Zastosowanie perspektyw słownika danych oraz perspektyw dynamicznych w administracji bazą danych Oracle8

Gerard Głowacki
Jeronimo Martins Dystrybucja
e–mail: gerardg@novci2.ae.poznan.pl

Abstrakt. Na rynku istnieje szereg pakietów wspomagających administrowanie i monitorowanie pracy bazy danych. Jednak w wielu wypadkach istnieje konieczność dotarcia do bardziej precyzyjnych informacji. Perspektywy dynamiczne, oraz perspektywy słownika danych stanowią cenne źródło informacji o systemie bazy danych Oracle8. W artykule podejmuje się próbę przedstawienia praktycznego wykorzystania perspektyw dynamicznych, oraz perspektyw słownika danych w administracji systemem Oracle8.

1. Wstęp

Katalog danych (słownik danych i perspektywy dynamiczne) jest zbiorem tabel i perspektyw umożliwiających wyszukanie wartościowych informacji o bazie danych. Definicje perspektyw znajdują się w pliku catalog.sql , który powinien być uruchomiony po utworzeniu bazy danych.

W niniejszym opracowaniu wybrane perspektywy słownika danych, oraz perspektywy dynamiczne zostaną przedstawione w grupach, logicznie związanych z pewnym aspektem funkcjonowania bazy danych. Każdy grupa perspektyw opatrzona jest krótkim opisem merytorycznym, przedstawiony jest rysunek obrazujący powiązania pomiędzy perspektywami, oraz kilka przykładowych zapytań .

Przygotowując niniejszy referat autor dołożył wszelkich starań, aby był on jak najbardziej rzetelny. Autor nie ponosi jakiejkolwiek odpowiedzialności za szkody (straty, jak również utracone potencjalne korzyści) pozostające w pośrednim lub bezpośrednim związku z lekturą i wykorzystaniem niniejszego opracowania.

2. Wykorzystanie perspektyw katalogu danych

Perspektywy katalogu danych można podzielić na trzy grupy :

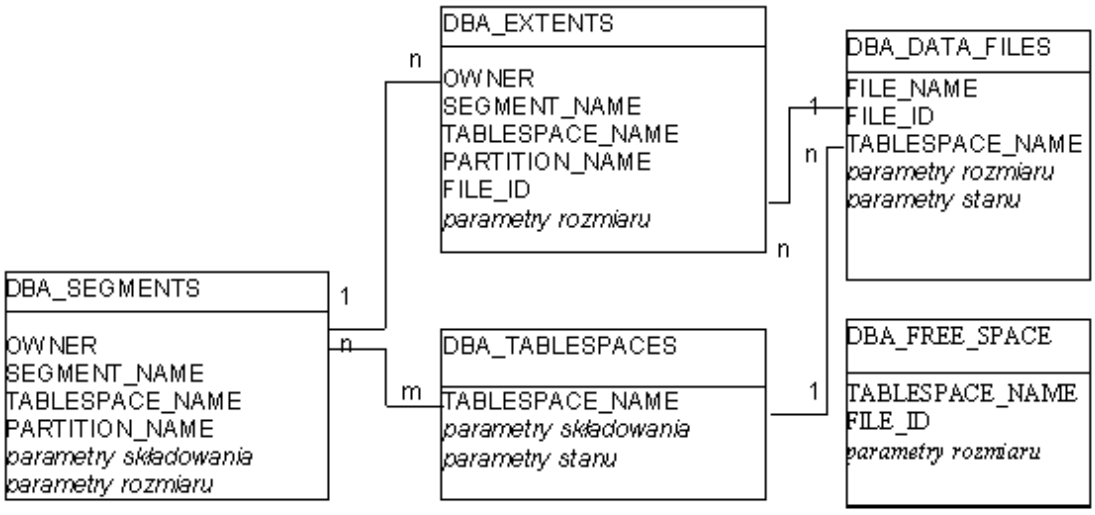
- ALL - zawierają wszystkie dostępne informacje dla użytkownika;
- DBA - zawierają informacje o wszystkich obiektach w bazie danych (do ich przeglądanie potrzebne jest przywilej SELECT ANY TABLE)
- USER - zawierają informacje dotyczące schematu użytkownika.

W dalszej części będą rozważane perspektywy z przedrostkiem DBA, wśród których zostaną wyróżnione następujące grupy :

- perspektywy dostarczające informacji o wykorzystaniu przestrzeni;
- perspektywy dostarczające informacji o tabelach oraz ich rozmieszczeniu w przestrzeni tabel;
- perspektywy dostarczające informacji o indeksach, oraz ich rozmieszczeniu w przestrzeni tabel;
- perspektywy dostarczające informacji o użytkownikach i ich profilach;
- perspektywy dostarczające informacji o rolach oraz uprawnieniach przyznanych użytkownikom i rolom.

2.1. Perspektywy dostarczające informacji o wykorzystaniu przestrzeni

Najmniejszą jednostką, która może być przenoszona z dysku do pamięci i z powrotem jest blok. Logicznie sąsiednie (możliwe do znalezienia w oparciu o adres przesunięcia bloku w stosunku do początku pliku) bloki tworzą zakresy (rozszerzenia - ang. extends). Zakres musi znajdować się w jednym pliku systemu operacyjnego. Kilka - kilkaset zakresów tworzy segmenty służące do przechowywania połączonych ze sobą danych (np. segment tabeli, segment indeksu, segment do wycofywania transakcji). Segment może być zbyt duży aby zmieścić się w jednym pliku. Baza danych jest podzielona na jednostki logiczne - obszary tabel. Każdy segment lub partycja segmentu musi w całości zmieścić się w jednym obszarze tabel. Obszar tabel składa się z wielu plików systemu operacyjnego.



Rys. 1. Powiązania pomiędzy perspektywami dostarczającymi informacji o wykorzystaniu przestrzeni

DBA_EXTENTS - parametry zakresu (ang. Extent) - numer w segmencie (EXTENT_ID), identyfikator pliku (FILE_ID), identyfikator bloku startowego (BLOCK_ID), rozmiar w bajtach oraz w blokach (BYTES, BLOCKS);

DBA_SEGMENTS - parametry segmentu - identyfikator pliku z nagłówkiem (HEADER_FILE), rozmiar w bajtach i w blokach (BYTES, BLOCKS), parametry składowania (INITIAL_EXTENTS, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE);

DBA_TABLESPACE - parametry przestrzeni tabel - parametry składowania (INITIAL_EXTENTS, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE), parametry określające stan przestrzeni tabel (STATUS [ONLINE, OFFLINE, READONLY], CONTENTS [PERNAMENT, TEMPORARY], LOGGING);

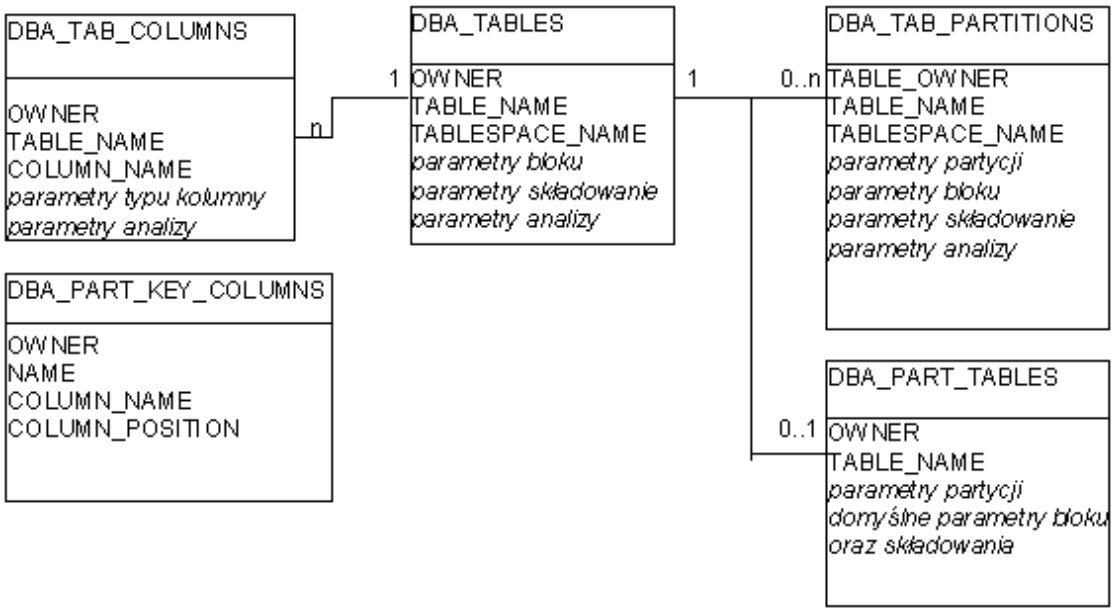
DBA_DATA_FILES - parametry pliku - parametry rozmiaru (BLOCKS, BYTES, MAXBLOCKS, MAXBYTES) , parametry stanu (STATUS, AUTOEXTENSIBLE);

DBA_FREE_SPACE - parametry wolnego miejsca - parametry rozmiary wolnych extentów (BYTES, BLOCKS), identyfikator bloku startowego

2.2. Perspektywy dostarczające informacji o tabelach i ich rozmieszczeniu w przestrzeni tabel

Podczas tworzenia tabeli poleceniem CREATE TABLE można podać parametr oznaczający nazwę przestrzeni tabel, w której będą składowane dane tabeli, a także parametry dotyczące

składowania. Domyślnie wykorzystywana jest przestrzeń tabel użytkownika tworzącego tabelę, oraz parametry składowania dla wyspecyfikowanej lub domyślnej przestrzeni tabel. W przypadku bardzo dużych tabel możliwa jest ich dekompozycje na mniejsze umożliwiające łatwe zarządzanie fragmenty zwane partycjami np. duża tabelę zawierającą dane z całego okresu rozliczeniowego można podzielić np. kwartalnie. Każdą z partycji można następnie umiejscowić na osobnej przestrzeni tabel definiując parametry składowania.



Rys. 2. Informacje o tabelach oraz ich rozmieszczeniu w przestrzeni tabel.

DBA_TABLES - parametry tabeli - parametry bloku (PCT_FREE, PCT_USED), parametry składowania (NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE), parametry analizy (NUM_ROWS, BLOCKS, EMPTY_BLOCKS, AVG_SPACE ...) ;

DBA_TAB_COLUMNS - parametry kolumn - parametry typy (DATA_TYPE, DATA_LENGTH, DATA_PRECISION, DATA_SCALE), parametry analizy (LOW_VALUE, HIGH_VALUE, NUM_DISTINCT, DENSITY, LAST_ANALYZED) ;

DBA_TAB_PARTITIONS - parametry partycji tabeli - parametry partycji (PARTITION_NAME, HIGH_VALUE, HIGH_VALUE_LENGTH, PARTITION_POSITION), parametry bloku (PCT_FREE, PCT_USED), parametry składowania (NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE), parametry analizy (NUM_ROWS, BLOCKS, EMPTY_BLOCKS, AVG_SPACE ...) ;

DBA_PART_TABLES - parametry tabeli partycjonowanej - parametry partycji (PARTITION_TYPE, PARTITION_COUNT, PARTITION_KEY_COUNT), domyślne parametry bloku oraz składowania dla instrukcji ADD PARTITION ;

DBA_PART_KEY_COLUMNS - kolumny wchodzące w skład klucza partycji ;

Do interesujących informacji mogą należeć :

- W jakim stopniu aktualnie składowane dane są pofragmentowane ?

```
select dt.tablespace_name, dt.owner, dt.table_name, dt.next_extent,
sum(de.bytes) USED ,  count(*) NO_EXTENTS
from dba_tables dt, dba_extents de where dt.tablespace_name =
de.tablespace_name and dt.table_name=de.segment_name group by
dt.tablespace_name, dt.table_name, dt.owner, dt.next_extent ;
```

- Ile jest wolnego miejsca na przestrzeni tabel ?

```
select dfs.tablespace_name, sum(dfs.bytes) from dba_free_space dfs group by
dfs.tablespace_name;
```

- W jakim stopniu wolna przestrzeń jest zdefragmentowana

```
select bytes, count(*), sum(bytes) from dba_free_space
where tablespace_name='Moja przestrzen'
group by tablespace_name;
```

- Zapytanie wiążące informacje o aktualnej zajętości z informacją o ilości wolnego miejsca dla tabel niepartycjonowanych; (z1 - podzapytanie zwraca wielkość kolejnego rozszerzenia, maksymalną liczbę dostępnych rozszerzeń, sumaryczną ilość zajętego miejsca, ilość wykorzystanych rozszerzeń; z2 - ilość potencjalnie wolnego miejsca na przestrzeni tabel, z3 - zwraca ilość wolnego miejsca, które może być alokowane na potrzeby danej tabeli - oczywiście te wielkości zmieniają się dynamicznie w zależności od rozmieszczenia pozostałych innych elementów na przestrzeni tabel, a także ustawień PCT_INCREASE).

```
SELECT z1.tablespace_name TABLESPACE, z1.owner OWNER, z1.table_name
TABLE_NAME, z1.used USED, z1.max_extents, z1.NO_EXTENTS_USED,
z1.next_extent NEXT_EXTENT, z2.total_free TOTAL_FREE, z3.FREE
from (select dt.tablespace_name, dt.owner, dt.table_name, dt.next_extent,
dt.max_extents, sum(de.bytes) USED , count(*) NO_EXTENTS_USED
from dba_tables dt, dba_extents de where dt.tablespace_name =
de.tablespace_name and dt.table_name=de.segment_name group by
dt.tablespace_name, dt.table_name, dt.owner, dt.next_extent, dt.max_extents
) z1 ,
( select dfs.tablespace_name, sum(dfs.bytes) TOTAL_FREE from dba_free_space
dfs group by dfs.tablespace_name ) z2 ,
( select dfs.tablespace_name, dt.table_name, sum(dfs.bytes) FREE from
dba_free_space dfs, dba_tables dt where dfs.tablespace_name =
dt.tablespace_name and dfs.bytes >= dt.next_extent
group by dfs.tablespace_name, dt.table_name) z3
where z1.tablespace_name = z2.tablespace_name and
z1.tablespace_name=z3.tablespace_name and z1.table_name=z3.table_name;
```

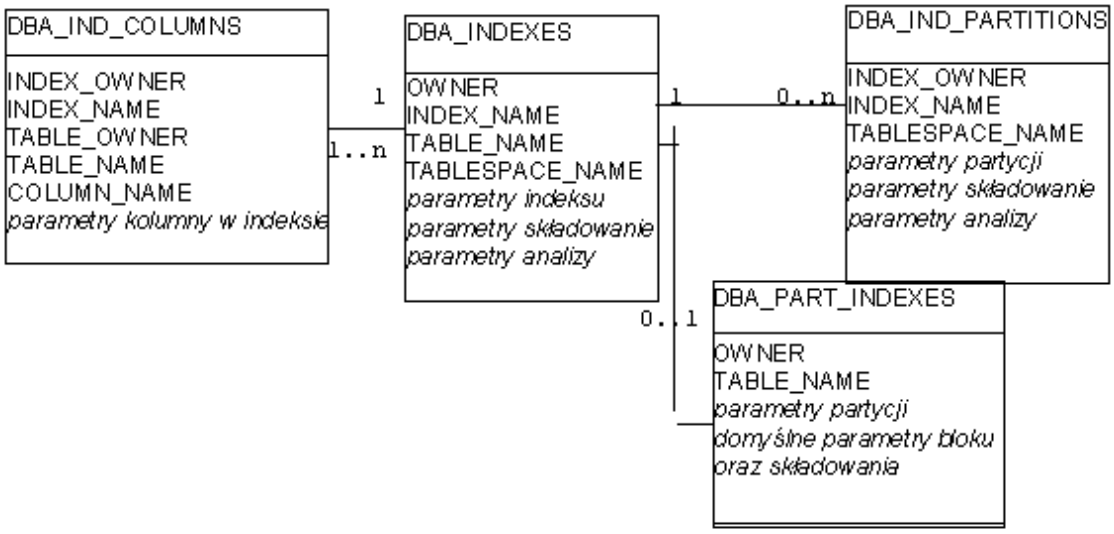
- Analogiczne zapytanie można zastosować w odniesieniu do tabeli partycjonowanej. (budowa tego zapytania jest analogiczna, wprowadzenie rozbicia z1 na q1 i q2 wynika z braku możliwości grupowania na podstawie pola HIGH_VALUE);

```
SELECT q1.tablespace_name TABLESPACE, q1.partition_name PARTITION,
q1.table_name TABLE_NAME,  q2.suma USED,  q2.no_extents_used,
q2.max_extents,  q1.next_extent NEXT_EXTENT,
q4.total_free TOTAL_FREE,  q3.free FREE, q1.high_value HIGH_VALUE
FROM ( SELECT tablespace_name, partition_name, table_name, high_value,
next_extent FROM dba_tab_partitions ) q1,
( SELECT dtp.tablespace_name, dtp.partition_name, dtp.table_name,
dtp.max_extents, SUM(de.bytes) suma , count(de.bytes) NO_EXTENTS_USED,
FROM dba_tab_partitions dtp, dba_extents de
WHERE dtp.tablespace_name=de.tablespace_name and dtp.partition_name =
de.partition_name and dtp.table_name = de.segment_name
GROUP BY dtp.tablespace_name, dtp.partition_name, dtp.table_name,
dtp.max_extents) q2,
( SELECT dfs.tablespace_name, dtp.partition_name, dtp.table_name,
sum(dfs.bytes) FREE  FROM dba_free_space dfs, dba_tab_partitions dtp  WHERE
dfs.tablespace_name = dtp.tablespace_name  and dfs.bytes >= dtp.next_extent
GROUP BY dfs.tablespace_name, dtp.partition_name, dtp.table_name) q3,
( SELECT tablespace_name, sum(bytes) TOTAL_FREE from dba_free_space
  GROUP BY tablespace_name) q4
WHERE  q1.tablespace_name = q3.tablespace_name and  q1.partition_name =
q3.partition_name and q1.table_name = q3.table_name and  q1.tablespace_name
```

```
= q4.tablespace_name and q1.tablespace_name = q2.tablespace_name and
q1.partition_name = q2.partition_name and q1.table_name = q2.table_name );
```

2.3. Perspektywy dostarczające informacji o tabelach i ich rozmieszczeniu w przestrzeni tabel

Celem tworzenia indeksów jest zwiększenie efektywności wykonywania zapytań do bazy danych. W przypadku tabel partycjonowanych tworzony jest indeks partycjonowany. Może być lokalny - wówczas wszystkie klucze w partycji indeksu odwołują się tylko do rekordów znajdujących w jednej partycji tabeli. Zasady partycjonowania indeksu lokalnego są identyczne jak w przypadku tabeli. Indeks lokalny jest indeksem prefixowanym (ang. local prefixed index), jeżeli kluczem partycjonowania jest lewy podzbiór kolumn indeksu. (w przeciwnym wypadku indeks jest nie prefixowany). W przypadku indeksu globalnego klucze indeksu w jednej partycji mogą odwoływać się do rekordów znajdujących się w wielu partycjach tabeli. Indeks globalny może być podobnie jak lokalny - prefixowany lub nie prefixowany.



Rys. 3. Informacje o indeksach oraz ich rozmieszczeniu w przestrzeni tabel.

DBA_INDEXES - parametry indeksu (INDEX_TYPE, UNIQUENESS), parametry składowania ((NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE), parametry analizy (LEAF_BLOCKS, DISTINCT_KEYS, AVG_LEAF_BLOCKS_PER_KEY, ...)

DBA_IND_COLUMNS - parametry kolumn indeksu (COLUMN_POSITION, COLUMN_LENGTH)

DBA_IND_PARTITIONS - parametry partycji indeksu - parametry partycji (PARTITION_NAME, HIGH_VALUE, HIGH_VALUE_LENGTH, PARTITION_POSITION, TABLES PACE_NAME), parametry składowania (NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE), parametry analizy (NUM_ROWS, BLOCKS, EMPTY_BLOCKS, AVG_SPACE ...)

DBA_PART_INDEX - parametry indeksu partycjonowanego - parametry partycji (PARTITION_TYPE, PARTITION_COUNT, PARTITION_KEY_COUNT), typ indeksu (LOCALITY, ALIGNMENT) , domyślne parametry bloku oraz składowania dla instrukcji ADD PARTITION

Do często zadawanych zapytań należą :

- jakie indeksy są założone na podanej tabeli :

```
SELECT table_name , index_name, index_type , uniqueness, partitioned FROM
dba_indexes WHERE table_name= 'Moja_tabelka'
```

- z jakich kolumn zbudowany jest indeks :

```
SELECT * FROM dba_ind_columns WHERE table_name = 'Moja_tabelka' ORDER BY
index_name, column_position;
```

- jakie jest rozmieszczenie indeksu (nie partycjonowanego) w stosunku do tabeli

```
SELECT dt.table_name, dt.tablespace_name, di.index_name, di.tablespace_name
FROM dba_tables dt, dba_indexes di
WHERE dt.table_name = di.table_name and
dt.owner = di.table_owner;
```

- jakie kolumny są kluczem partycjonowania :

```
SELECT * FROM dba_part_key_columns WHERE name = 'Mój_indeks';
```

- jakie sa parametry indeksu partycjonowanego (lokalny, globalny, ilość partycji) :

```
SELECT index_name, partitioning_type, partition_count,
partitioning_key_count, locality, alignment FROM dba_part_indexes WHERE
index_name = 'My index' ;
```

- ile miejsca zajmuje indeks, ile jest jeszcze wolnego .(dla indeksu nie partycjonowanego)

```
SELECT q1.tablespace_name TABLESPACE, q1.owner OWNER, q1.index_name
INDEX_NAME, q1.table_name TABLE_NAME, q1.USED USED, q1.next_extent
NEXT_EXTENT, q1.EXTENTS_NO, q2.total_free TOTAL_FREE, q3.free FREE
from
( SELECT di.tablespace_name, di.owner, di.index_name, di.table_name,
di.next_extent, sum(de.bytes) USED, COUNT(*) EXTENTS_NO   FROM  dba_indexes
di, dba_extents de
WHERE di.tablespace_name = de.tablespace_name and
di.index_name = de.segment_name
GROUP BY  di.tablespace_name, di.owner, di.index_name, di.table_name,
di.next_extent) q1,
( SELECT dfs.tablespace_name, sum(dfs.bytes) TOTAL_FREE from dba_free_space
dfs GROUP BY dfs.tablespace_name ) q2,
( SELECT dfs.tablespace_name, di.index_name, sum(dfs.bytes) FREE
FROM dba_free_space dfs, dba_indexes di
WHERE dfs.tablespace_name = di.tablespace_name and
dfs.bytes >= di.next_extent
GROUP BY dfs.tablespace_name, di.index_name ) q3
WHERE q1.tablespace_name = q2.tablespace_name and
q1.tablespace_name = q3.tablespace_name and q1.index_name = q3.index_name;
```

- jak rozmieszczony jest, jaki obszar zajmuje indeks partycjonowany ?

```
SELECT q1.tablespace_name TABLESPACE, q1.partition_name PARTITION,
q1.index_name INDEX_NAME, q1.table_name TABLE_NAME, q2.suma USED, q2.
Extents_no EXTENTS_NO, q1.next_extent NEXT_EXTENT, q4.total_free TOTAL_FREE,
   q3.free FREE, q1.high_value HIGH_VALUE
FROM
( SELECT dip.tablespace_name, dip.partition_name, dip.index_name,
di.table_name, dip.high_value, dip.next_extent
FROM dba_ind_partitions dip, dba_indexes di
WHERE dip.index_name = di.index_name ) q1,
( SELECT dip.tablespace_name, dip.partition_name, dip.index_name,
SUM(de.bytes) suma, count(*) EXTENTS_NO
FROM dba_ind_partitions dip, dba_extents de
WHERE dip.tablespace_name = de.tablespace_name and
dip.partition_name=de.partition_name and dip.index_name = de.segment_name
GROUP BY dip.tablespace_name, dip.partition_name, dip.index_name ) q2,
```

```
( SELECT dfs.tablespace_name, dip.partition_name, dip.index_name,
sum(dfs.bytes) FREE FROM dba_free_space dfs, dba_ind_partitions dip
WHERE dfs.tablespace_name = dip.tablespace_name and dfs.bytes >=
dip.next_extent GROUP BY dfs.tablespace_name, dip.partition_name,
dip.index_name ) q3,
( SELECT tablespace_name, sum(bytes) TOTAL_FREE from dba_free_space GROUP BY
tablespace_name ) q4
WHERE q1.tablespace_name = q3.tablespace_name and q1.partition_name =
q3.partition_name and  q1.index_name = q3.index_name and q1.tablespace_name
= q4.tablespace_name and  q1.tablespace_name = q2.tablespace_name and
q1.partition_name = q2.partition_name and q1.index_name = q2.index_name
```

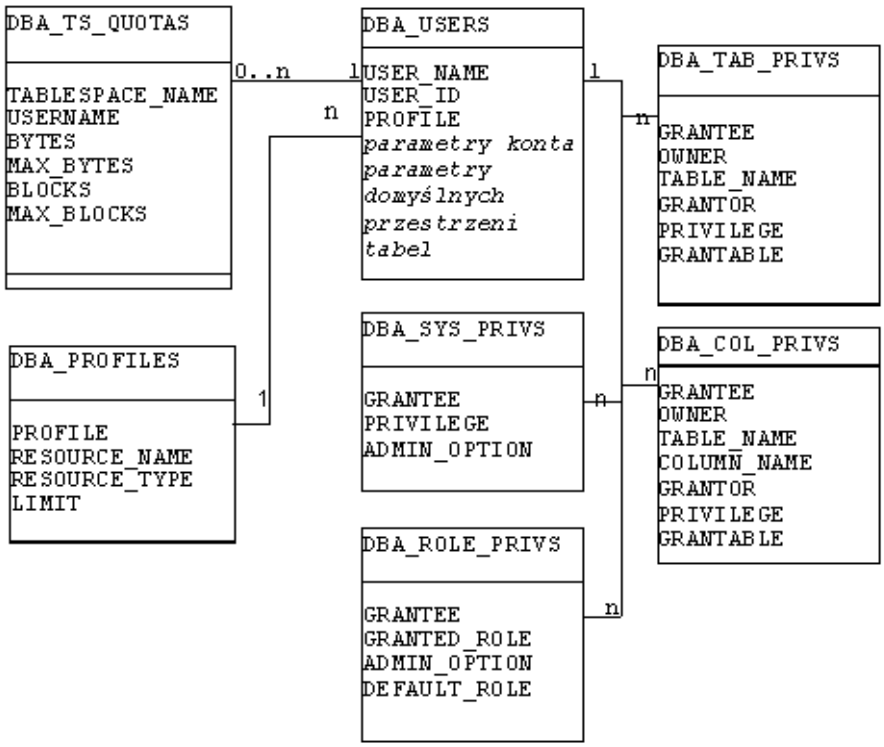
2.4. Perspektywy dostarczające informacji o użytkownikach, profilach i uprawnieniach

Użytkownik bazy danych tworzy obiekty w przestrzeni logicznej zwanej schematem użytkownika. Użytkownik może posiadać uprawnienia systemowe (np. do tworzenia sesji, tworzenia obiektów itd.) oraz uprawnienia dostępu do obiektów (np. do odczytu tabeli, wykonania procedury).

Rola to zbiór uprawnień systemowych oraz praw dostępu, tworzona ze względu na uproszczenie zarządzania dostępem.

Profil użytkownika to zbiór ograniczeń zasobowych (np. wykorzystanie procesora, czas pracy użytkownika, liczba sesji). Każdy użytkownik posiada jeden profil.

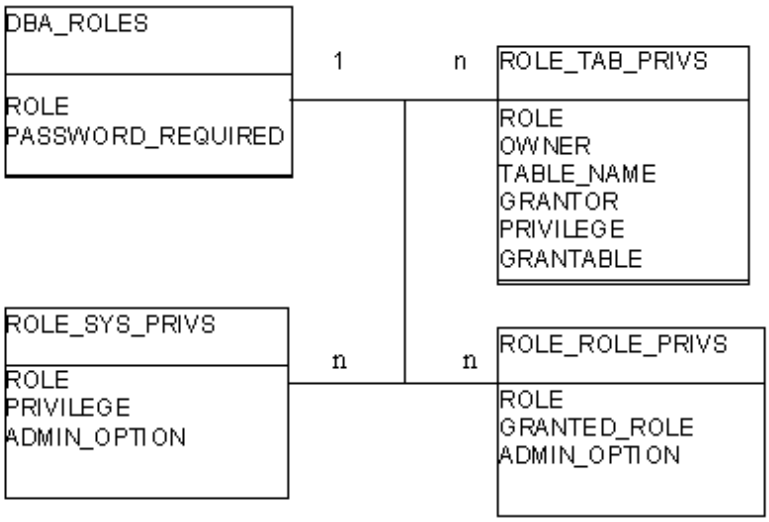
Dla każdego użytkownika można zdefiniować ograniczenia w wykorzystaniu poszczególnych przestrzeni tabel.



Rys.4. Użytkownicy, profile, uprawnienia

DBA_USERS - parametry konta (hasło, data utworzenia, profil, status konta), parametry domyślnych przestrzeni tabel (domyślna przestrzeń tabel dla danych, domyślna przestrzeń tabel dla tabel tymczasowych)

DBA_PROFILES	-	zawiera definicję profili użytkowników - czyli definicje limitów wykorzystania zasobów.
DBA_TS_QUOTAS	-	ograniczenia w wykorzystaniu przestrzeni tabel.
DBA_ROLE_PRIVS	-	role przydzielone do użytkowników i ról ,
DBA_TAB_PRIVS	-	przywileje obiektowe przyznane użytkownikom i rolom ,
DBA_SYS_PRIVS	-	przywileje systemowe przyznane użytkownikom i rolom ,
DBA_COL_PRIVS	-	przywileje do kolumn przyznane użytkownikom i rolom



Rys. 5. Role, przywileje.

Rola to zbiór uprawnień systemowych oraz obiektowych. Istnieje możliwość przydzielania ról innym rolom.

DBA_ROLES	- zdefiniowane w bazie role, z zaznaczeniem czy są identyfikowane hasłem;
ROLE_ROLE_PRIVS	- informacje o rolach przyznanych innym rolom;
ROLE_SYS_PRIVS	- informacje o przywilejach systemowych przyznanych rolom;
ROLE_TAB_PRIVS	- informacje o przywilejach obiektowych przyznanych rolom;

Do przedstawionych perspektyw można skomponować następujące zapytania :

- Którzy użytkownicy mają odblokowane konto :
```
SELECT * FROM dba_users WHERE account_status  = 'UNLOCKED';
```
- Którzy użytkownicy mają domyślny profil określony jako domyślny :
```
SELECT * FROM dba_users WHERE profile = 'DEFAULT'
```

- Komu zostało przyznane uprawnienie systemowe 'UMLIMITED TABLESPACE' (to uprawnienie nadpisuje ewentualne ograniczenia występujące w DBA_TS_QUOTAS co może mieć znaczący wpływ na zarządzanie przestrzenią)

```
SELECT * FROM dba_sys_privs WHERE privilege = 'UNLIMITED TABLESPACE'
```

- Kto posiada rolę DBA ?

```
SELECT * FROM dba_role_privs WHERE granted_role='DBA';
```

- Jakie uprawnienia obiektowe są zdefiniowane w bazie danych dla użytkownika na poziomie kolumn ?

```
SELECT * FROM dba_col_privs WHERE grantee = 'Uzytkownik'
```

- Często występującym w praktyce problemem jest przeniesienie użytkowników wraz ze zdefiniowanymi uprawnieniami na inną bazę danych np. środowisko testowe. Poniższe zapytania tworzą potrzebne do tego skrypty. Zakłada się , że potrzebne profile zostały już zdefiniowane, oraz że nie ma zdefiniowanych uprawnień obiektowych na poziomie kolumn. Hasła użytkowników są identyczne z ich identyfikatorami, zakłada się, że role nie są chronione hasłem.

Utworzenie użytkowników ...

```
SELECT  'CREATE USER ' || username ||  ' IDENTIFIED BY  ' || username ||
' DEFAULT TABLESPACE ' || default_tablespace ||' TEMPORARY TABLESPACE ' ||
temporary_tablespace || ' PROFILE ' || profile || ';'
FROM dba_users ORDER BY username;
```

Utworzenie ról ...

```
SELECT 'CREATE ROLE '|| role ||';'  FROM dba_roles;
```

Zdefiniowanie ograniczeń dla przestrzeni tabel ...

```
SELECT 'ALTER USER '|| tsq.username || ' QUOTA '
|| DECODE( tsq.max_bytes, -1, 'unlimited', tsq.max_bytes) || ' ON
'||tsq.tablespace_name ||';'
FROM dba_ts_quotas tsq ORDER BY tsq.username;
```

Przyznanie ról rolom ...

```
SELECT 'GRANT '|| granted_role ||' TO '|| role;
FROM role_role_privs ORDER BY role;
```

Przyznanie rolom uprawnień systemowych ...

```
SELECT 'GRANT '|| privilege || ' TO '|| role ||
DECODE (admin_option, 'YES', 'with admin option', 'NO',null,null) || ';'
FROM role_sys_privs ORDER BY role;
```

Przyznanie rolom uprawnień obiektowych ...

```
SELECT 'GRANT '|| privilege'||' ON '|| table_name|| ' to ' || role||
DECODE(grantable,' YES', ' with grant option ', 'NO', null, null )|| ';'
FROM role_tab_privs ORDER BY role;
```

Przyznanie ról użytkownikom ...

```
SELECT 'GRANT '|| granted_role || ' TO '|| grantee||
DECODE (admin_option,'YES',' with admin option ','NO', null, null ) || ';'
FROM dba_role_privs ORDER BY grantee;
```

Przyznanie uprawnień systemowych użytkownikom

```
SELECT 'GRANT '|| privilege || ' TO '|| grantee ||
DECODE (admin_option, 'YES', 'with admin option', 'NO',null,null) || ';'
FROM dba_sys_privs
```

```
ORDER BY grantee;
```

Przyznanie uprawnień obiektowych użytkownikom

```
SELECT 'GRANT '|| privilege'||' ON '|| table_name|| ' to ' || grantee ||
DECODE(grantable,' YES', ' with grant option ', 'NO', null, null )|| ';'
FROM dba_tab_privs ORDER BY grantee;
```

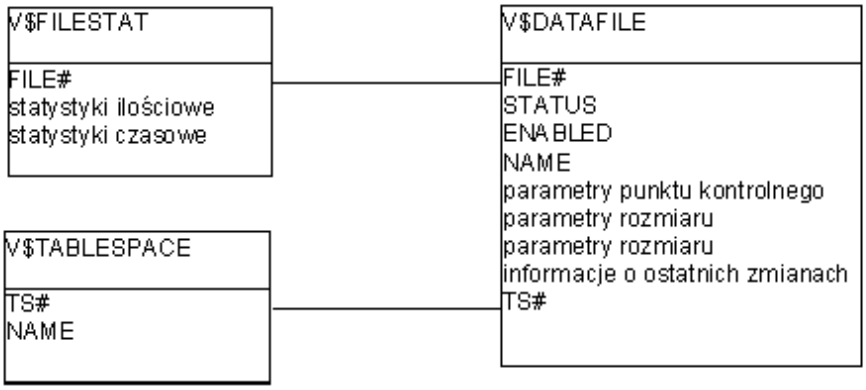
3. Wykorzystanie perspektyw dynamicznych

Perspektywy dynamiczne (ang. performance views) - są na bieżącą uaktualniane podczas działania bazy danych. Zawartość dotyczy przede wszystkim wykorzystania zasobów systemowych. Szczegółowiej omówione zostaną :

- perspektywy obrazujące intensywność operacji I/O na plikach z danymi;
- perspektywy obrazujące intensywność operacji na segmentach do wycofywania transakcji;
- perspektywy umożliwiające powiązanie informacji o wykorzystaniu segmentów do wycofywania transakcji z identyfikacją uruchomionych zapytań SQL;

Perspektywa v$fixed_table zawiera informacje o wszystkich perspektywach dynamicznych. Przydatne w formułowaniu zapytań informacje zawiera perspektywa v$indexed_fixed_column - zawiera informacje o kolumnach indeksowanych w tabelkach x$ - co w połączeniu z definicją perspektyw dostarcza wskazówek odnośnie efektywnego formułowania zapytań, oraz możliwości łączenia informacji w perspektywach.

3.1. Perspektywy dostarczające informacji o intensywności operacji na plikach z danymi



Rys. 6. Intensywność operacji wejścia / wyjścia na plikach z danymi.

V$FILESTAT - informacje o statystykach ilościowych (PHYRDS, PHYWRTS, PHYBLKRD, PHYBLKWRT - ilość odczytów i zapisów, oraz ilość odczytanych i zapisanych bloków dyskowych), informacje o statystykach czasowych (READTIM, WRITETIM - czas operacji odczytu, zapisu; AVGIOTIM - średni czas operacji I/O; LSTIOTIM - czas ostatniej operacji I/O, MINIOTIM - minimalny czas operacji I/O; MAXIOWTM, MAXIORTM - maksymalny czas pojedyńczej operacji zapisu/ odczytu

V$DATAFILE - parametry punktu kontrolnego (CHECKPOINT_CHANGE#, CHECKPOINT_TIME) , parametry rozmiaru (BYTES, CREATE_BYTES, BLOCKS, BLOCK_SIZE), informacje o ostatnich zmianach (LAST_CHANGE#, LAST_TIME, OFFLINE_CHANGE#, ONLINE_CHANGE#, ONLINE_TIME)

```
select name, phyrds, phywrts, readtim, writetim
from v$filestat f, v$datafile d
where f.file# = d.file#
order by readtim; ( lub order by (phyrds + phywrts) );
```

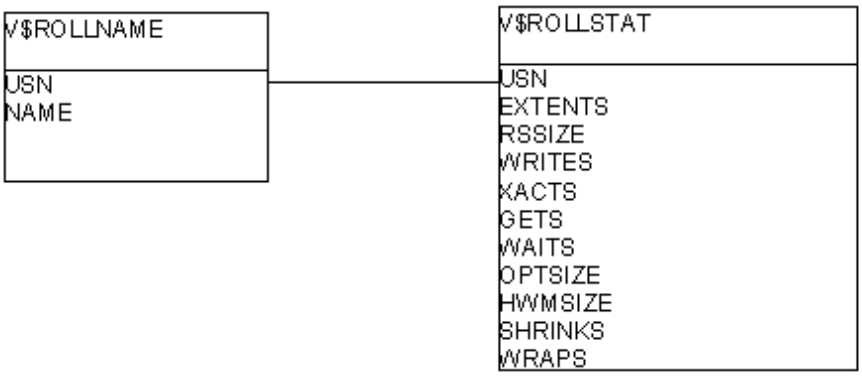
Duże różnice pomiędzy ilościami fizycznych zapisów i odczytów pomiędzy dyskami sugerują rozważenie innego rozłożenia tabel i indeksów. Za pośrednictwem perspektywy v$tablespace można dotrzeć do przestrzeni tabel, a następnie wykorzystując omówione w poprzednich punktach powiązania perspektyw słownika danych, dokonać rewizji rozmieszczenia tabel i indeksów w przestrzeni tabel.

3.2. Perspektywy dostarczające informacji o wykorzystywaniu segmentów wycofywania transakcji

Segmenty wycofywania są wykorzystywane do przechowywania danych zmodyfikowanych przez niezatwierdzoną transakcję w celu przywrócenia poprzednich wartości w przypadku wycofania transakcji. Każda transakcja wykorzystuje jeden segment wycofania. Istotną kwestią jest dostosowanie segmentów wycofywania do stopnia złożoności transakcji - ilości dokonywanych modyfikacji. Można przydzielić transakcji konkretny segment wycofania przy pomocy polecenia :

```
SET TRANSACTION USE ROLLBACK SEGMENT nazwa_RS;
```

Jeżeli przydzielony segment wycofania będzie zbyt mały dla danej transakcji, zostaną przydzielone kolejne rozszerzenia. Po zakończeniu transakcji należy zmniejszyć rozmiar segmentu. Zmniejszenie rozmiaru segmentu może odbywać się automatycznie - po ustawieniu parametru optimal. Wielkość parametru optimal można określić na podstawie obserwacji zachowania się segmentów wycofania podczas pracy.



Rys. 7. Statystyki dotyczące segmentów wycofania

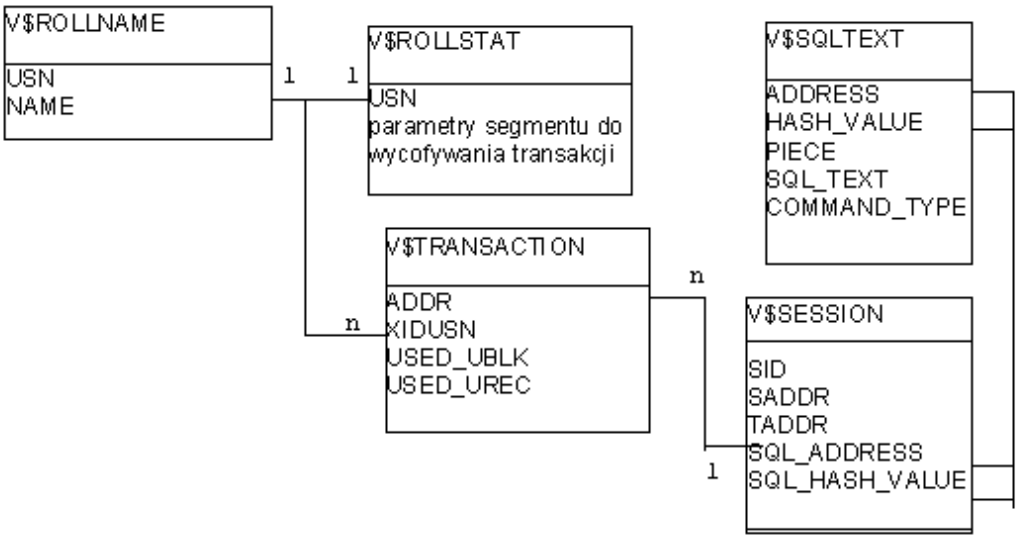
V$ROLLSTAT – parametry rozmiaru (EXTENTS – ilość rozszerzeń, RSSIZE rozmiar w bajtach, WRITES – ilość informacji w bajtach zapisana, OPTSIZE – rozmiar optymalny), parametry zmienności (HWMSIZE, SHRINKS – ile razy rozmiar był zmniejszany, WRAPS – , AVESHRINK, AVEACTIVE, STATUS , CUREXT - bieżące rozszerzenie, CURBLK - bieżący blok).

```
select rn.name, rs.extents, rs.rssize, rs.xacts, rs.waits, rs.gets, rs.optsize,
rs.status from v$rollname rn, v$rollstat rs
where rn.usn = rs.usn ;
```

3.3. Perspektywy umożliwiające powiązanie informacji o wykorzystaniu segmentów do wycofywania transakcji z identyfikacją sesji, transakcji i uruchomionych zapytań SQL

Transakcja jest zbiorem operacji, które są zatwierdzane albo wycofane. Sesja jest tworzona w momencie uruchomienia programu umożliwiającego dostęp do bazy danych np. SQL*Plus. Sesja jest związana z konkretnym użytkownikiem, który w obrębie jednej sesji zatwierdza albo wycofuje kolejne transakcje.

Jeżeli na podstawie zapytania przedstawionego w poprzednim punkcie stwierdzimy istnienie dużych transakcji, które powodują znaczący rozrost segmentów wycofania, konieczne jest zidentyfikowanie sesji, transakcji i zapytań SQL, w celu np. jawnego przydzielenia im innych dostatecznie dużych segmentów wycofania.



Rys. 8. Powiązanie informacji o wykorzystywaniu segmentów wycofania z sesją, transakcją i zapytaniem SQL.

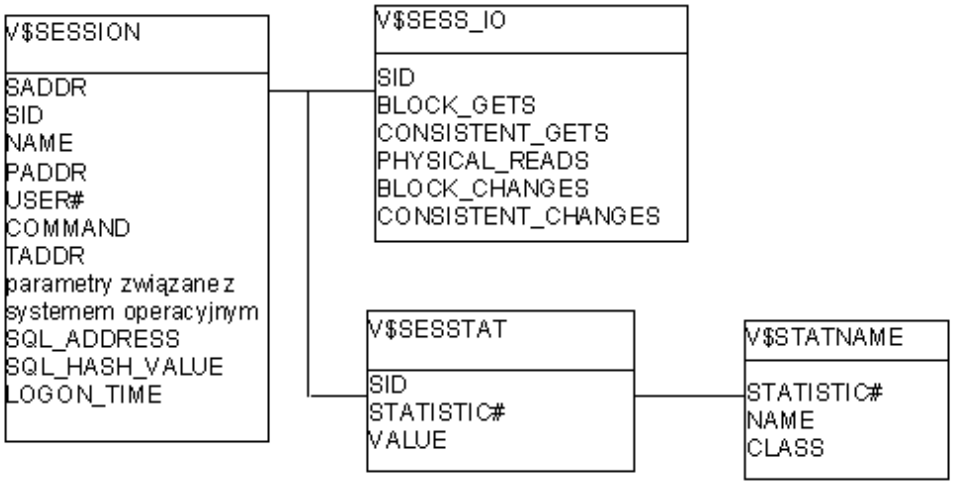
V$TRANSACTION - wiąże informacje o wykorzystaniu segmentu do wycofywania transakcji z sesją użytkownika. Zawiera parametry transakcji m.in. ilość wykorzystanych bloków (USED_UBLK) i rekordów (USED_UREC) w segmencie do wycofywania.

V$SESSION – w niniejszym zapytaniu wiążę transakcję z aktualnie wykonywanym zapytaniem SQL. Szerzej omówiona w kolejnym punkcie

V$SQLTEXT - zawiera aktualnie wykonywane SQL.

```
select a.name, b.xacts, c.sid, c.serial#, c.username, d.sql_text
from v$rollname a, v$rollstat b, v$session c, v$sqltext d, v$transaction e
where   a.usn=b.usn and   b.usn=e.xidusn and c.taddr = e.addr and
c.sql_address=d.address and c.sql_hash_value=d.hash_value
order by a.name, c.sid, d.piece;
```

3.4. Perspektywy dostarczające informacje o sesji.



Rys. 9. Statystyki sesji.

V$SESSION - do najczęściej używanych parametrów należą : SADDR (adres sesji), SID (identyfikator sesji), PADDR (adres procesu), USER# (numer użytkownika), COMMAND (numer komendy), parametry związane z systemem operacyjny(OSUSER, PROCESS, MACHINE, TERMINAL, PROGRAMM), SQL_ADDRESS, SQL_HASH_VALUE (umożliwia identyfikacje aktualnie wykonywanej komendy SQL)

V$SESS_IO - statystyki operacji wejścia / wyjścia dla sesji - SID (identyfikator sesji), BLOCK_GETS, CONSISTENT_GETS, PHYSICAL_READS, BLOCK_CHANGES, CONSISTENT_CHANGES.

V$SESSTAT - statystyki sesji - są różnych klas - dotyczące użytkownika, segmentów do ponawiania operacji, kolejkowania, pamięci, systemu operacyjnego, przetwarzania równoległego, zapytań SQL.

Zapytanie wyświetlające statystyki dotyczące sesji, użytkownika o znanym identyfikatorze na poziomie systemu operacyjnego.

```
Select rs.name, rs.class, ss.*
from v$statname rs, v$sesstat ss, v$session s
where rs.statistic# = ss.statistic#
and s.sid = ss.sid
and s.osuser= 'UnixUser'
order by rs.class;
```

Istotną informacją jest współczynnik odczytów dokonywanych z pamięci operacyjnej w stosunku do wszystkich odczytów (ang. hit ratio).

```
Select (physical_reads/(block_gets+consistent_gets))
from v$sess_io vs, v$session ss
where vs.sid = ss.sid
and ss.username='Uzytkownik';
```

Bibliografia

1. Wrembel R., Jezierski J., Zakrzewicz M.: System zarządzania bazą danych Oracle7 i Oracle8, BUM, 1999, ISBN 83-86969-34-2
2. Austin D.: Poznaj Oracle8, MIKOM, 1999, ISBN 83-7158-153-X
3. Dokumentacja ze szkolenia Oracle8 : Administracja bazą danych
4. Oracle Administrator's guide
5. Oracle Server Reference
6. Niemiec T.: Oracle Performance Tuning Tips & Techniques, Osborne McGraw-Hill, 1999