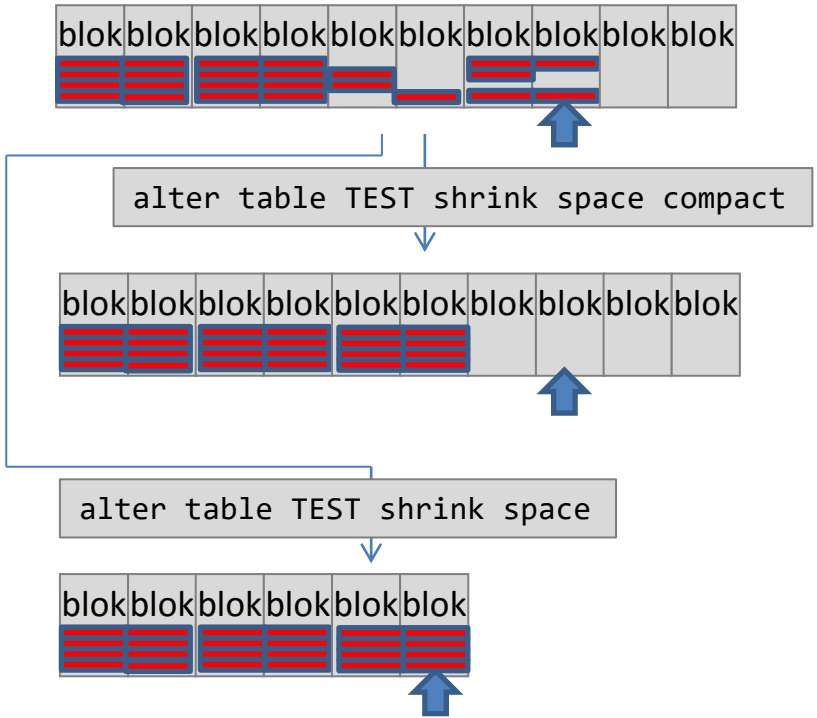


Tabela musi mieć włączone pozwolenie na ROW MOVEMENT:

```
alter table TEST enable
row movement;
```

Jak utworzyć tabelę zwykłą (sterta)

polecenie SQL i własności

```
create table DEMO (
   ID number(8),
   NAZWA varchar2(100),
   CENA number(8,2))
tablespace USERS
[pctfree 20]
[nologging];
```

- Domyślna struktura tabeli
- Fizyczna kolejność rekordów wynika z kolejności ich wstawiania
- Klauzula `NOLOGGING` umożliwia pominięcie zapisów operacji `INSERT` do dziennika powtórzeń gdy dla tabeli będą wykonywane:
  - `INSERT` ścieżką bezpośrednią
  - SQL*Loader ścieżką bezpośrednią

JAK NAZYWAĆ TABELĘ

zasady i ograniczenia dotyczące nazw obiektów

- Od 1 do 30 znaków
  - wyjątek: łączniki bazy danych (db link) – do 128 znaków
- Nie wolno stosować słów kluczowych SQL (np. SELECT)
- Pierwszy znak musi być literą
- Dozwolone znaki: litery, cyfry, _, $, #
- Małe litery są zamieniane na wielkie
- Powyższe ograniczenia (oprócz limitu długości) można ominąć otaczając nazwę obiektu znakami cudzysłowia
  - w przyszłości każde odwołanie do takiego obiektu też musi otaczać jego nazwę znakami cudzysłowia!

JAK UTWORZYĆ TABELĘ PARTYCJONOWANĄ

polecenie SQL i własności

```
create table DEMO (
   ID number(8),
   NAZWA varchar2(100),
   CENA number(8,2))
pctfree 20
partition by range (CENA)
(partition CHEAP values less than (1000)
           tablespace DATA01,
 partition MEDIUM values less than (10000)
           tablespace DATA02,
 partition HIGH values less than (maxvalue)
           tablespace DATA03)
```

- Partycjonowanie zakresowe
  - podział tabeli na partycje
  - każda partycja to oddzielny segment
  - rekordy są automatycznie umieszczane w odpowiedniej partycji na podstawie wartości kolumny partycjonującej

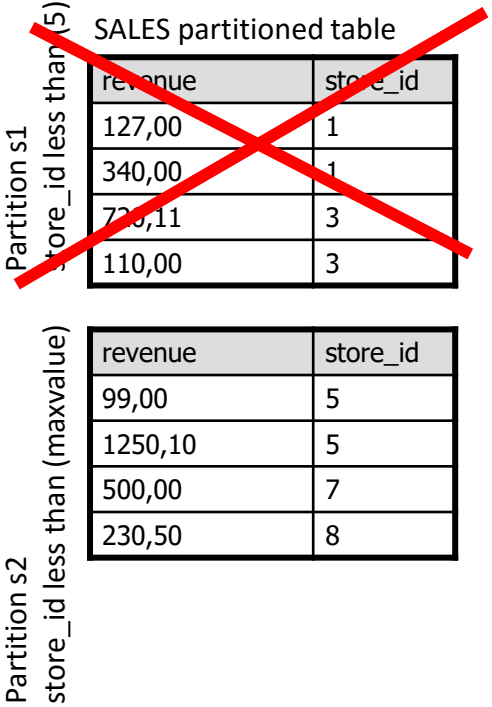
METODY PARTYCJONOWANIA TABEL

partycjonowanie jednopoziomowe i dwupoziomowe

- Jednopoziomowe:
  - zakresowe (Range)
  - haszowe (Hash)
  - listowe (List)
- Dwupoziomowe (subpartycje)
  - Range-hash
  - Range-list
  - Range-range
  - List-list
  - List-range
- Pozostałe
  - interwałowe (Interval)
  - systemowe (System)
  - referencyjne (Reference)

TABELE PARTYCJONOWANE: WYDAJNOŚĆ

korzyści z partycjonowania tabel - Partition Pruning

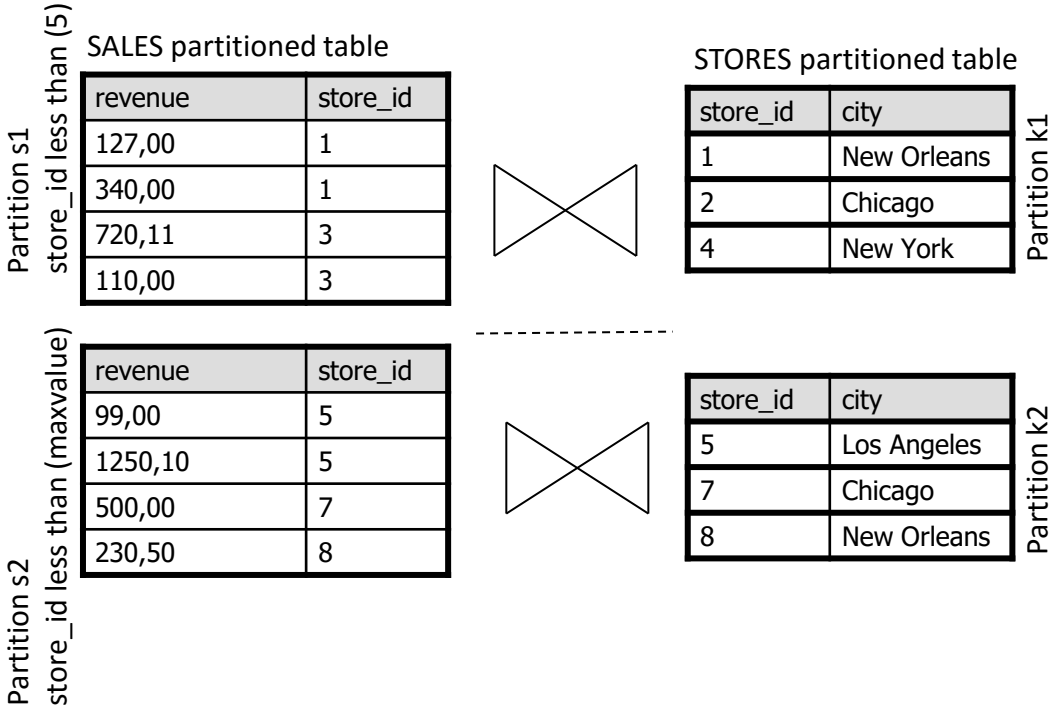


- Tabela partycjonowana
- Optymalizator zapytań pomija odczyt partycji, które na pewno nie zawierają szukanych rekordów
- Przykład zapytania:

```
SELECT SUM(revenue)
FROM stores
WHERE store_id BETWEEN 6 AND 8
```

TABELE PARTYCJONOWANE: WYDAJNOŚĆ

korzyści z partycjonowania tabel - Partition-Wise Joins



- Tabele ekwipartycjonowane (taki sam schemat partycjonowania)
- Połączenie tabel może być realizowane jako połączenia par partycji
- Przykład zapytania:

```
SELECT SUM(revenue)
FROM sales NATURAL JOIN stores
WHERE city='New Orleans'
```

PARTYCJE: OPERACJE ADMINISTRACYJNE

operacje na tabelach partycjonowanych - polecenie ALTER TABLE

- Dodawanie partycji i subpartycji (`ADD PARTITION`)
- Zmiana liczby haszowych partycji i subpartycji (`COALESCE PARTITION`)
- Usuwanie partycji i subpartycji (`DROP PARTITION`)
- Zastąpienie partycji i subpartycji tabelą (`EXCHANGE PARTITION`)
- Scalanie partycji i subpartycji (`MERGE PARTITIONS`)
- Modyfikacja parametrów domyślnych (`MODIFY DEFAULT ATTRIBUTES`)
- Modyfikacja parametrów partycji i subpartycji (`MODIFY PARTITION`)
- Dodawanie/usuwanie wartości do list partycjonowania listowego (`MODIFY PARTITION ADD/DROP VALUES`)
- Przenoszenie partycji i subpartycji (`MOVE PARTITION`)
- Zmiana nazwy partycji i subartycji (`RENAME PARTITION`)
- Podział partycji i subpartycji (`SPLIT PARTITION`)
- Obcinanie partycji i subpartycji (`TRUNCATE PARTITION`)

JAK UTWORZYĆ TABELĘ KOMPRESOWANĄ

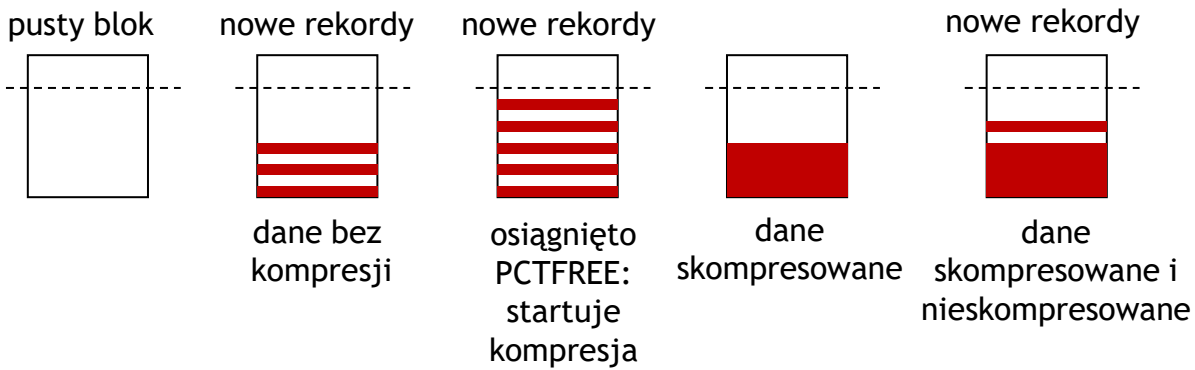
polecenie SQL i własności

```
create table DEMO (
   ID number(8),
   NAZWA varchar2(100),
   CENA number(8,2))
tablespace USERS
pctfree 20
row store compress basic
```

- Metody kompresji:
  - row basic
  - row advanced
  - [column for query]
  - [column for archive]
- Kompresja na poziomie bloku
- Każdy blok tabeli jest kompresowany oddzielnie
- Blok w Buffer Cache także jest w postaci skompresowanej

JAK DZIAŁA KOMPRESJA BLOKÓW

w tabelach kompresowanych



- Kompresja bloku jest wykonywana w chwili osiągnięcia poziomu wypełnienia (100%-PCTFREE)

Jak utworzyć tabelę zewnętrzną

polecenie SQL i własności

```
create directory DYSK1 as '/home/oracle';

create table DEMO (
   ID number(8),
   NAZWA varchar2(100),
   CENA number(8,2))
organization external (
  type ORACLE_LOADER
  default directory DYSK1
  access parameters (
    records delimited by newline
    load when (ID != BLANKS)
    fields terminated by ';'
    (
      ID char(8),
      NAZWA char(100),
      CENA char(10)))
  location ('DANE.TXT'))
```

- Plik DANE.TXT:

```
1;LenovoX1 Carbon;7999
2;Dell Inspiron 15;2500
3;HP Stream 13;990
```

- Plik z danymi odczytywany podczas każdego zapytania do tabeli zewnętrznej

JAK UTWORZYĆ TABELĘ TYMCZASOWĄ

polecenie SQL i własności

```
create global temporary table DEMO (
   ID number(8),
   NAZWA varchar2(100),
   CENA number(8,2))
```

- Zapisywana w tymczasowej przestrzeni tabel
- Ograniczenia
  - nie może być partycjonowana, klastrowana, IOT
  - nie może posiadać kluczy obcych
  - nie może posiadać tabel zagnieżdżonych
  - nie stosuje równoległych UPDATE, DELETE i MERGE are not supported
  - nie obsługuje transakcji rozproszonych
  - nie może zawierać kolumn INVISIBLE

WŁASNOŚCI TABEL TYMCZASOWYCH

trwałość definicji, nietrwałość danych

- Definicja tabeli jest trwała
- Dane tabeli są nietrwałe
  - są tracone po zakończeniu sesji (`ON COMMIT PRESERVE ROWS`) lub
  - po zakończeniu bieżącej transakcji (`ON COMMIT DELETE ROWS`)
- Definicja tabeli tymczasowej jest widoczna dla wszystkich uprawnionych sesji
- Dane tabeli tymczasowej są widoczne tylko dla sesji, która je wprowadziła do tabeli
- Mogą być wykorzystywane przez programistów jako pamięć prywatna, przeznaczona do przetwarzania dużych zbiorów danych
- Mogą mieć indeksy, ograniczenia integralnościowe i wyzwalacze
- Mogą być wykorzystywane przez perspektywy i synonimy oraz łączone z innymi tabelami
- Dlaczego warto korzystać z tabel tymczasowych?
  - redukcja ilości generowanych danych Undo i Redo
  - poprawa wydajności przetwarzania danych tymczasowych

JAK UTWORZYĆ TABELĘ SZYFROWANĄ

polecenie SQL i własności

```
create table DEMO (
   ID number(8),
   NAZWA varchar2(100) encrypt,
   CENA number(8,2))
```

Wyłączenie szyfrowania kolumny:

```
alter table DEMO modify NAZWA decrypt
```

- Wymagane jest wcześniejsze przygotowanie i otwarcie portfela z kluczami szyfrującymi na poziomie instancji Oracle
- Możliwość wskazania algorytmu kryptograficznego, np.:
  - `encrypt using 'AES256' no salt`
- Dane zachowują postać zaszyfrowaną w Buffer Cache, plikach danych i dziennika powtórzeń, kopiach bezpieczeństwa
- Zajmują więcej miejsca na dysku

JAK PRZYGOTOWAĆ PORTFEL Z KLUCZEM SZYFRUJĄCYM

na poziomie instancji Oracle

- Wskaż lokalizację pliku portfela w sqlnet.ora

```
ENCRYPTION_WALLET_LOCATION=
  (SOURCE=(METHOD=FILE)(METHOD_DATA=
  (DIRECTORY=/u01/app/oracle/product/12.1.0/dbhome_1/wallet)))
```

- Zrestartuj Listenera

```
lsnrctl stop
lsnrctl start
```

- Wygeneruj portfel z Master Key (zabezpiecza właściwe klucze w słowniku danych), ustaw hasło do portfela

```
SQL> alter system set encryption key identified by „pass";
```

- Przy każdym starcie instancji Oracle otwieraj portfel za pomocą hasła

```
SQL> alter system set wallet open identified by „pass";
```

TABELE: OPERACJE ADMINISTRACYJNE

zmiana struktury tabeli

- Dodawanie kolumny:
  - `alter table EMP add (DEPT_ID number)`
- Modyfikacja definicji kolumny:
  - `alter table EMP modify (NAME varchar2(100))`
- Usuwanie kolumny:
  - `alter table EMP drop column COMM`
- Oznaczenie kolumny jaki niepotrzebnej (do późniejszego usunięcia) :
  - `alter table EMP set unused column COMM`
- Usunięcie wszystkich kolumn zaznaczonych jako niepotrzebne:
  - `alter table EMP drop unused columns`
- Zmiana nazwy kolumny:
  - `alter table EMP rename column COMM to C`
- Przełączenie tabeli w tryb read-only:
  - `alter table EMP read only`
- Przeniesienie tabeli do innej przestrzeni tabel:
  - `alter table EMP move tablespace USERS`

TABELE: OPERACJE ADMINISTRACYJNE

usuwanie i obcinanie tabeli

- TRUNCATE TABLE:
  - usuwa wszystkie rekordy tabeli, pozostawia jej definicję
- DROP TABLE:
  - usuwa wszystkie rekordy tabeli oraz jej definicję
  - można odzyskać usuniętą tabelę z RECYCLE BIN jeśli nie wykonano PURGE (RECYCLE BIN może być wyłączony)
- TRUNCATE TABLE vs. DELETE
  - TRUNCATE to komenda DDL, wykonująca niejawny COMMIT
  - DELETE to komenda DML (ROLLBACK jest możliwy)
  - DELETE może usuwać wybrane rekordy (WHERE), TRUNCATE zawsze usuwa wszystkie
  - TRUNCATE jest szybsze
    - DELETE indywidualnie usuwa każdy rekord, TRUNCATE zwalnia zaalokowaną pamięć

JAKIE TABELE ZNAJDUJĄ SIĘ W BAZIE DANYCH

perspektywa DBA_TABLES

```
SQL> select owner, table_name, tablespace_name, pct_free,
            read_only, compression, temporary, partitioned
     from dba_tables where owner='HR';

OWNER TABLE_NAME  TABLESPACE_NAME PCT_FREE REA COMPRESS T PAR
----- ----------- --------------- -------- --- -------- - ---
HR    COUNTRIES   EXAMPLE          0       NO  DISABLED N NO
HR    JOB_HISTORY EXAMPLE         10       NO  DISABLED N NO
HR    EMPLOYEES   EXAMPLE         10       NO  DISABLED N NO
HR    JOBS        EXAMPLE         10       NO  DISABLED N NO
HR    DEPARTMENTS EXAMPLE         10       NO  DISABLED N NO
HR    LOCATIONS   EXAMPLE         10       NO  DISABLED N NO
HR    REGIONS     EXAMPLE         10       NO  DISABLED N NO
```

OGRANICZENIA INTEGRALNOŚCIOWE DLA TABEL

rodzaje i własności

- Ochrona poprawności danych
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - REFERENCES
  - CHECK
- Zarządzanie ograniczeniami integralnościowymi
  - ALTER TABLE <table> ENABLE CONSTRAINT <constraint>
  - ALTER TABLE <table> DISABLE CONSTRAINT <constraint>
  - ALTER TABLE <table> DROP CONSTRAINT <constraint>
  - ALTER TABLE <table> DROP CONSTRAINT <constraint> CASCADE
  - ALTER TABLE <table> ADD CONSTRAINT <constraint> <definition>
  - DROP TABLE <table> CASCADE CONSTRAINTS

OGRANICZENIA INTEGRALNOŚCIOWE DLA TABEL

stan ograniczeń integralnościowych

- Ograniczenie integralnościowe może przyjąć jeden ze stanów:
  - `ENABLE VALIDATE` – musi być spełnione przez nowe i istniejące rekordy
  - `ENABLE NOVALIDATE` – musi być spełnione przez nowe rekordy
  - `DISABLE VALIDATE` – musi być spełnione przez istniejące rekordy
  - `DISABLE NOVALIDATE` – brak walidacji
  - `RELY` – brak walidacji, jedynie wskazówka dla optymalizatora zapytań
- Moment walidacji ograniczeń integralnościowych:
  - `IMMEDIATE` – w chwili wykonywania modyfikacji danych (domyślnie)
  - `DEFERRED` – w chwili zatwierdzania transakcji (konieczne użycie `DEFERRABLE` w definicji ograniczenia integralnościowego)
    - `INITIALLY DEFERRED` – domyślnie `DEFERRED`
    - `INITIALLY IMMEDIATE` – domyślne `IMMEDIATE`, można przełączyć w `DEFERRED` za pomocą "`SET CONSTRAINTS ... DEFERRED`"

JAKIE OGRANICZENIA INTEGRALNOŚCIOWE MA TABELA

perspektywy dba_constraints i dba_cons_columns

```
SQL> select constraint_name, constraint_type, deferrable,
     deferred, search_condition
     from dba_constraints
     where owner='SCOTT' and table_name='DEMO';

CONSTRAINT_NAME C DEFERRABLE     DEFERRED  SEARCH_CONDITION
--------------- - -------------- --------- ------------------
SYS_C0010353    P NOT DEFERRABLE IMMEDIATE
SYS_C0010352    C NOT DEFERRABLE IMMEDIATE CENA>0
SYS_C0010351    C NOT DEFERRABLE IMMEDIATE "NAZWA" IS NOT NULL

SQL> select constraint_name, column_name from dba_cons_columns
     where owner='SCOTT' and table_name='DEMO';

CONSTRAINT_NAME COLUMN_NAME
--------------- -----------
SYS_C0010353    ID
SYS_C0010352    CENA
SYS_C0010351    NAZWA
```

```
create table DEMO (
   ID number(8) PRIMARY KEY,
   NAZWA varchar2(100) NOT NULL,
   CENA number(8,2) CHECK
      (CENA>0))
```