PL/SQL - procedurálny jazyk

Dátové typy

ZNAKOVÉ DÁTOVÉ TYPY --- CHAR, NCHAR, VARCHAR2, NVARCHAR2 + LONG

CHAR(dĺžka) [BYTE | CHAR] Parameter dĺžka je z intervalu 1-2000. Slúži na reťazec pevnej dĺžky.
NCHAR(dĺžka) Parameter dĺžka je z intervalu 1-2000. Slúži na reťazec pevnej dĺžky vo vybranej národnej znakovej sade.
VARCHAR2(dĺžka) [BYTE | CHAR] Parameter dĺžka je z intervalu 1-4000. Slúži na reťazec premenlivej dĺžky.
NVARCHAR2(dĺžka) Parameter dĺžka je z intervalu 1-4000. Slúži na reťazec premenlivej dĺžky vo vybranej národnej znakovej sade.
LONG Slúži na reťazec premenlivej dĺžky do veľkosti až 2GB.

ČÍSELNÝ DÁTOVÝ TYP

Platforma ORACLE ma zabudovaný len tento jediný číselný dátový typ.

NUMBER(n_cislic, n_cislic_za_des_ciarkou)
Príklad: Uloženie cisla 1234567.89 pre rôzne definovane dátové typy NUMBER.
NUMBER
1234567.89 - číselný dátový typ s pohyblivou čiarkou.
NUMBER(9) - 1234568 - zaokrúhli posledne miesto
NUMBER(9,2) - 1234567.89 - znamená, že všetkých čísiel môže byt 9 a z toho sú 2 za des. čiarkou
NUMBER(9,1) - 1234567.9
NUMBER(6) - ????????? - číslo sa nezapíše. Nastane chyba. Nedostatočný rozsah.
NUMBER(7,-2) 1234600 - číslo sa zaokrúhli na stovky
NUMBER(7,2) ????????? - číslo sa nezapíše. Nastane chyba. Nedostatočný rozsah.
NUMBER(4,5) Desatinne číslo 0.000127 sa uloží ako 0.00013

DÁTOVÉ TYPY PRE KALENDÁRNE A ČASOVÉ HODNOTY

DATE

Na riešenie problémov ohľadne rôznych zvyklostí zadávania dátumu slúži funkcia **TO_DATE**
TO_DATE('25.07.2002','dd.mm.yyyy')
TO_DATE('25.07.2002 13:30:22','dd.mm.yyyy, HH24:MI:SS')
TO_DATE('98-DEC-07:17:30','YY-MON-DD:HH24:MI')
Na uloženie aktuálneho dátumu a času sa používa funkcia SYSDATE
Tiež sa dá použiť v tvare:
SYSDATE+1 znamená zajtra
SYSDATE-1 znamená včera
SYSDATE+3 znamená o 3 dni
Tiež sa dá pracovať zo zlomkami. Deň ma 24 hodín alebo tiež 1440 minút:
SYSDATE+(1/12) znamená za 2 hodiny
SYSDATE+(10/1440) znamená za 10 minút

TIMESTAMP [presnosť sekúnd]
V podstate to isté ako DATE ale presnosť sekundy je veľký. Štandartne je nastavený na 6 miest(milióntina sekundy). Používa sa s funkciou TO_TIMESTAMP.

INTERVAL YEAR [(rozsah_rokov)] **TO MONTH**
Vyjadrenie väčšieho časového intervalu. Parameter rozsah_rokov slúži k nastaveniu poctu číslic čísla udávajúceho počet rokov. Štandartne je nastavený na hodnotu 2.

INTERVAL DAY [(rozsah_dni)] **TO SECOND** [presnost_sekund]
Vyjadrenie kratšieho a presnejšieho intervalu. Parameter rozsah_dni slúži k nastaveniu poctu číslic čísla udávajúcich počet dni. Štandartne je nastavený na hodnotu 2. Parameter presnost_sekund slúži k určeniu poctu platných číslic sekúnd. Štandartne je nastavený na hodnotu 2.

TYPY PRE OBJEMNE DATA.

Na priame uloženie do databázy sa používajú typy:
BLOB - rozsiahly binárny objekt - max. veľkosť 4GB
CLOB - veľký binárny objekt - max. veľkosť 4GB
NCLOB - rozsiahly binárny objekt, obsahuje znaky UNICODE - max. veľkosť 4GB

Pre externe uloženie objemných dát sa používa dátový typ
BFILE - max. veľkosť 4GB

Vyber dát z tabuliek

Definovanie Alias

3 možnosti - buď do úvodzoviek, s použitím AS, bez použitia AS
SELECT cpartnerid AS "Číslo partnera", datecreated AS dátum, ctypeid typ from odbytcdod_lis

Zreťazenie ||

select cpartnerid || ctypeid "Číslo partnera a typ", datecreated as dátum from odbytcdod_lis
ak tam chceme dať nejakú pomlčku tak: ||'-'||
select cpartnerid ||'-'|| ctypeid "Číslo partnera a typ", datecreated as dátum from odbytcdod_lis

Omedzenie duplicity(DISTINCT)

Podmienka WHERE

PRE CISLA
= WHERE ("MZDA" = ‘3000‘)
<> WHERE ("MZDA" <> ‘3000‘)
> WHERE ("MZDA" > ‘3000‘)
< WHERE ("MZDA" < ‘3000‘)
>= WHERE ("MZDA" >= ‘3000‘)
<= WHERE ("MZDA" <= ‘3000‘)
BETWEEN WHERE ("MZDA" BETWEEN ‘3000‘ AND ‘4000‘)
NOT BETWEEN WHERE ("MZDA" NOT BETWEEN ‘3000‘ AND ‘4000‘)
PRE STRING
= WHERE ("MENO" = ‘JANKO‘)
<> WHERE ("MENO" <> ‘JANKO‘)
obsahuje WHERE ("MENO" LIKE ‘%JANKO%‘)
neobsahuje WHERE (NOT "MENO" LIKE ‘%JANKO%‘)
začína na WHERE ("MENO" LIKE ‘JANKO%‘)
nezačína na WHERE (NOT "MENO" LIKE ‘JANKO%‘)
končí na WHERE ("MENO" LIKE ‘%JANKO‘)
nekončí na WHERE (NOT "MENO" LIKE ‘%JANKO‘)

VYBER Z INTERVALU --- BETWEEN .. END
SELECT * from odbytcdod_lis WHERE cpartnerid BETWEEN 1 and 3
SELECT * from odbytcdod_lis WHERE datecreated BETWEEN to_date('29.06.2002','dd.mm.yyyy') and to_date('30.06.2002','dd.mm.yyyy')
SELECT * from odbytcdod_lis WHERE datecreated < to_date('30.06.2002','dd.mm.yyyy')

VYBER Z MNOZINY --- IN
Pre čísla cpartnerid z množiny 1 az 3
SELECT * from odbytcdod_lis WHERE cpartnerid IN (1,3)
Pre stringy
SELECT * from odbytckarta WHERE ckname IN('Farba biela','Farba modra')
Inak sa to dalo tiež
SELECT * from odbytckarta WHERE ckname = 'Farba biela' OR ckname ='Farba modra'

VYBER PRAZDNYCH HODNOT --- IS NULL
SELECT unitid,ckname from odbytckarta WHERE ckname IS NULL
SELECT unitid,ckname from odbytckarta WHERE ckname IS NOT NULL

VYBER PODLA PRIBLIZNEHO OBSAHU --- % a _
SELECT * from odbytckarta WHERE ckname LIKE '%Farba%'
SELECT * from odbytckarta WHERE ckname LIKE '%Fa_ba%'

TRIEDENIE --- ORDER BY
SELECT * from odbytckarta ORDER BY unitid, ckname DESC
SELECT * from odbytcdod_lis ORDER BY datecreated
SELECT * from odbytcdod_lis ORDER BY datecreated ASC, cpartnerid DESC

SPOJOVANIE TABULIEK --- JOIN

zjednodušene
- SELECT DATECREATED, CTYPEID, PARTNERNAME, CITY FROM ODBYTCDOD_LIS, ODBYTCPARTNER WHERE CPARTNERID = ODBYTCPARTNER.ID
zložitejšie
- SELECT ODBYTCDOD_LIS.DATECREATED, ODBYTCDOD_LIS.CTYPEID, ODBYTCPARTNER.PARTNERNAME, ODBYTCPARTNER.CITY FROM ODBYTCDOD_LIS, ODBYTCPARTNER WHERE ODBYTCDOD_LIS.CPARTNERID = ODBYTCPARTNER.ID
S použitím alias mien tabuliek
- SELECT dl.DATECREATED, dl.CTYPEID, pa.PARTNERNAME, pa.CITY FROM ODBYTCDOD_LIS dl, ODBYTCPARTNER pa WHERE dl.CPARTNERID = pa.ID
Tiež sa môže kombinovať aj s ďalšou WHERE podmienkou
- SELECT dl.DATECREATED, dl.CTYPEID, pa.PARTNERNAME, pa.CITY FROM ODBYTCDOD_LIS dl, ODBYTCPARTNER pa WHERE dl.CPARTNERID = pa.ID AND pa.CITY = 'Bardejov'
Platia tu aj ďalšie zásady ktoré boli popísane v časťi PODMIENKA WHERE

VONKAJSIE SPOJENIE TABULIEK --- LEFT JOIN (+)
SELECT dl.ID, pa.PARTNERNAME, vy.CKARTAID, vy.QUANTITY FROM ODBYTCDOD_LIS dl, ODBYTCPARTNER pa, ODBYTCVYDAJ vy WHERE dl.CPARTNERID = pa.ID AND dl.ID = vy.CDOD_LISID(+)
Vypíše :
ID PARTNERNAME CKARTAID QUANTITY
-- ---------- --------- ---------
2 Partner2 8 4
2 Partner2 1 2
2 Partner2 8 2
4 Partner3 1 1
4 Partner3 7 3
4 Partner3 2 2
6 Partner3
Ale bez (+) vypíše len
ID PARTNERNAME CKARTAID QUANTITY
-- ---------- --------- ---------
4 Partner3 1 1
4 Partner3 2 2
2 Partner2 8 4
2 Partner2 1 2
2 Partner2 8 2
4 Partner3 7 3

SPOJENIE TABULKY SAMO SO SEBOU
Select spojí položky systémom každá s každou
SELECT ka1.CKNAME, ka2.CKNAME FROM ODBYTCKARTA ka1, ODBYTCKARTA ka2 WHERE ka1.ID < ka2.ID

ZOSKUPOVANIE DAT --- GROUP BY a HAVING

SELECT CKARTAID, SUM(QUANTITY) FROM odbytcvydaj GROUP BY CKARTAID
CKARTAID SUM(QUANTITY)
-------- -------------
SELECT CKARTAID, QUALITY, SUM(QUANTITY) FROM odbytcvydaj GROUP BY CKARTAID, QUALITY
CKARTAID QUALITY SUM(QUANTITY)
-------- --------- -------------

POZN. WHERE slúži na obmedzenie záznamov, ktoré sa potom zoskupili pomocou klauzule GROUP BY. HAVING slúži na obmedzenie už zoskupených dat.
SELECT CKARTAID, QUALITY, SUM(QUANTITY) FROM odbytcvydaj GROUP BY CKARTAID, QUALITY HAVING SUM(QUANTITY)>3
CKARTAID QUALITY SUM(QUANTITY)
-------- --------- -------------

POZN. Samozrejme sa môže kombinovať WHERE s HAVING.
SELECT CKARTAID, QUALITY, SUM(QUANTITY) FROM odbytcvydaj WHERE CKARTAID <> 8 GROUP BY CKARTAID, QUALITY HAVING SUM(QUANTITY)>3
CKARTAID QUALITY SUM(QUANTITY)
-------- --------- -------------

VNORENÉ SELECTY

JEDNORIADKOVÉ VNORENÉ SELECTY
Vnorený SELECT musí vracať len 1 záznam

Výsledkom tohoto selectu je 3:
SELECT QUANTITY
FROM odbytcvydaj
WHERE ID=12

Ak chceme všetky riadky s QUANTITY > ako 3:
SELECT * FROM odbytcvydaj WHERE QUANTITY > 3

Ak chceme všetky riadky s QUANTITY > ako výsledok prvého selectu:
SELECT * FROM odbytcvydaj
WHERE QUANTITY > (SELECT QUANTITY FROM odbytcvydaj WHERE ID=12)

Ak chceme vypísať všetky výdaje ktoré sú väčšie ako priemer QUANTITY*SALEPRICE:
SELECT * FROM odbytcvydaj WHERE QUANTITY*SALEPRICE > (SELECT AVG(QUANTITY*SALEPRICE) FROM odbytcvydaj)
Vnorený SELECT sa dá použiť za klauzulami FROM, WHERE, HAVING.

VIACRIADKOVE VNORENE SELECTY
Vnorený SELECT môže vracať viac záznamov ale tam sa používajú viac riadkové operátory porovnávania. ALL, ANY a IN
SELECT * FROM odbytcvydaj
WHERE QUANTITY < ALL (SELECT AVG(QUANTITY) FROM odbytcvydaj
GROUP BY CKARTAID)

VIACSTĹPCOVÉ VNORENÉ SELECTY
Tieto selecty umožňujú porovnanie stĺpcov vnorených selectov so stĺpcami vybranými hlavným selectom.
Zobrazenie dát o každom zamestnancovi ktorého mzda a provízia sú zhodne so mzdou a províziou zamestnanca v oddelení 30.
SELECT MENO, ODDELENIE, MZDA, PROVIZIA FROM zamestnanci WHERE (MZDA, NVL(PROVIZIA, -1)) IN (SELECT mzda, NVL (PROVIZIA, -1) FROM zamestnanci WHERE ODDELENIE = 30)
POZN. Funkcia NVL prinúti agregačnú funkciu, aby brala do úvahy tiež hodnoty NULL. Ak je hodnota NULL vráti funkcia hodnotu druhého parametra.

Ďalší príklad: Potrebujeme vypísať všetky výdaje z tabuľky odbytcvydaj kde karta použitá vo výdaji ma UNITID = 1
Napr.
SELECT * FROM odbytcvydaj, odbytckarta WHERE CKARTAID = odbytckarta.ID AND UNITID = 1
Alebo vnoreným selectom:
SELECT * FROM odbytcvydaj WHERE CKARTAID IN (SELECT ID FROM odbytckarta WHERE UNITID = 1)
Alebo s použitím EXISTS (tento spôsob je najrýchlejší)
SELECT * FROM odbytcvydaj WHERE EXISTS (SELECT 'X' FROM odbytckarta WHERE CKARTAID = odbytckarta.ID AND UNITID = 1)

Úprava dát v tabuľkach

PRIDANIE ZAZNAMOV --- INSERT INTO

INSERT INTO nazov_tabulky [(stlpec1 [,stlpec2 ..])] VALUES (hodnota_polozky1 [,hodnota_polozky2 ..])
INSERT INTO odbytcvydaj (id,ckartaId, Quantity, PurchasePrice, SalePrice, dateinserted, quality, cdod_lisId) values (idcvydaj.NEXTVAL, 13, 3, 28, 34, TO_DATE(sysdate,'dd.mm.yyyy'), 1, 1)

Ak sa zadajú do VALUES hodnoty v poradí ako sú v databáze tak sa vypisovať položky nemusia:
INSERT INTO odbytckarta values (1, idckarta.NEXTVAL, 'Snurky 80 cm cervene')

Dá sa pridať naraz jedným príkazom aj viac položiek do tabuľky. Napr. Ak chceme vytvoriť nejakú pomocnú tabuľku už z existujúcej (rovnakej alebo oklieštenej štruktúry)
Najprv sa vytvorí pomocná tabuľka.
CREATE TABLE TMP_KARTA
(UNITID NUMBER(4) NOT NULL,
CKNAME VARCHAR(50))
INSERT INTO TMP_KARTA (UNITID, CKNAME) SELECT UNITID, CKNAME FROM odbytckarta WHERE odbytckarta.CKNAME LIKE 'Farba%'

ZMENA ZÁZNAMOV --- UPDATE

UPDATE tabulka SET stlpec = hodnota [,stlpec1 = hodnota ..]
UPDATE odbytckarta SET CKNAME = 'Farba sivo-modra', UNITID = 2 WHERE ID = 14
Ak chceme zmeniť záznam podľa nejakého iného môžeme použiť vnorený select.
UPDATE odbytckarta SET UNITID = (SELECT UNITID FROM odbytckarta WHERE CKNAME = 'Farba modra') WHERE CKNAME = 'Farba sivo-modra'

Môžeme meniť cez vnorený select aj viac stĺpcov.
UPDATE odbytcvydaj SET (CKARTAID, QUANTITY) = (SELECT CKARTAID, QUANTITY*2 FROM odbytcvydaj WHERE ID = 15) WHERE ID = 10
Zmeniť sa dá aj viac záznamov (alebo všetky)
UPDATE odbytcvydaj SET SALEPRICE = SALEPRICE * 1.2

MAZANIE ZÁZNAMOV --- DELETE

DELETE [FROM] tabuľka WHERE podmienka
DELETE FROM odbytcvydaj WHERE QUALITY > 1
POZN. Kde sa dá delete na celu tabuľku vymažú za len dáta z nej. Na odstránenie tabuľky treba použiť príkaz DROP TABLE tabuľka

Transakcie

COMMIT, ROLLBACK, SAVEPOINT
COMMIT - potvrdenie zmien
ROLLBACK - vrátenie zmien (pred COMMITom)
SAVEPOINT TO meno - bod návratu. ROLLBACK sa vykoná (ak je tak zapísaný) len po ten bod návratu:
SAVEPOINT zvýšenie;
UPDATE odbytcvydaj SET SALEPRICE = SALEPRICE * 1.2;
SAVEPOINT zmazanie;
DELETE FROM odbytcvydaj WHERE CKARTAID = 7;
ROLLBACK TO zmazanie;
V skutočnosti sa vykoná len UPDATE pretože DELETE bol pomocou ROLLBACK TO vrátený naspäť ku bodu návratu. (Samozrejme na konci musí byt potvrdenie zmien pomocou COMMIT)

FUNKCIE

Na kontrolu a ukážku funkcii môžeme použiť tabuľku DUAL. Obsahuje len jeden stĺpec DUMMY varchar2(1) a obsahuje len jednu hodnotu X.
SELECT 1+1 FROM DUAL
1+1

2

1.1.1.1 GONIOMETRICKÉ FUNKCIE --- SIN(n), COS(n), TAN(n)

SIN(30 * 3.14159265359/180)
SELECT SIN(30 * 3.14159265359/180) FROM DUAL
SIN(30*3.14159265359/180)

.5
Alebo krajšie je to:
SELECT SIN(30 * 3.14159265359/180) AS " SIN(30 stupnov)" FROM DUAL
SIN(30 stupňov)

.5

INVERZNE GONIOMETRICKÉ FUNKCIE --- ASIN(n), ACOS(n), ATAN(n)
Funkcie vracajú hodnotu uhlu v radiánoch
SELECT ASIN(0.5) FROM DUAL
ASIN(0.5)

.52359878
Kde chceme výsledok v stupňoch a nie radiánoch:
SELECT ASIN(0.5) * 180/3.14159265359 FROM DUAL
ASIN(0.5)*180/3.14159265359

30

HYPERBOLICKE FUNKCIE --- SINH(n), COSH(n), TANH(n)
Funkcie vracajú hodnotu príslušnej hyperbolickej funkcie pre parameter n.

DALSIE MATEMATICKE FUNKCIE --- ...

Prirodzený logaritmus --- LN(n)
SELECT LN(1000) FROM DUAL
LN(1000)

6.9077553

Mocnina čísla e --- EXP(n)
SELECT EXP(6.90775528) FROM DUAL
EXP(6.90775528)

1000

Absolútna hodnota --- ABS(n)
SELECT ABS(-40) FROM DUAL
ABS(-40)

40

Zistenie znamienka čísla --- SIGN(n)
SELECT SIGN(10) FROM DUAL
SIGN(10)

1
SELECT SIGN(-0) FROM DUAL
SIGN(-0)

0
SELECT SIGN(-10) FROM DUAL
SIGN(-10)

-1

Zaokrúhľovanie čísiel --- ROUND(n, [m])
SELECT ROUND(3.14159265359, 2) FROM DUAL
ROUND(3.14159265359,2)

3.14
SELECT ROUND(3.14159265359, 4) FROM DUAL
ROUND(3.14159265359,4)

3.1416
SELECT ROUND(3.14159265359) FROM DUAL
ROUND(3.14159265359)

3
Da sa použiť aj pri zaokrúhľovaní dátovej a časovej hodnoty.

Odstránenie desatinnej časťi čísla --- TRUNC(n [,m])
SELECT TRUNC(3.14159265359,4) FROM DUAL
TRUNC(3.14159265359,4)

3.1415
SELECT TRUNC(3.14159265359) FROM DUAL
TRUNC(3.14159265359)

3

Najbližšie menšie cele číslo --- FLOOR(n [,m])
SELECT FLOOR(3.14159265359) FROM DUAL
FLOOR(3.14159265359)

3
SELECT FLOOR(3.9) FROM DUAL
FLOOR(3.9)

3

Najbližšie vyššie celé číslo --- CEIL(n [,m])
SELECT CEIL(3.14159265359) FROM DUAL
CEIL(3.14159265359)

4

Všeobecná mocnina --- POWER(m,n)
n-ta mocnina čísla m
SELECT POWER(2,8) FROM DUAL
POWER(2,8)

256

Druha odmocnina --- SQRT(n)
SELECT SQRT(25) FROM DUAL
SQRT(25)

5

Delenie modulo (vracia zbytok po delení) --- MOD(delenec, deliteľ)
SELECT MOD(13,7) FROM DUAL
MOD(13,7)

6

DALSIE FUNKCIE --- BITAND, DECODE, WIDTH_BUCKET
Bitový logicky súčin --- BITAND(m ,n)
Priklad:
0 1 1 1 1 1 1 1 (127 dekadicky)
AND 1 1 1 1 0 0 0 0 (240 dekadicky)
---
0 1 1 1 0 0 0 0 (112 dekadicky)
SELECT BITAND(127,240) FROM DUAL
Výsledok by mal byt 112 ale vyhlásilo to chybu: ORA-00932: inconsistent datatypes ?????

Dekódovacia funkcia --- DECODE(vyraz, {vstup, výstup} , [default])
Ak je v tabuľke VOLANIA:
Novak 45
Fedor 32
Pokorny 33
Tak potom:
SELECT meno, DECODE(číslo,32,'Belgicko',33,'Francuzsko',45,'Dansko') FROM volania
vypíše:
Novak Dansko
Fedor Belgicko
Pokorny Francuzsko
! V praxi sa časťo používa spojenie týchto dvoch funkcii na výpočty bitových máp, stavoch zariadení. ! (Dobrý priklad na tuto temu kniha: ORACLE (Luboslav Lacko)
Rovnomerne zoskupenie --- WIDTH_BUCKET(vyraz, minimum, maximum, počet_skupin)
V prípade potreby sa vytvorí tiež skupina 0 ktorá bude obsahovať hodnoty menšie ako je požadované minimum a skupina počet_skupin+1 ktorá bude obsahovať hodnoty väčšie ako požadované maximum.
Príklad: Rozdelenie zamestnancov do platových skupín.
V tabuľke zamestnanci mame zamestnancov s ich platmi:
Novak 5280
Fedor 12300
Pokorny 3600
Pavlovic 15990
Pukanec 7250
Fesna 4100
Zajac 45000
SELECT meno, mzda, WIDTH_BUCKET(mzda, 4000, 20000,5)
Novak 5280 1
Fedor 12300 3
Pokorny 3600 0
Pavlovic 15990 4
Pukanec 7250 2
Fesna 4100 1
Skerlik 45000 6
Novak je pod stanoveným minimom, **Skerlik** je nad nim a v skupine 5 nie je nikto.

FUNKCIE PRE PRACU S TEXTOVYMI REŤAZCAMI

Konverzia na veľké písmena --- UPPER

SELECT UPPER ('Hore - Dole') FROM DUAL
UPPER('HORE

HORE - DOLE

Konverzia na malé písmena --- LOWER
SELECT LOWER ('Hore - Dole') FROM DUAL
LOWER('HORE
---
hore - dole

Konverzia pravého písmena slova na veľké písmeno --- INITCAP
SELECT INITCAP ('isli dievcata po ceste, ale potom stretli skerlika') FROM DUAL
INITCAP('ISLYDIEVCATAPOCESTE,ALEPOTOMSTRETLISKERLI

Isli Dievcata Po Ceste, Ale Potom Stretli Skerlika

Konverzia čísla na znak --- CHR
CHR(číslo)
SELECT CHR(65) FROM DUAL
CHR(65)
-
A
SELECT CHR(65) || CHR(66) || CHR(67) FROM DUAL
CHR(65) || CHR(66) || CHR(67)

ABC

Nahradenie znakov v reťazci --- REPLACE
REPLACE(reťazec1, reťazec2 [,reťazec3])
reťazec1 - pôvodný reťazec
reťazec2 - reťazec, ktorý sa ma v pôvodnom reťazci nahradit reťazcom3
reťazec3 - reťazec, ktorý nahradzuje reťazec2
Kde nie je reťazec 3 uvedený, budú všetky výskyty jednoducho odstranene.
SELECT REPLACE('Ludia vedia','Ludia', 'Zvierata') FROM DUAL
REPLACE('LUDIA

Zvieratá vedia
SELECT REPLACE('Ludia vedia','vedia') FROM DUAL
REPLAC

Ludia
SELECT REPLACE('JACK and JUE','J','BL') FROM DUAL
REPLACE('JACKA

BLACK and BLUE

Nahradenie špecifickej skupiny znakov v reťazci --- TRANSLATE
TRANSLATE(reťazec1, reťazec2 [,reťazec3])
reťazec1 - pôvodný reťazec
reťazec2 - reťazec obsahujúci zoznam znakov ktorý sa ma v pôvodnom reťazci nahradiť znakmi z reťazca3
reťazec3 - reťazec obsahujúci nove znaky
Kde ma tretí parameter menší počet znakov ako druhy, budú všetky znaky, ktoré sa v reťazci2 nachádzajú na chýbajúcich miestach vymazane (z reťazca1)
SELECT TRANSLATE('PIN mojej kerditky je 9821','0123456789','XXXXXXXXXX') FROM DUAL
TRANSLATE('PINMOJEJKERDITK

PIN mojej kreditky je XXXX
SELECT TRANSLATE('registracne číslo mojej kopie Windows je
321QWE456ASD','0123456789ABCDEFGHIJKLMNOPRSTUVWQXYZ','XXXXXXXXXXYYYYYYYYYYYYYYYYYYYYYYYY') FROM DUAL
TRANSLATE('REGISTRACNEČÍSLOMOJEJKOPIEWINDOWSJE321QWE4
---
registračné číslo mojej kópie Windows je XXXYYYXXXYYY

Orezanie znakov --- TRIM
TRIM(prikaz)
LEADING - odstránenie zľava
TRAILING - odstránenie sprava
Odstráni zadané znaky zľava a sprava z reťazca
SELECT TRIM(0 FROM 00012340056789000) FROM DUAL

TRIM(0FROM0

12340056789
SELECT TRIM(LEADING 0 FROM 00012340056789000) FROM DUAL
TRIM(LEADING0F

12340056789000
SELECT TRIM(TRAILING 0 FROM 00012340056789000) FROM DUAL
TRIM(TRAILI

12340056789
Pozn. Výsledok je správny lebo je to číslo! Ak to dáme ako znaky:
SELECT TRIM(TRAILING '0' FROM '00012340056789000') FROM DUAL
TRIM(TRAILING'

00012340056789

Orezanie znakov zľava --- LTRIM
LTRIM(reťazec1, reťazec2)
Odstráni z ľavej časťi reťazca1 všetky znaky ktoré obsahujú reťazec2. Ak je reťazec2 medzera (alebo nie je) odstráni funkcia zľava všetky medzery.
SELECT LTRIM('aabbccAABBCCaabbccDD','abc') FROM DUAL
LTRIM('AABBCCA

AABBCCaabbccDD
SELECT LTRIM(' aabbccAABBCCaabbccDD') FROM DUAL
LTRIM('AABBCCAABBCCA
----
aabbccAABBCCaabbccDD

Orezanie znakov sprava --- RTRIM
RTRIM(reťazec1, reťazec2)
Odstráni z pravej časťi reťazca1 všetky znaky ktoré obsahujú reťazec2. Ak je reťazec2 medzera (alebo nie je) odstráni funkcia sprava všetky medzery.
SELECT RTRIM('aabbccAABBCCaabbccDD','cD') FROM DUAL
RTRIM('AABBCCAAB
----
aabbccAABBCCaabb
SELECT RTRIM('aabbccAABBCCaabbccDD ') FROM DUAL
RTRIM('AABBCCAABBCCA
----
aabbccAABBCCaabbccDD

Vyber podmnoziny znakov v reťazci --- SUBSTR
SUBSTR(reťazec, pozicia, dĺžka)
Novy reťazec bude obsahovať počet znakov zadaných parametrom dĺžka od pozície. Kde je druhy parameter záporný, uplatňuje sa od konca reťazca.
SELECT SUBSTR('Druhy parameter slúži pre pozíciu nového reťazca',7,9) FROM DUAL
SUBSTR('D
----
parameter
SELECT SUBSTR('Druhy parameter slúži pre pozíciu nového reťazca',0,5) FROM DUAL
SUBST

Druhy
SELECT SUBSTR('Druhy parameter slúži pre pozíciu nového reťazca',-7,7) FROM DUAL
SUBSTR(
----
reťazca

Doplnenie reťazca zľava --- LPAD
LPAD(reťazec1, dĺžka, reťazec2)
reťazec1 - pôvodný reťazec
dĺžka - dĺžka nového reťazca
reťazec2 - obsahuje znaky, ktorými sa cyklicky doplní reťazec1 na požadovaný počet znakov
SELECT LPAD('Zoznam',20,'<>') FROM DUAL
LPAD('ZOZNAM',20,'<>

<><><><><><><>Zoznam

Doplnenie reťazca sprava --- RPAD

RPAD(reťazec1, dĺžka, reťazec2)
reťazec1 - pôvodný reťazec
dĺžka - dĺžka nového reťazca
reťazec2 - obsahuje znaky, ktorými sa cyklicky doplní reťazec1 na požadovaný počet znakov
SELECT RPAD('Zoznam',20,'><') FROM DUAL
RPAD('ZOZNAM',20,'><

Zoznam><><><><><><><
Pozn. Dobrý príklad na tuto tému kniha: ORACLE (Luboslav Lacko)
Spojovanie reťazcov --- CONCAT
CONCAT(reťazec1,reťazec2)
reťazec1 - prvý reťazec(ľavá časť výsledného reťazca)
reťazec2 - druhy reťazec(pravá časť výsledného reťazca)
SELECT CONCAT('Jan','Novak') FROM DUAL
CONCAT('

JanNovak
Slúži vlastne tak isto ako zreťazenie |
SELECT 'Jan' || ' ' || 'Novak' FROM DUAL
'JAN'||'

Jan Novak

Dĺžka reťazca --- LENGHT
LENGHT(reťazec)
SELECT LENGHT('Novak') FROM DUAL

FUNKCIE PRE PRACU S DÁTUMOM A ČASOM

Systémový aktuálny dátum a čas --- SYSDATE
SELECT SYSDATE FROM DUAL
SYSDATE

12-JAN-03
Zmena výpisu dátumu a času --- TO_CHAR
SELECT TO_CHAR(SYSDATE,'DD.MM.YYYY HH24:MI:SS') FROM DUAL
TO_CHAR(SYSDATE,'DD

13.01.2003 09:05:59

Aktuálny dátum a čas vzhľadom k časovému pásmu --- CURRENT_DATE
SELECT CURRENT_DATE FROM DUAL

Formát výpisu dátového typu je možné nastaviť natrvalo pomocou:
ALTER SESSION SET NLS_DATE_FORMAT = 'DD.MM.YYYY HH24:MI:SS'
Session altered.
A po kontrole:
SELECT SYSDATE FROM DUAL
SYSDATE

13.01.2003 09:11:28

Ďalšie funkcie pre dátumovú aritmetiku
NEXT_DAY(dátum, den_v_tyzdni)
Funkcia vracia novy dátum vytvorený z pôvodného dátumu a to tak, že nove dátum pripadá na nasledujúci zadaný deň v týždni.
SELECT SYSDATE, NEXT_DAY(SYSDATE,'SUNDAY') FROM DUAL
SYSDATE NEXT_DAY(SYSDATE,'S
--- ---
13.01.2003 09:20:39 19.01.2003 09:20:39

LAST_DAY(dátum)
Funkcia vracia dátum posledneho dna v mesiaci podľa mesiaca zadaneho dátumu.
SELECT LAST_DAY(SYSDATE) FROM DUAL
LAST_DAY(SYSDATE)

31.01.2003 09:24:45
Príklad:
SELECT 'Do konca mesiaca ostáva ešte dni ',LAST_DAY(SYSDATE) - SYSDATE FROM DUAL

'DOKONCAMESIACAOSTAVAESTEDNI' LAST_DAY(SYSDATE)-SYSDATE
---------- ----------
Do konca mesiaca ostáva ešte dni 18

ADD_MONTHS(dátum, n_mesiacov)
SELECT SYSDATE, ADD_MONTHS(SYSDATE,2) FROM DUAL
SYSDATE ADD_MONTHS(SYSDATE,
---------- ----------
13.01.2003 09:31:32 13.03.2003 09:31:32

MONTHS_BETWEEN(dátum1, dátum2)
Vracia číslo ktoré vyjadruje počet mesiacov medzi dátumami
SELECT SYSDATE, MONTHS_BETWEEN(SYSDATE, TO_DATE('13.5.2003')) FROM DUAL
SYSDATE MONTHS_BETWEEN(SYSDATE,TO_DATE('13.5.2003'))
---------- ----------
13.01.2003 09:37:06 -4

SELECT SYSDATE, MONTHS_BETWEEN(SYSDATE, TO_DATE('24.7.1999')) FROM DUAL
SYSDATE MONTHS_BETWEEN(SYSDATE,TO_DATE('24.7.1999'))
---------- ----------
13.01.2003 09:37:52 41.658106

Zaokrúhľovanie dátumu a času --- ROUND
ROUND(dátum, zaokrúhlenie)
zaokrúhlenie - určuje ako sa bude zaokrúhľovať:
YEAR - zaokrúhľuje na cele roky (od 1.7. hore)
MONTH - zaokrúhľuje na cele mesiace (od 16 hore)
DDD - zaokrúhľuje na cele dni
DAY - zaokrúhľuje na prvý deň v týždni
HH - zaokrúhľuje na cele hodiny
MI - zaokrúhľuje na cele minúty
SELECT SYSDATE, ROUND(SYSDATE) FROM DUAL
SYSDATE ROUND(SYSDATE)
---------- ----------
13.01.2003 20:48:27 14.01.2003 00:00:00
SELECT SYSDATE, ROUND(SYSDATE,'YEAR'), 'YEAR' FROM DUAL
SYSDATE ROUND(SYSDATE,'YEAR 'YEA
---------- ---------- ----
13.01.2003 09:45:10 01.01.2003 00:00:00 YEAR

Orezanie dátumu a času --- TRUNC
TRUNC(dátum, časť)
časť - určuje ako sa ma pôvodný dátum orezať. Hodnoty parametrov ako u funkcie ROUND
SELECT SYSDATE, TRUNC(SYSDATE) FROM DUAL
SYSDATE TRUNC(SYSDATE)
---------- ----------
13.01.2003 20:49:05 13.01.2003 00:00:00
Rozdiel medzi ROUND a TRUNC je evidentný:
SELECT SYSDATE, ROUND(SYSDATE), TRUNC(SYSDATE) FROM DUAL
SYSDATE ROUND(SYSDATE) TRUNC(SYSDATE)
---------- ---------- ----------
13.01.2003 20:50:43 14.01.2003 00:00:00 13.01.2003 00:00:00
Vyber požadovanej časti dátumu a času --- EXTRACT
EXTRACT(prikazovy_reťazec)
prikazovy_reťazec je v tvare ‘časť FROM dátum’ kde časť: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND
SELECT SYSDATE, EXTRACT(YEAR FROM SYSDATE) FROM DUAL
SYSDATE EXTRACT(YEARFROMSYSDATE)
---------- ----------
13.01.2003 09:58:07 2003

FUNKCIE PRE KONVERZIU DATOVYCH TYPOV

Prevod binárneho čísla na desatinne --- BIN_TO_NUM
SELECT BIN_TO_NUM(1,1,1,1,0,0,0,0) FROM DUAL
Pozn. Z nepochopitelnych pricin mi vyhlasilo chybu - invalid column name

Prevod dátumových typov na reťazec --- TO_CHAR
TO_CHAR(datetime, [,’fmt_reťazec’,’nls_parameter’])
fmt_reťazec - nepovinny formatovaci reťazec

nls_parameter - parameter určujúci jazyk
SELECT TO_CHAR (SYSDATE,'YEAR') FROM DUAL
TO_CHAR(SYSDATE,'YEAR')

TWO THOUSAND THREE

Prevod číselných typov na reťazec --- TO_CHAR
NUMBER prevádza na VARCHAR2
TO_CHAR(number, [,'fmt_reťazec','nls_parameter'])
SELECT TO_CHAR (-10000,'L99G999D99MI') FROM DUAL
TO_CHAR(-10000,'L99G

$10,000.00-

Prevod textových reťazcov na dátum a čas --- TO_DATE
TO_DATE(character, 'fmt_reťazec'[,'nls_parameter'])
Pozn. Popis parametrov v tabuľke v knihe: ORACLE (Luboslav Lacko) na str. 177 a 178
SELECT TO_DATE('25.12.1998 13:30:22','dd.mm.yyyy, HH24:MI:SS') FROM DUAL
TO_DATE('25.12.1998

25.12.1998 13:30:22
SELECT TO_DATE('January 15, 1989, 11:00 A.M.','Month dd, YYYY, HH:MI A.M.','NLS_DATE_LANGUAGE = American') FROM DUAL
TO_DATE('JANUARY15,

15.01.1989 11:00:00

Prevod textových reťazcov na dátové typy LOB --- TO_LOB
TO_LOB(long_column)
long_column - stĺpec obsahujúci dátový typ LONG alebo LONG RAW
Tuto funkciu je mozne použiť len v časti SELECT príkazu INSERT INTO.

Prevod textovych reťazcov na ciselne datove typy --- TO_NUMBER
TO_NUMBER(character, 'fmt_reťazec'[,'nls_parameter'])
Pozn. Popis parametrov v tabuľke v knihe: ORACLE (Luboslav Lacko) na str.179
SELECT TO_NUMBER ('50000') FROM DUAL
TO_NUMBER('50000')

50000

AGREGACNE FUNKCIE

Aritmeticky priemer --- AVG
AVG([DISTINCT] [ALL] vyraz)
DISTINCT spôsobí, že každá hodnota sa započíta len raz, ALL zaisti započítanie príslušnej hodnoty pri jej každom výskyte
SELECT AVG(QUANTITY) "Priemerne mnozstvo" FROM odbytcvydaj
Priemerne množstvo

3.2857143

Počet hodnôt --- COUNT
COUNT([DISTINCT] [ALL] vyraz)
SELECT COUNT(*) "Celkom záznamov" FROM odbytcvydaj
Celkom záznamov

14
SELECT COUNT(DISTINCT cdod_lisid) "Výdaje z poctu dodacích listov" FROM odbytcvydaj
Výdaje z poctu dodacích listov

5

Maximum --- MAX
NULL sa ignoruje
MAX(vyraz)
SELECT MAX(QUANTITY) "Maximalne predane mnozstvo" FROM odbytcvydaj
Maximálne predane množstvo

7
Minimum --- MIN

NULL sa ignoruje
MIN(vyraz)
SELECT MIN(QUANTITY) "Minimalne predane mnozstvo" FROM odbytcvydaj
Minimálne predane množstvo

1
Funkcie MIN a MAX sa daju použiť aj na porovnanie stringovych a dátumových hodnôt.
Sucet --- SUM
SUM([DISTINCT] [ALL] vyraz)
SELECT SUM(QUANTITY) "Cele predane mnozstvo" FROM odbytcvydaj
Cele predane množstvo

46

Akceptovanie hodnoty NULL --- NVL
NVL(vyraz1, vyraz2)
V prípade:
SELECT AVG(QUANTITY) "Priemerne mnozstvo" FROM odbytcvydaj
by do výpočtu priemeru zobralo len hodnoty kde nebola NULL
avšak takto
SELECT AVG(NVL(QUANTITY,0)) "Priemerne mnozstvo" FROM odbytcvydaj
by do vypoctu priemeru zobralo aj hodnoty kde bola NULL

Ohodnotenie --- RANK
Vyhodnotenie záznamu podľa určitých kritérií. Na ktoré miesto sa dostane novy vydaj.
RANK(hodnota) WITHIN GROUP (ORDER BY stĺpce [ASC|DESC])
SELECT RANK(5) WITHIN GROUP (ORDER BY QUANTITY) "Poradie" FROM odbytcvydaj
Pozn. Z nepochopiteľných príčin mi vyhlásilo chybu - FROM keyword not found where expected

Percentuálne ohodnotenie --- PERCENT_RANK
Výsledok je v rozmedzí 0 az 1
PERCENT_RANK(hodnota) WITHIN GROUP (ORDER BY stĺpce [ASC|DESC])
SELECT PERCENT_RANK(5) WITHIN GROUP (ORDER BY QUANTITY) "Poradie" FROM odbytcvydaj

Odchylka --- VARIANCE
Vracia premenlivu odchylku pre skupinu záznamov
VARIANCE([DISTINCT|ALL] vyraz)
SELECT VARIANCE(QUANTITY) "Odchylka" FROM odbytcvydaj
Odchýlka

2.6813187

Štandartná odchýlka --- STDDEV
STDDEV ([DISTINCT|ALL] vyraz)
SELECT STDDEV(QUANTITY) "Standartna odchylka" FROM odbytcvydaj
Standartna odchylka

1.6374733

ANALYTICKE FUNKCIE

Aritmeticky priemer z množiny hodnôt --- AVG
AVG([DISTINCT] [ALL] vyraz [OVER] analyticky_vyraz
Dobrý príklad na tuto tému kniha: ORACLE (Luboslav Lacko)

Počet hodnôt --- COUNT
SELECT ID, CKARTAID, CDOD_LISID, QUANTITY COUNT(*) OVER (ORDER BY QUANTITY DESC RANGE BETWEEN 0 PRECEDING AND 0 FOLLOWING) as NEW_COUNT FROM odbytcvydaj

OSTATNE FUNKCIE

Výpis jednotlivých znakov --- DUMP
DUMP (text [,fmt_reťazec [,zaciatok [,dĺžka]]])
fmt_reťazec - definuje ciselnu sustavu
8 - osmičková sústava
10 - desiatková sústava
16 - šestnásťková sústava
17 - výsledok bude vypísaný po jednotlivých znakoch
SELECT DUMP('abc',16) FROM DUAL

DUMP('ABC',16)

Typ=96 Len=3: 61,62,63
Ak sa pripočíta k číslu fmt_reťazca 1000 vypíše sa tiež informácia o znakovej sade.
SELECT DUMP('abc',1016) FROM DUAL
DUMP('ABC',1016)

Typ=96 Len=3 CharacterSet=WE8ISO8859P1: 61,62,63

Najvyššia hodnota --- GREATEST
GREATEST (vyraz [,vyraz ..])
SELECT GREATEST(92,16,278,34) FROM DUAL
GREATEST(92,16,278,34)

278
Výrazy môžu byť rôznych typov a prevedú sa na dátový typ pravého výrazu.
SELECT GREATEST('vlado',92,'16','Peter',34) FROM DUAL
GREAT
---
vlado

Najmenšia hodnota --- LEAST
LEAST (vyraz [,vyraz ..])
Opak funkcie GREATEST
SELECT LEAST(92,16,278,34) FROM DUAL
LEAST(92,16,278,34)

16

Aktuálny užívateľ --- USER
SELECT USER FROM DUAL
USER

VSKERLIK

Identifikátor aktuálneho užívateľa --- UID
SELECT UID FROM DUAL
UID

37
Veľkosť dátového typu --- VSIZE
SELECT VSIZE(314159) "bajtov" FROM DUAL
bajtov

4
SELECT VSIZE(3.14159) "bajtov" FROM DUAL
bajtov

5

Príkazy pre tabuľku

1.1.1.2 VYTVORENIE NOVEJ TABUĽKY --- CREATE TABLE

CREATE TABLE [schema.]nazov_tabuľky
(
nazov_stĺpca datovy_typ [DEFAULT vyraz] [,
nazov_stĺpca2 datovy_typ [DEFAULT vyraz],
...
nazov_stĺpcaN datovy_typ [DEFAULT vyraz]]
)
Napr.
CREATE TABLE ODBYTCVYDAJ (
ID NUMBER NOT NULL,
CKARTAID NUMBER NOT NULL,
CDOD_LISID NUMBER NOT NULL,
DATEINSERTED DATE NOT NULL,
QUANTITY NUMBER (3) NOT NULL,

PURCHASEPRICE NUMBER (2),
SALEPRICE NUMBER (2),
QUALITY NUMBER,
UNIQUE (ID)
)

Integritne Obmedzenia

NOT NULL - nesmie obsahovať NULL
PRIMARY KEY - jednoznačne identifikuje každý záznam v tabuľke. Môže byt maximalne jeden ale môže byt vytvorený aj z viacerých stĺpcov.
UNIQUE KEY - stĺpec alebo viac stĺpcov. Musí byt jedinečný v tabuľke
FOREIGN KEY - cudzí kľuč. Definuje vzťah k primárnemu kľúču v inej tabuľke pripadne aj tej istej.

CREATE TABLE ODBYTCVYDAJ1 (
ID NUMBER NOT NULL,
CKARTAID NUMBER NOT NULL,
CDOD_LISID NUMBER NOT NULL,
DATEINSERTED DATE NOT NULL,
QUANTITY NUMBER (3) NOT NULL,
PURCHASEPRICE NUMBER (2),
SALEPRICE NUMBER (2),
QUALITY NUMBER,
UNIQUE (ID),
FOREIGN KEY (CDOD_LISID) REFERENCES ODBYTCDOD_LIS(ID)
)

Musí byt však definovaný PRIMARY KEY(ID) v tabuľke ODBYTCDOD_LIS

CHECK - definuje podmienku ktorá musí byt pre ukladane hodnoty splnena.
Napr. Aby hodnoty v QUALITY bola len v intervale 1 az 9

CREATE TABLE ODBYTCVYDAJ2 (
ID NUMBER NOT NULL,
CKARTAID NUMBER NOT NULL,
CDOD_LISID NUMBER NOT NULL,
DATEINSERTED DATE NOT NULL,
QUANTITY NUMBER (3) NOT NULL,
PURCHASEPRICE NUMBER (2),
SALEPRICE NUMBER (2),
QUALITY NUMBER,
UNIQUE (ID),
FOREIGN KEY (CDOD_LISID) REFERENCES ODBYTCDOD_LIS(ID),
CHECK (QUALITY BETWEEN 1 AND 9)
)

Všetky obmedzenia je možné pridávať aj odoberať ale nie meniť(pomocou ALTER TABLE sa to nedá). Obmedzenia je možné tiež deaktivovat a aktivovať bez toho aby sa muselo mazať.
Pozrieť aktuálne obmedzenia:

SELECT constraint_name, constraint_type, search_condition FROM user_constraints WHERE table_name = 'ODBYTCVYDAJ2'

CONSTRAINT_NAME C SEARCH_CONDITION
------------------------------ - --------------------------------
SYS_C001295 C "QUANTITY" IS NOT NULL
SYS_C001296 C QUALITY BETWEEN 1 AND 9
SYS_C001297 U
SYS_C001298 R
SYS_C001291 C "ID" IS NOT NULL
SYS_C001292 C "CKARTAID" IS NOT NULL
SYS_C001293 C "CDOD_LISID" IS NOT NULL
SYS_C001294 C "DATEINSERTED" IS NOT NULL

ZMENA V TABUĽKE --- ALTER TABLE

ALTER TABLE nazov_tabuľky
(
ADD nazov_stlca typ [integritne obmedzenie],
MODIFY ...
DROP COLUMN ...
)

ADD - pridanie nového stĺpca do tabuľky
MODIFY - zmena definície stĺpca
DROP COLUMN - odstránenie stĺpca z tabuľky
Prida do tabuľky odbytcvydaj stĺpec ZLAVA:

ALTER TABLE odbytcvydaj ADD (ZLAVA number(6,2))

Zmaže z tabuľky odbytcvydaj stĺpec ZLAVA:

ALTER TABLE odbytcvydaj DROP COLUMN ZLAVA

Ak chcem zmeniť veľkosť rozsahu u stĺpca QUANTITY:

ALTER TABLE odbytcvydaj MODIFY (QUANTITY number(7,3))
Pozn. Spätná zmena na menší rozsah už nepôjde(len ak je stĺpec bez údajov). U stringovych typov ide aj menšia zmena ale len vtedy ak sa do nej zmestia všetky záznamy.
Integritné obmedzenia sa dajú pridávať, rušiť, aktivovať(už vytvorene) a deaktivovat. Nedajú sa modifikovať. To všetko sa dá robiť len vtedy ak to vyhovuje dátam v tabuľke.
Pridanie unique:
ALTER TABLE odbytcpartner ADD UNIQUE (PARTNERNAME)
Odstránenie unique:
ALTER TABLE odbytcpartner DROP UNIQUE (PARTNERNAME)
Obmedzenie je možné tiež deaktivovat a aktivovať ENABLE CONSTRAINT a DISABLE CONSTRAINT.
ALTER TABLE odbytckarta DISABLE UNIQUE (CKNAME)
ALTER TABLE odbytckarta ENABLE UNIQUE (CKNAME)
Tiež sa dá pridať cudzí kľuč:
ALTER TABLE ODBYTCDOD_LIS ADD FOREIGN KEY (CPARTNERID) REFERENCES ODBYTCPARTNER(ID)

ODSTRANENIE, ZMENA NAZVU A VYMAZANIE ZÁZNAMOV TABUĽKY --- DROP, RENAME, TRUNCATE

DROP TABLE nazov_tabuľky
Príkaz potvrdí všetky neukončené transakcie. Nedá sa na neho použiť ROLLBACK.
DROP TABLE odbytcvydaj1

RENAME pôvodny_nazov TO novy názov
RENAME odbytcvydaj2 TO ODBYTCVYDAJ200
Overiť sa to dá (okrem iného) aj otázkou do tabuľky systémového katalógu USER_CATALOG
SELECT * FROM cat
Vypíše nám všetky tabuľky, sekvencie ... pod daným userom.

TRUNCATE TABLE nazov_tabuľky
TRUNCATE TABLE vymaže všetky záznamy v tabuľke podobne ako DELETE - ten ale neuvoľňuje použitý priestor. Tiež sa na neho neda sa na neho použiť ROLLBACK.
TRUNCATE TABLE odbytcvydaj200

Pohľady --- VIEW

VYTVORENIE, ZMENA A ZMAZANIE POHLADU

CREATE [OR REPLACE] [FORCE] [NOFORCE] VIEW nazov_pohladu [(alias1 [,alias2]...)] AS vnorený_select [zoznam obmedzenia]
Nad jednou tabuľkou
CREATE VIEW ZOZNAM AS SELECT CKARTAID, QUANTITY, PURCHASEPRICE FROM odbytcvydaj
Zobraziť pohľad:
SELECT * FROM ZOZNAM
Nad dvoma tabuľkami:
CREATE VIEW ZOZNAM AS SELECT CKNAME, QUANTITY, PURCHASEPRICE FROM odbytcvydaj, odbytckarta WHERE odbytcvydaj.ckartaid = odbytckarta.id
Zmena existujúceho pohľadu(ale aj vytvorenie)
CREATE OR REPLACE VIEW ZOZNAM AS SELECT CKNAME, QUANTITY, PURCHASEPRICE, PARTNERNAME FROM odbytcvydaj, odbytckarta, odbytcdod_lis, odbytcpartner WHERE odbytcvydaj.ckartaid = odbytckarta.id AND odbytcvydaj.cdod_lisid = odbytcdod_lis.id AND odbytcdod_lis.cpartnerid = odbytcpartner.id
Odstránenie pohľadu:
DROP VIEW ZOZNAM
Štandartne slúžia pohľady len na prezeranie dát. Samotne dáta neobsahujú. Čiastočne sa dajú používať aj na úpravu dát. Dosiahnutie obmedzenia aby sa úprava nedala robiť --- WITH READ ONLY
CREATE VIEW ZOZNAM AS SELECT CKARTAID, QUANTITY, PURCHASEPRICE FROM odbytcvydaj WITH READ ONLY

MATERIALIZOVANÉ POHĽADY

Materializovaný pohľad je možné vytvoriť len nad tabuľkami, nad ktorými bol vytvorený protokol materializovaného pohľadu (MATERIALIZED VIEW LOG)
Vytvorím 2 tabuľky:

```
CREATE TABLE videokazety
(
ev_číslo NUMBER(4) PRIMARY KEY,
nazov VARCHAR2(15)
);
INSERT INTO videokazety VALUES (1, 'Pocahontas');
INSERT INTO videokazety VALUES (2, 'Lion King');
INSERT INTO videokazety VALUES (3, 'Cinderella');
CREATE TABLE uhrady
(id_uhrady NUMBER(6) PRIMARY KEY,
ev_číslo NUMBER(6),
uhrada NUMBER(9,2)
```

);
INSERT INTO uhrady VALUES (1,1,320);
INSERT INTO uhrady VALUES (2,2,260);
INSERT INTO uhrady VALUES (3,3,350);
INSERT INTO uhrady VALUES (4,1,170);
INSERT INTO uhrady VALUES (5,2,180);
INSERT INTO uhrady VALUES (6,3,210);
CREATE MATERIALIZED VIEW LOG ON videokazety WITH PRIMARY KEY,
ROWID(názov)
INCLUDING NEW VALUES
CREATE MATERIALIZED VIEW LOG ON uhrady WITH PRIMARY KEY,
ROWID(ev_číslo, uhrada)
INCLUDING NEW VALUES
CREATE MATERIALIZED VIEW pozicovna REFRESH FAST ON COMMIT
AS SELECT v.nazov, SUM(u.uhrada) AS suma_uhrad
FROM uhrady u, videokazety v
WHERE u.ev_číslo = v.ev_číslo GROUP BY v.nazov
Vyhlasilo chybu:
ORA-12051: ON COMMIT attribute is incompatible with other options
Ide mi to len bez: REFRESH FAST ON COMMIT
CREATE MATERIALIZED VIEW pozicovna
AS SELECT v.nazov, SUM(u.uhrada) AS suma_uhrad
FROM uhrady u, videokazety v
WHERE u.ev_číslo = v.ev_číslo GROUP BY v.nazov
SELECT * FROM pozicovna
NAZOV SUMA_UHRAD
---------- ----------
Cinderella 560
Lion King 440
Pocahontas 490
Pozn. Ak tam nešlo pridať aj REFRESH FAST ON COMMIT tak ani po pridaní do pôvodných tabuliek a commitnuti sa nezmení materializovaný pohľad a ostanú v ňom len pôvodne dáta ktoré tam boli pri vytvorení.
Materializovaný pohľad zrušíme:
DROP MATERIALIZED VIEW požičovňa

Sekvencie --- SEQUENCE

CREATE SEQUENCE nazov_sekvencie
[START WITH číslo]
[INCREMENT BY číslo]
[MAXVALUE číslo]
[MINVALUE číslo]
[CYCLE | NOCYCLE]
[CACHE číslo | NOCACHE]
CYCLE - generuje hodnoty opakovane
NEXTVAL - ďalšia hodnota
CURVAL - aktuálna hodnota
Príklad:
Začne číslovať od 1000 s prírastkom 100
CREATE SEQUENCE sekvencia1 START WITH 1000 INCREMENT BY 100
Zmeniť sekvenciu:
ALTER SEQUENCE sekvencia1 INCREMENT BY 150 MAXVALUE 2000 CYCLE NOCACHE
Zmazať sekvenciu:
DROP SEQUENCE sekvencia1

Indexy

Automaticky sa vytvoria u primárneho kľúča.
CREATE [UNIQUE][BITMAP] INDEX nazov_indexu ON nazov_tabuľky (stĺpec_tabuľky [,stĺpec_tabuľky])
CREATE INDEX ix_city ON odbytcpartner(city)
Zmazanie indexu:
DROP INDEX ix_city

BITMAPOVE INDEXY
Sa používajú tam kde je jasne koľko možností naplnenia bude mat stĺpec. Napr. Pohlavie, stav, kraj.
Napr. Pre tabuľku:
CREATE TABLE Ludia
(

meno VARCHAR2(25),
pohlavie CHAR(1)
CHECK (pohlavie IN ('M','Z'))
)
Nad stĺpcom pohlavie vytvoríme bit mapový index:
CREATE BITMAP INDEX ix_pohlavie ON Ludia (pohlavie)

*****Premenná stringova:**
SET SERVEROUT ON SIZE 10000;
DECLARE
num_premenna NUMBER(3) := 77;
BEGIN
DBMS_OUTPUT.PUT_LINÉ('num_premenna = ' || num_premenna);
END;
Výpis:
num_premenna = 77
PL/SQL procedure successfully completed.

Premenná dátumová:
DECLARE
dátum_čas DATE;
BEGIN
dátum_čas := TO_DATE('25.12.2002 13:30:20','DD.MM.YYYY, HH24:MI:SS');
DBMS_OUTPUT.PUT_LINÉ('Dátum a čas = ' || dátum_čas);
END;
Dátum a čas = 25-DEC-02
PL/SQL procedure successfully completed.
Pozn. Ak chceme zmeniť výpis:
ALTER SESSION SET NLS_DATE_FORMAT = 'DD.MM.YYYY HH24:MI:SS'
Potom bude výpis:
Dátum a čas = 25.12.2002 13:30:20

Premenná BOOLEAN
Nemôžeme ju uložiť do tabuľky a ani výpis pomocou DBMS_OUTPUT.PUT_LINÉ nie je možný.
DECLARE
logicka_mremenna BOOLEAN;
BEGIN
logicka_mremenna := TRUE;
END;

Premenná podľa typu inej premennej alebo stĺpca tabuľky
Konštrukcia %TYPE zadefinuje typ a rozsah premennej podľa inej premennej alebo stĺpca tabuľky.
DECLARE
str_Text VARCHAR2(50);
str_Text1 str_Text%TYPE;
var_partnername odbytcpartner.PARTNERNAME%TYPE;
BEGIN
str_Text1 := 'Toto je dlhsi text';
var_partnername := 'VSK software';
END;

Komentáre
Jedno riadkový komentár --
Viac riadkový komentár začína sa s /* a konci */
DECLARE
-- deklarujem numericku premennu
num_premenna NUMBER(3) := 77;
BEGIN
/ Spúšťa sa výpis na obrazovku*
ale len keď je predtým SET SERVEROUT ON SIZE číslo/*
DBMS_OUTPUT.PUT_LINÉ('num_premenna = ' || num_premenna);
END;