SELECT - pełna postać:

```
SELECT [DISTINCT | ALL] { *
        | { [schema.]{table | view | snapshot}.*
        | expr [ [AS] c_alias ] }
        [, { [schema.]{table | view | snapshot}.*
        | expr [ [AS] c_alias ] } ] ... }

FROM { (subquery)
        | [schema.]{table | view | snapshot}[@dblink] } [t_alias]
        [, { (subquery)
        | [schema.]{table | view | snapshot}[@dblink] }
[t_alias] ] ...

[WHERE condition ]

[ [START WITH condition] CONNECT BY condition]

[GROUP BY expr [, expr] ...] [HAVING condition]

[{UNION | UNION ALL | INTERSECT | MINUS} SELECT command ]

[ORDER BY {expr | c_alias | position} [ASC | DESC]
        [, {expr | c_alias | position}
        [ASC | DESC]] ...]

[FOR UPDATE [OF [[schema.]{table | view}.]column
        [, [[schema.]{table | view}.]column] ...]
         [NOWAIT] ]
```

Wyrażenia w SQLu:

expr {= | != | ^= | <\> | > | < | >= | <=} expr

(różnice: != *nie równy* dla VAX, UNIX, PC
^= *nie równy* dla IBM
_= *nie równy* dla IBM
<> *nie równy* - wszystkie systemy
)

{ expr [NOT] IN {expr_list | (subquery)}

expr [NOT] BETWEEN expr AND expr

expr IS [NOT] NULL

EXISTS (subquery)

{ (condition) | NOT condition | condition AND condition | condition OR condition }

Funkcje numeryczne, pojedyńcze wiersze[1]

Funkcja	Zwracana wartość
ABS(n)	Część całkowita *n*.
CEIL(n)	Najmniejsza liczba całkowita nie większa od *n.99*
COS(n)	Kosinus *n* (*n* w radianach).
COSH(n)	Kosinus hiperboliczny *n* (*n* w radianach).
EXP(n)	e^n.
FLOOR(n)	Najmniejsza liczba całkowita nie większa od *n*
LN(n)	Logarytm naturalny z *n*, (*n* > 0)
LOG(m,n)	$\log_m(n)$
MOD(m,n)	Reszta z dzielenia *m* przez *n*.
POWER(m,n)	m^n
ROUND(n[,m])	*n* zaokrąglone do *m* miejsc po przecinku (defaultowo *m* = 0).
SIGN(n)	**jeżeli** *n*<0 **to** -1; **jeżeli** *n*=0, **to** 0; **jeżeli** *n*>0, **to** 1.
SIN(n)	Sinus *n* (*n* w radianach)
SINH(n)	Sinus hiperboliczny *n* (*n* w radianach)
SQRT(n)	$\sqrt{n}$; **jeżeli** *n*<0, **to** *NULL*.
TAN(n)	Tangens *n* (*n* w radianach)
TANH(n)	Tangens hiperboliczny *n* (*n* w radianach)
TRUNC(n[,m])	*n* obcięte do *m* miejsc po przecinku (defaultowo *m* = 0).

Funkcje znakowe, pojedyńcze wiersze

Funkcja	Zwracana wartość
CHR(n)	zwraca znak o podanym kodzie
CONCAT($char_1$,$char_2$)	konkatenuje $char_1$ i $char_2$.
INITCAP(char)	zmienia wszystkie pierwsze znaki w słowach na duże litery
LOWER(char)	zmienia wszystkie litery w *char* na małe
LPAD($char_1$,n[,$char_2$])	dostawia z lewej strony łańcucha $char_1$ *n* znaków $char_2$. Defaultowo $char_2$ równy spacji.
LTRIM(char[,set])	usuwa z początku łańcucha *char* wszystkie znaki zgodne ze znakami z *set*. Defaultowo usuwa spacje.
REPLACE(char,search_string [,replacement_string])	w łańcuchu *char* zamień każde wystąpienie *search_string* na *replacement_string*. Jeżeli *replacement_string* nie występuje - usuń każde wystąpienie *search_string*.
RPAD($char_1$,n[,$char_2$])	dostawia z prawej strony łańcucha $char_1$ *n* znaków $char_2$. Defaultowo $char_2$ równy spacji.
RTRIM(char[,set])	usuwa z końca łańcucha *char* wszystkie znaki zgodne ze znakami z *set*. Defaultowo usuwa spacje.
SOUNDEX(char)	zwraca kod 'słowa podobnego' do *char*. Nie działa dla polskich słówek.
SUBSTR(char,m[,n])	zwraca podłańcuch z *char*, zaczynający się od znaku o numerze *m* mający *n* znaków. Jeżeli *n* nie wystąpi - podłańcuch zaczynający się od znaku *m* zawierający wszystkie pozostałe znaki (do końca łańcucha *char*).
SUBSTRB(char,m[,n])	jak ***substr*** ale numery *m* oraz *n* określają[1] bajty, nie znaki (dla niektórych platform sprzętowych to nie jest to samo)
TRANSLATE(char,from,to)	zamień w *char* wszystkie znaki występujące w łańcuhcu *from* na znaki występujące w łańcuchu *to*
UPPER(char)	zmień wszystkie litery w *char* na duże litery
NLS_INITCAP(char [, nls_sort])	jak ***initcap*** ale pozwala zdefiniować '*narodowe´* zasady kapitalizacji wyrazów ('*nls_sort*')

[1] Wymienione zostały funkcje najczęściej używane - pełny zestaw funkcji zależy od konkretnej aplikacji.

NLS_LOWER(char [, nls_sort])	jak ***lower*** ale z możliwością zdefiniowania zasad '*narodowych*' zmiany liter na małe ('*nls_sort*')
NLS_UPPER(char [, nls_sort])	jak ***upper*** ale z możliwością zdefiniowania zasad '*narodowych*' zmiany liter na duże ('*nls_sort*')

Funkcje zwracające wyrażenia znakowe, działające na pojedyńczych wierszach

Funkcja	Zwracana wartość
ASCII(char)	zwraca kod ASCII znaku *char*. W systemach w których znaki mają kody wielobajtowe zwraca kod pierwszego bajtu.
INSTR($char_1$,$char_2$[,n[,m]])	zwraca pozycję m_{tego} (default: pierwszego) wystąpienia znaku $char_2$ w łańcuchu $char_1$. Zaczyna wyszukiwanie od n_{tego} (default: od pierwszego) znaku.
INSTRB(char1,char2[,n[,m]])	jak ***instr*** ale zwraca pozycję *bajtu* a nie znaku
LENGTH(char)	zwraca długość łańcucha *char* liczoną w znakach
LENGTHB(char)	zwraca długość łańcucha *char* liczoną w bajtach

Funkcje daty i czasu[2]

Funkcja	Zwracana wartość
ADD_MONTHS(d,n)	data *d* plus *n* miesięcy
LAST_DAY(d)	data ostatniego dnia miesiąca, do którego należy data *d*
MONTHS_BETWEEN(d,e)	ilość miesięcy o jaką data *e* poprzedza datę *d*
NEW_TIME(d,a,b)	jaka data w strefie *b* odpowiada dacie *d* w strefie *a*. *a* i *b* są znakowymi kodami stref czasowych
NEXT_DAY(d,char)	zwraca datę pierwszego dnia tygodnia o nazwie *char* która jest nie wcześniejsza niż data *d*
SYSDATE	bieżąca data i czas

'Inne' funkcje

Funkcja	Zwracana wartość
DECODE(expr,$search_1$,$return_1$, [$search_2$,$return_2$,]...[default])	**jeżeli** *expr* jest równe jakiejś wartości $search_i$ **to** zwróć odpowiadającą mu wartość $return_i$ **w przeciwnym razie** zwróć *default*
GREATEST(expr[,expr]...)	zwróć wartość maksymalną.
LEAST(expr[,expr]...)	zwróć wartość minimalną.
NVL($expr_1$,$expr_2$)	**jeżeli** $expr_1$ jest równe **null to** zwróæ wartoœæ $expr_2$ **w przeciwnym wypadku** zwróæ $expr_1$
UID	zwróć unikalny numer identyfikujący użytkownika
USER	nazwa bieżącego użytkownika
VSIZE(expr)	ilość bajtów zajmowanych przez wewnętrzną reprezentację *expr*
SYSDATE	bieżąca data i czas systemowy

[2]Na zmiennych typu **data** można wykonywać następujące operacje:

data + liczba	oblicz datę o *liczbę* dni późniejszą[1]
data - liczba	oblicz datę o *liczbę* dni wcześniejszą[1]
data - data	oblicz ilość dni pomiędzy datami
data + liczba/24	dodaj pewną[1] ilość godzin do daty

Zaokrąglanie i 'obcinanie' daty i czasu

Funkcja	Zwracana wartość
ROUND(d[,fmt])	data *d* zaokrąglona według formatu *ftm*
TRUNC(d[,fmt])	data *d* 'obcięta' według formatu *ftm*

fmt:	cc	zamień na wiek
	y	zamień na rok (5)
	yy	zamień na rok (95)
	yyy	zamień na rok (995)
	yyyy	zamień na rok (1995)
	q	zamień na kwartał
	mm	zamień na miesiąc (1,..)
	mon	zamień na miesiąc (jan,...)
	month	zamień na miesiąc (january)
	ww	zamień na najbliższy poniedziałek (== początek tygodnia)
	w	zamień na najbliższy dzień który jest takim samym dniem tygodnia jak pierwszy dzień miesiąca wskazywanego przez datę
	j	zamień na numer dnia w tygodniu
	dd	zamień na numer dnia w miesiącu
	ddd	zamień na numer dnia w roku
	hh	zamień na godzinę
	hh12	zamień na godzinę w formacie am/pm
	hh24	zamień na godzinę w formacie 24 godzinnym
	mi	zamień na minutę

Funkcje konwersji

Funkcja	Zwracana wartość
TO_CHAR(expr[,fmt])	skonwertuj *expr* z wartości typu **number** lub **date** do wartości znakowej według formatu *fmt*.
TO_DATE(char[,fmt])	konweruje wyrażenie typu **char** do typu **date** według formatu *fmt*
TO_NUMBER(char [,fmt])	konwertuje wyrażenie typu **char** do typu **number** wed lug formatu *fmt*

Formaty konwersji, używane w funkcjach **to_char** *i* **to_date**

używane tylko w **TO_CHAR**

format	**znaczenie**
"string"	wypisywany bez zmian
fm	poprzedza **Month** lub **Day**, powoduje, że nazwy miesięcy i dni będą zajmowały tylko tyle znaków, ile występuje w nazwie.**w przeciwnym razie** wszystkie nazwy będą wyrównywane do tej samej długości
TH	dodane do numeru roku, miesiąca, dnia, godziny, minuty lub sekundy powoduje wypisanie przyrostka 'th', kapitalizacja liter - jak w formacie liczby
SP	dodane do liczby powoduje wypisanie jej wartości słownie
SPTH	jak **SP** + **TH**
THSP	jak **TH** + **SP**

N/w znaki będą ignorowane przez **TO_DATE** i wypisywane bez zmian przez **TO_CHAR**:

/	**,**	**-**	**:**
.	**A.M.**	**P.M.**	**a.m.**
p.m.	**AM**	**am**	**PM**
pm	**CC (wiek)**	**SCC (wiek, - dla pne)**	**B.C.**
BC	**AD**	**A.D.**	**bc**
b.c.	**ad**	**a.d.**	

***wspólne dla* TO_CHAR *i* TO_DATE**

format	**znaczenie**
MM	numer miesiąca, cyframi arabskimi
RM	numer miesiąca, cyframi rzymskimi
MON	trzyliterowy skrót nazwy miesiąca, dużymi literami
Mon	j/w, ale tylko pierwsza litera duża
mon	jw, wszystkie litery małe
MONTH	pełna nazwa miesiąca, dużymi literami
Month	j/w, tylko pierwsza litera duża
month	j/w, wszystkie litery małe
DDD	numer dnia w roku
DD	numer dnia w miesiącu
D	numer dnia w tygodniu
DY	trójliterowy skrót nazwy dnia tygodnia, dużymi literami
Dy	j/w, tylko pierwsza litera duża
dy	j/w, wszystkie litery małe
DAY	pełna nazwa dnia tygodnia, wszystkie litery duże
Day	j/w, tylko pierwsza litera duża
day	j/w, wszystkie litery małe
YYYY	czterocyfrowy numer roku
SYYYY	j/w, dla lat pne - jako liczba ujemna
IYYY	j/w zgodny z normą ISO

format	**znaczenie**
YYY	trzy ostatnie cyfry numeru roku
IYY	j/w, ISO format
YY	dwie ostatnie cyfry roku
IY	j/w, ISO format
Y	ostatnia cyfra numeru roku
I	j/w, ISO format
RR	ostatnie dwie cyfry numeru roku, być może z innego wieku
YEAR	pełna, słowna nazwa roku, wszystkie litery duże
Year	j/w, tylko pierwsza litera duża
year	j/w, wszystkie litery małe
Q	numer kwartału w roku
WW	numer tygodnia w roku
W	numer tygodnia w miesiącu
IW	numer tygodnia w roku, ISO format
J	data Juliańska (od 31/12 4713 pne)
HH	numer godziny w dniu, 1 - 12

HH12	j/w
HH24	numer godziny, format 24 godzinny
MI	minuta w godzinie
SS	sekundy minuty
SSSSS	sekundy od północy

formaty używane do wypisywania liczb
***używane w* TO_CHAR, TO_NUMBER *i* COLUMN**

format	znaczenie
9	cyfra, ilość 9 określa szerokość pola
0	wyświetlaj wiodące zera
$	pozycja wyświetlania znaku $
.	pozycja kropki dzisiętnej
,	pozycja przecinka, oddzielającego trzycyfrowe grupy cyfr
MI	wypisuj minus **po prawej stronie** liczby ujemnej
PR	liczbe ujemne wyświetlaj w nawiasach
EEEE	wyświetlaj liczby w postaci wykładniczej
V	wyświetlaj liczbę pomnożoną przez 10^n, *n* jest ilością cyfr po **V**
B	wyświetlaj końcowe zera jako zera a nie spacje

Funkcje grupowe

Funkcja	Zwracana wartość
AVG([DISTINCT|ALL]n)	wartość średnia *n*, wartości **null**s[1] ignorowane
COUNT([ALL]*)	ilość wierszy zwracana przez zapytanie lub podzapytanie, **'*'** oznacza '*policz ilość wierszy*'
COUNT([DISTINCT|ALL]expr})	ilość wierszy dla których *expr*ma wartoœæ ró¿n[1] od **null**
MAX([DISTINCT|ALL]expr)	maksymalna wartość *expr*
MIN([DISTINCT|ALL]expr)	minimalna wartość *expr*
STDDEV([DISTINCT|ALL]n)	odchylenie standardowe *n*, wartości **null** są ignorowane
SUM([DISTINCT|ALL]n)	suma wartości *n*
VARIANCE([DISTINCT|ALL]n)	wariancja *n*, wartości **null** są ignorowane

Wszystkie funkcje grupowe - oprócz **count(*)** - ignorują wartości **null**. Użycie klauzuli **group by** powoduje wykonywanie obliczeń grupowych na wydzielonych grupach rekordów. Jeżeli w klauzuli **group by** użyjemy nazw kilku kolumn otrzymamy 'grupy wewnątrz grup'. Kolejność grupowania - od lewej do prawej. **Na liście wyboru komendy *select* używającego funkcji grupowych wolno umieszczać tylko te kolumny, które występują w klauzuli *group by*** - oczywiście oprócz samych funkcji grupowych.

Można dodatkowo użyć klauzuli

having *wyrażenie logiczne*

Spowoduje to wybranie tylko tych grup, które spełniają warunek z **having**. Kolejność występowania klauzul w rozkazie **select** jest ściśle określona:

```
select      list kolumn
from        lista tablic
where       warunek dla wierszy
group by    kolumny
having      warunek dla grup
order by    kolumny;
```

Tworzenie tabel

Tworzenie tablic:

- **CREATE TABLE**

```
CREATE TABLE table
( { column [datatype] [column_constraint] ... | table_constraint}  ...
```

można stworzyć tabelę poprzez podzapytanie

```
CREATE TABLE tablica
[ ( nazwa kolumny [ NULL / NOT NULL ] , ... ) ]
AS SELECT parametry select;
```

jeżeli nie podamy nazw kolumn - te które są wymienione w selekcie zostaną utworzone w tabeli. Jeżeli podamy nazwy kolumn - select zostanie użyty do wstawienia wartości do nowo utworzonej tabeli.

zasady ogólne:

- tworzenie nowych tabel i modyfikacja struktury istniejących tabel może się odbywać w czasie pracy (''*on-line*'')
- postać nazwy:
  - do 30 znaków, musi się zaczynać od litery, może zawierać liery, cyfry, _, $, # ($ i # są 'nie zalecane')
  - musi być unikalna pośród nazw dostępnych dla danego użytkownika
  - jeżeli jest taka jak słowo kluczowe PL/SQL (nie zalecane) to musi być ujmowana w cudzysłowy
- jeżeli w instrukcji ***create table*** nazwę tabeli napiszemy w cudzysłowach (***"emp"***) to duże i małe litery w nazwach kolumn i nazwie tablicy nie są utożsamiane

Klauzule CONSTRAINTS

W instrukcjach *create table* i *alter table* można zdefiniować klauzule tworzące ograniczenia na dane wprowadzane do tablicy. Pozwala to na zdefiniowanie warunków - zarówno w obrębie jednej tablicy jak i wielu tablic (np. nie można zmienić numeru departamentu w którym pracuje pracownik - tablica EMP - o ile numer tego departamentu nie występuje w tablicy DEPT). *Oracle* zapewnia przestrzeganie tych ograniczeń podczas pracy z tabelemi (generuje *exceptions*).

Rodzaje ograniczeń:

- wartośc różna od NULL - klauzula **not null**

- występowanie tylko unikalnych wartości w kolumnie lub kolumnach - klauzula **unique**

  ... empno number(4) unique ...
  ... unique (empno, deptno) ...

- definiowanie kolumny lub kolumn jako kolumny ideksującej tablicę - klauzula **primary key**

  create table assignments
  (project number(4), employee number(4),
  primary key (project, employee));

  create table dept
  (deptno number primary key,
  ...);

  Tylko jedna klauzula ***primary*** może wystąpić w definicji tabeli. Żadna kolumna z atrybutem ***primary*** nie może mieć atrybutu ***null***.

- określenie związku '*klucz obcy*' - klauzula **foreign key**, może się odnosić zarówno do innej jak i do tej samej tabeli

  ... constraint emp_dept foreign key (mgr) references emp (empno) ...

  '*mgr*' jest '*kluczem obcym*' odnoszącym się do kolumny EMPNO tabeli EMP. '*emp_dept*' jest nazwą warunku *constraint*

- wymaganie aby wartości kloumny (kolumn) spełniały określony warunek - klauzula **check**

  ... hiredate date check (hiredate < sysdate) ...

możliwe '*constraints*':

constraint *nazwa*	określa nazwę więzów. Jeżeli pominiemy nazwę - Oracle utworzy własną (*SYS_Cn*). Nazwa więzów pojawia się w komunikatach o błędach
null	defaultowo *not null*, określa czy kolumny muszą mieć nadane
not null	wartości
unique	wymaganie aby wszystkie wartości w kolumnie (we wszystkich rekordach) były wzajemnie różne. Kolumna powinna mieć atrybut *not null* i nie być *primary key*
foreign key (*kol*, ...)	ta kolumna będzie łączyć table (rózne lub rekordy tej samej
references	tabeli) poprzez wartości wskazanej kolumny. Kolumna ta musi
***tabela* (*kol*, ...)**	być typu *primary key* we wskazanej tablicy. '*references*' pozwoli na wstawienie wartości tylko wtedy, kedy występuje ona we wskazanej kolumnie wskazanej tabeli
check	definiuje wrunek, który musi być spełniony w każdym wierszu tabeli. Może się odnosić tylko do wierszy tabeli w której jest definiowany

Modyfikacja danych w tabeli:

- **INSERT**

```
INSERT INTO [schema.]{table | view}[@dblink]
        [ (column, ...) ]
        VALUES (expr, ...);

INSERT INTO [schema.]{table | view}[@dblink]
        [ (column, ...) ]
         subquery;
```

dodaje (append) nowy wiersz do tabeli

- można opuścić nazwy kolumn - o ile wstawiamy dane do wszystkich kolumn ale nie jest to zalecane - struktura tabeli może się zmienić (zostaną dodane nowe kolumny) i pojawią się błędy w aplikacjach
- wartości znakowe i daty muszą być ujęte w apostrofy. Jeżeli w dacie nie wstawiamy jawnie godziny to data jest wpisywana z czasem ustawionym na północ (00:00:00)
- użycie podzapytania umożliwia wstawienie wielu wierszy na raz. Nie wsytępuje wtedy słowo kluczowe *VALUES*.

- **DELETE**

```
DELETE [FROM] [schema.]{table | view}[@dblink] [alias]
        [WHERE condition];
```

usuwa z tabeli wiersze wybrane klauzulą *where*. **Jeżeli nie użyjemy *where* zostaną usunięte wszyskie wiersze tabeli!!!**

- **UPDATE**

```
UPDATE [schema.]{table | view}[@dblink] [alias]
        SET column = expr,
              column = (subquery), ...
        [WHERE condition]
```

zmienia wartości kolumn w wierszach wybranych klauzulą *where*. **Jeżeli nie użyjemy *where* zmodyfikowane zostaną wszystkie wiersze tabeli!** Można dowolnie mieszać użycie wyrażeń i podzapytań. Użycie konstrukcji

... (*kolumna*, *kolumna*, ...) = (*subquery*, *subquery*, ...) ...

pozwala nadać nazwy wielu kolumnom za pomocą jednego podzapytania

Modyfikacja struktury danych:

- **ALTER**

ALTER TABLE [schema.]table

[ADD (column [datatype] [DEFAULT expr] [column_constraint])
table_constraint
]

[MODIFY (column [datatype] [DEFAULT expr])
]

[DROP COLUMN column] ... [DROP INDEX index] ... [DROP TRIGGER trigger] ...
[DROP SYNONYM synonim] ... [DROP VIEW widok] ...

- ***add*** służy do dodawania nowych kolumn lub więzów (*constraints*)
- ***modify*** służy do zmodyfikowania definicji istniejącej kolumny

alter table emp
modify (ename char(25));

ograniczenia:
- nie można zmieniać atrybutu na *not null* jeżeli w kolumnie występują wiersze z wartością *null*
- nie można dodać nowej kolumny typu *not null*. Trzeba dodać kolumnę jako *null*, a następnie zmienić jej typ na *not null*.
- nie można zmniejszyć rozmiaru kolumny - chyba że jest pusta
- nie można zmienić nazwy kolumny

- **DROP**

DROP TABLE [schema.]table

[CASCADE CONSTRAINTS] ;

usuwa wszystkie dane i wartości indeksów. **Operacja nieodwracalna!!!**. Klauzula '*cascade constraints*' powoduje usunięcie wszystkich więzów związanych z usuwaną tablicą.

Usuwane są dane i definicja tabeli.

- **COMMENT ON TABLE *nazwa* IS *'tekst'*;**

definiuje komentarz do tabeli

- **COMMENT ON COLUMN *nazwa_tabeli.nazwa_kolumny* IS *'tekst'*;**

definiuje komentarz do kolumny

usunięcie - przez ponowne zdefiniowanie z pustym tekstem(**''**). Wartości komentarzy znajdują się w tabelach ALL_COL_COMMANTS i USER_COL_COMMENTS, kolumna COMMENTS.

- **RENAME *stara_nazwa* TO *nowa_nazwa*;**

zmienia nazwę tabeli. **Trzeba samemu zmienić odwołania do tabeli we wszystkich operacjach!!!**

Perspektywy

Tworzenie:

create view ***nazwa_perspektywy***
[(*kolumna*, ...)]
as select ...
[with check option [constraint *ograniczenie*]]

- pozwala tworzyć 'pseudo-tablice', możliwe składanie elementów różnych tablic
- używana dokładnie tak samo jak zwykła tablica
- składowymi perspektywy mogą być kolumny i wyrażenia stworzone z kolumn tablic lub innych perspektyw
- kolumny tworzonej perspektywy mogą być kolumnami tablic lub wyrażeniami
- nie można używać ***order by*** przy tworzeniu perspektywy, można używać przy selekcji danych z perspektywy
- w stosunku do perspektywy można normalnie używać rozkazów DML - ***INSERT, UPDATE***
- jeżeli perspektywę tworzymy bez ***'with check option'*** - nie działa sprawdzanie poprawności wpisywanych danych. Powoduje to również wpisywanie błędnych danych do tabel, z których powstała perspektywa. O ile dopisane wartości nie będą spełniać warunków ograniczających perspektywę będą dołączone do tablic bazowych ale nie będą wyświetlane w perspektywie.

Podczas tworzenia perspektywy (***create view***) ***select*** występujący po ***as*** nie jest wykonywany - jest tylko zapisany w słowniku danych. Prawa dostępu do tablic też nie są sprawdzane w momencie tworzenia perspektywy - są sprawdzane dopiero w momencie pobierania lub modyfikowania danych za pomocą perspektywy. Wynika stąd, że twórca perspektywy *nie musi mieć zagwarantowanych praw dostępu do tabel, z których tworzy perspektywę!!!* Jest to potencjalna '***dziura***' w systemie zabezpieczeń Oracla.

Dane o perspektywach są pamiętane w tabeli USER_VIEWS.

Ograniczenia dotyczące modyfikowania danych za pomocą perspektyw:

- **DELETE nie jest dozwolone, jeżeli perspektywa zawiera:**
  - warunek łączący (tabele)
  - funkcje grupowe
  - klauzulę GROUP BY
  - kwalifikator ***distinct***
  - pseudokolumnę ***rownum*** (rownum zwraca numer (kolejność) w jakim wiersz był wybierany podczas selekcji danych z tabeli)
  - skorelowane zapytania

- **UPDATE nie jest dozwolone, jeżeli perspektywa zawiera:**
  - jakąkolwiek z opcji wymienionych dla DELETE
  - kolumnę tworzoną za pomocą wyrażenia

- **INSERT nie jest dozwolone jeżeli perspektywa zawiera:**
  - jakąkolwiek z opcji wymioenionych dla UPDATE
  - kolumna z atrybutem NOT NULL, znajdująca się w tablicy z jakiej tworzona jest perspektywa nie jest składnikiem perspektywy

usuwanie perspektyw

komenda

drop view nazwa_perspektywy;

usuwa perspektywę. Perspektywę może usunąć tylko jej twórca. Usunięcie perspektywy może spowodować, że inne perspektywy, zbudowane na bazie usuwanej perspektywy staną się błędne

można użyć rozkazu

create or replace view ***nazwa***
as
select **...**

żeby usunąć i ponownie założyć perspektywę bez zmiany nadanych wcześniej praw dostępu do perspektywy.

Wybrane dane o użytkowniku

View	Zawartość
USER_CATALOG	tablice, widoki, synonimy i sekwencje należące do użytkownika
USER_COL_PRIVS	prawa dostępu do kolumn których użytkownik jest właścicielem, 'gwarantującym' lub 'posiadającym dostęp
USER_COL_PRIVS_MADE	wszystkie prawa dostępu do obiektów należących do użytkownika
USER_COL_PRIVS_RECD	prawa dostępu do kolumn dla których użytkownik jest 'gwarantującym'
USER_CONSTRAINTS	ograniczenia zdefiniowane dla tabel użytkownika-
USER_INDEXES	definicje indeksów użytkownika
USER_OBJECTS	obiekty należące do użytkownika
USER_SEQUENCES	sekwencje należące do użytkownika
USER_SNAPSHOTS	'snapshoty' które użytkownik może używać
USER_SOURCE	teksty opisujące obiekty użytkownika pamiętane w bazie
USER_SYNONYMS	synonimy zdefiniowane przez użytkownika
USER_SYS_PRIVS	przywileje systemowe zagwarantowane użytkownikowi
USER_TABLES	tablice należące do użytkownika
USER_TAB_COMMENTS	teksty komentarzy opisujących tabele i widoki użytkownika
USER_TRIGGERS	opisy triggerów użytkownika
USER_USERS	informacja o bieżącym użytkowniku
USER_VIEWS	definicje widoków należących do użytkownika

Wybrane dane o wszystkich:

View	Zawartość
ALL_CATALOG	wszystkie tabele, widoki, synonimy i sekwencje dostępne dla użytkownika
ALL_COL_PRIVS_MADE	prawa dostępu do kolumn dla których użytkownik lub PUBLIC jest gwarantującym dostęp
ALL_CONSTRAINTS	definicje ograniczeń dla tabel dostępnych dla użytkownika
ALL_ERRORS	informacje o błędach - dla wszystkich obiektów dostępnych użytkownikowi
ALL_INDEXES	definicje indeksów tabel dostępnych dla użytkownika
ALL_OBJECTS	obiekty dostępne dla użytkownika
ALL_SEQUENCES	definicje sekwencji distępnych dla użytkownika
ALL_SNAPSHOTS	'snapshoty' dostępne dla użytkownika
ALL_SOURCE	teksty definiujące obiekty dostępne dla użytkownika
ALL_SYNONYMS	synonimy dostępne dla użytkownika
ALL_TABLES	opisy tabel dostępnych dla użytkownika
ALL_TAB_COLUMNS	opisy kolumn z tabel i widoków dostępnych dla użytkownika
ALL_TAB_COMMENTS	teksty komentarzy dla tabel i widoków dostępnych dla użytkownika
ALL_TRIGGERS	triggery dostępne dla użytkownika
ALL_USERS	informacje o użytkownikach

Dodatek - definicje struktur tabel używanych w ćwiczeniach:

tabela EMP			
pole	***NULL?***	***typ***	***komentarz***
EMPNO	NOT NULL	number(4)	numer pracownika
ENAME		varchar2(10)	nazwisko pracownika
JOB		varchar2(9)	nazwa stanowiska
MGR		number(4)	numer pracownika, który jest kierownikiem bieżącego pracownika
HIREDATE		date	data zatrudnienia
SAL		number(7,2)	zarobki
COMM		number(7,2)	prowizja
DEPTNO		number(2)	numer departamentu

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

tabela DEPT - informacja o departamentach			
pole	***NULL?***	***typ***	***komentarz***
DEPTNO	NOT NULL	number(2)	numer departamentu
DNAME		varchar2(14)	nazwa departamentu
LOC		varcher2(13)	lokacja - nazwa miasta w którym znajduje siê departament

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

tabela SALGRADE			
pole	***NULL?***	***typ***	***komentarz***
GRADE		number	kod zaszeregowania
LOSAL		number	płaca minimalna
HISAL		number	płaca maksymalna

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999