POGLAVLJE

11

Funkcije za konverziju i transformaciju

Ovo poglavlje je posvećeno funkcijama koje su namenjene za konverziju, ili transformaciju jednog tipa podatka u neki drugi tip. Do sada smo u ovoj knjizi obradili četiri osnovna tipa podataka i njima pridružene funkcije:

- Podaci tipa CHAR (znakovni stringovi fiksne dužine) i VARCHAR2 (znakovni stringovi promenljive dužine) mogu u sebi sadržati bilo koje slovo abecede, bilo koju cifru i sve ostale simbole koji se mogu uneti sa tastature. Doslovni znakovni stringovi moraju se unositi između jednostrukih navodnika:

  ```
  'Sault Ste. Marie!'
  ```

- Podaci tipa NUMBER u sebi mogu sadržati isključivo cifre od 0 do 9, decimalnu tačku i, po potrebi, znak za minus, kojim se označavaju brojevi manji od nule. Numerički podaci konstantne vrednosti unose se bez navodnika:

  ```
  -246.320
  ```

- DATE je specijalni tip podataka, koji u sebi sadrže informacije o datumu, vremenu i časovnoj zoni. Podrazumevani format podataka ovog tipa glasi DD-MON-YY, ali ih pomoću funkcije TO_CHAR možete prikazivati na mnoštvo različitih načina, o čemu je bilo više reči u Poglavlju 10. Doslovni (konstantni) datumski podaci se takođe moraju unositi pod jednostrukim navodnicima:

  ```
  '26-AUG-81'
  ```

Kao što ste mogli da naučite iz poglavlja 7, 8, 9 i 10, svakom od ovih tipova podataka pridružena je odgovarajuća grupa funkcija koje su specijalno namenjene za manipulaciju podacima određenog tipa. Znakovne (string) funkcije se primenjuju na znakovne konstante i kolone znakovnog tipa, aritmetičke funkcije se upotrebljavaju kod numeričkih kolona i konstanti, dok se za manipulisanje konstantama i kolonama tipa DATE koriste takozvane datumske funkcije. Postoje još i takozvane grupne, kao i funkcije opšte namene (miscellaneous), od kojih se većina može primeniti na bilo koji od pomenutih tipova podataka. Neke od ovih funkcija modifikuju objekat na koji su primenjene (bilo da je reč o podacima tipa CHAR, VARCHAR2, NUMBER, ili DATE), dok druge samo saopštavaju određene informacije o tim objektima.

U izvesnom smislu, većinu funkcija kojima smo se do sada bavili mogli bismo svrstati u grupu transformacijskih funkcija, s obzirom da se pomoću njih vrši nekakva promena na određenom objektu. Ipak, funkcije o kojima ćemo govoriti u ovom poglavlju objekte menjaju na jedan neobičan način. One ih, naime, transformišu iz jednog tipa podatka u drugi, ili, pak, vrše temeljnu transformaciju podataka koji su u tim objektima smešteni. Sve funkcije iz ove kategorije ukratko su opisane u tabeli 11.1.

Tabela 11.1: Funkcije za konverziju

Naziv funkcije	Definicija
ASCIISTR	Znakovni string iz bilo kog znakovnog skupa, prevodi u ASCII string koji pripada definisanom znakovnom skupu konkretne baze podataka.
BIN_TO_NUM	Binarnu vrednost konvertuje u njen numerički ekvivalent.
CAST	Promoviše (casts) jedan ugrađeni (built-in) ili kolekcijski tip podataka u neki drugi; ova funkcija se, obično, primenjuje na umetnute (nested) tabele i promenljive matrice.
CHARTOROWID	Neki znakovni string pretvara u Oracleov interni identifikator reda, ili RowID.

Naziv funkcije	Definicija
COMPOSE	String bilo kog tipa prevodi u Unicode string potpuno normalizovanog oblika, koji pripada istom znakovnom skupu kao i ulazni (input) string.
CONVERT	Konvertuje znakovni string iz znakovnog skupa jednog nacionalnog jezika u drugi.
DECODE zavisno od	Dekodira podatke tipa CHAR, VARCHAR2, ili NUMBER i prevodi ih u jedan od nekoliko različitih znakovnih stringova ili u podatke numeričkog tipa, konkretne vrednosti ulaznog podatka. Ovo je veoma moćna funkcija tipa if-then-else, koja će biti detaljno objašnjena u Poglavlju 16.
DECOMPOSE	String bilo kog tipa prevodi u Unicode string nakon takozvane kanonske dekompozicije u istom znakovnom skupu kome pripada i ulazni string.
HEXTOROW	Znakovni string sačinjen od heksadecimalnih cifara prevodi u binarni broj.
NUMTODSINTERVAL OND.	Numerički podatak konvertuje u konstantu tipa INTERVAL DAY TO SEC-
NUMTOYMINTERVAL MONTH.	Numerički podatak konvertuje u konstantu tipa INTERVAL YEAR TO
RAWTOHEX	String binarnih cifara konvertuje u znakovni string heksadecimalnih cifara.
RAWTONHEX	"Sirovi" (raw) binarni podatak konvertuje u znakovnu vrednost tipa NVAR CHAR2, koji u sebi sadrži njegov heksadecimalni ekvivalent.
ROWIDTOCHAR	Interni Oracleov identifikator reda, ili RowID pretvara u znakovni string.
ROWIDTONCHAR	RowID vrednost konvertuje u podatak tipa NVARCHAR2.
SCN_TO_TIMESTAMP	Takozvani "broj promene sistema" (system change number) konvertuje u timestamp podatak približne vrednosti.
TIMESTAMP_TO_SCN	Timestamp podatak konvertuje u približan "broj promene sistema".
TO_BINARY_DOUBLE	Daje broj sa pokretnom decimalnom tačkom (floating-point), dvostruke tačnosti.
TO_BINARY_FLOAT	Daje broj sa pokretnom decimalnom tačkom (floating-point), jednostruke tačnosti.
TO_CHAR	Podatke tipa NUMBER ili DATE konvertuje u znakovne stringove.
TO_CLOB	NCLOB vrednosti iz kolone tipa LOB i druge vrste znakovnih stringova pretvara u vrednosti tipa CLOB.
TO_DATE	Podatke tipa NUMBER, CHAR, ili VARCHAR2 konvertuje u podatke datumskog tipa (DATE).
TO_DSINTERVAL	Znakovne stringove tipa CHAR, VARCHAR2, NCHAR, ili NVARCHAR2 konvertuje u podatke tipa INTERVAL DAY TO SECOND.
TO_LOB	Podatke tipa LONG konvertuje u tip LOB prilikom izvršavanja izjave insert as select.
TO_MULTI_BYTE	Sve jednobitne znakove iz nekog znakovnog stringa konvertuje u višebajtne znakove.
TO_NCHAR	Podatke znakovnog, numeričkog, ili datumskog tipa iz znakovnog skupa baze podataka konvertuje u nacionalni znakovni skup.
TO_NCLOB	CLOB vrednosti iz neke kolone tipa LOB i druge znakovne stringove konvertuje u odgovarajuće NCLOB vrednosti.
TO_NUMBER	Podatke tipa CHAR ili VARCHAR2 pretvara u numeričke podatke.

Naziv funkcije	Definicija
TO_SINGLE_BYTE	Sve višebajtne znakove iz podataka tipa CHAR ili VARCHAR2 prevodi u jednobajtne znakove.
TO_TIMESTAMP	Znakovne stringove konvertuje u odgovarajuće vrednosti tipa TIMESTAMP.
TO_TIMESTAMP_TZ	Znakovne stringove konvertuje u podatke tipa TIMESTAMP WITH TIME ZONE.
TO_YMINTERVAL	Znakovne stringove tipa CHAR, VARCHAR2, NCHAR ili NVARCHAR2 pretvara u podatke tipa INTERVAL YEAR TO MONTH.
TRANSLATE	Znakove iz nekog znakovnog stringa prevodi u neke druge znakove.
UNISTR	Određeni string pretvara u Unicode string iz Unicode znakovnog skupa konkretne baze podataka.

Elementarne funkcije za konverziju

Mada je u tabeli 11.1 prikazan veoma veliki broj postojećih funkcija za konverziju, svakako ćete najčešće koristiti svega tri funkcije iz ove kategorije, čija je namena da podatke jednog tipa konvertuju u neki drugi tip podataka:

- **TO_CHAR** Podatke tipa DATE ili NUMBER ova funkcija transformiše u znakovne stringove.
- **TO_DATE** Podatke tipa NUMBER, CHAR ili VARCHAR2 transformiše u podatke tipa DATE. Pri radu sa "timestamp" podacima treba koristiti funkcije TO_TIMESTAMP ili TO_TIMESTAMP_TZ.
- **TO_NUMBER** Podatke tipa CHAR ili VARCHAR2 transformiše u podatke tipa NUMBER.

Zbog čega su ove transformacije veoma važne? Očigledno je da je funkcija TO_DATE neophodna prilikom izvođenja računskih operacija iz oblasti datumske aritmetike. Funkcija TO_CHAR omogućava da numeričkim podacima manipulišete kao da je reč o znakovnim stringovima, odnosno da na njih primenjujete znakovne funkcije. Znakovne stringove koji se zbog nekog razloga sastoje isključivo od cifara, pomoću funkcije TO_NUMBER možete pretvoriti u numeričke podatke i, zatim, nad njima vršiti operacije sabiranja, oduzimanja, množenja, deljenja i slično.

To praktično znači da, na primer, neki devetocifreni poštanski broj (ZIP code), koji ste u tabelu uskladištili kao numerički podatak, možete transformisati u znakovni string, pa na njega primeniti funkciju SUBSTR i funkciju za spajanje (concatenation) stringova da biste mu u sredini dodali crticu (što je uobičajeni format za štampanje poštanskih brojeva na kovertama):

```
select SUBSTR(TO_CHAR(948033515),1,5)¦¦'-'¦¦
       SUBSTR(TO_CHAR(948033515),6) AS Zip
   from DUAL;

ZIP
----------------------------------------
94803-3515
```

U ovom primeru je čisto numerički podatak 948033515 (obratite pažnju na potpuno odsustvo jednostrukih navodnika, koji su obavezni kod podataka tipa CHAR ili VARCHAR2) pomoću funkcije **TO_CHAR** transformisan u znakovni string. Najpre su pomoću funkcije SUBSTR iz konkretnog broja izvučene i pretvorene u podstring sve cifre od prve do pete pozicije, to jest 94803. Nakon toga je sa desne strane ovog podstringa umetnuta jedna crtica (-), a zatim je pomoću funkcije **TO_CHAR** kreiran još jedan podstring, koji je upotrebom druge po redu funkcije **SUBSTR** izvučen iz konkretne grupe cifara, počev od šeste pozicije, pa do kraja. Ovaj drugi podstring - 3515 "zalepljen" je na već postojeći string sa desne strane umetnute crtice. Ovako rekonstruisani string je, na kraju, preimenovan u Zip, nakon čega ga je Oracle prikazao na ekranu kao: 94803-3515. Kao što vidite, funkcija TO_CHAR omogućava da funkcije za manipulaciju stringovima bez ikakvih problema primenite na podatke numeričkog (i datumskog) tipa, baš kao da je reč o najobičnijim znakovnim stringovima. Obratite pažnju na sledeći primer:

```
select SUBSTR(948033515,1,5)||'-'||
       SUBSTR(948033515,6) AS Zip
   from DUAL;

ZIP
----------------------------------------
94803-3515
```

Na osnovu onoga što je do sada rečeno, ovaj upit je neispravan, jer 948033515 predstavlja numerički podatak, a ne znakovni string. Pa, ipak, znakovna funkcija **SUBSTR** je očigledno dala očekivani razultat! Sada se postavlja pitanje da li bi se ova funkcija mogla primeniti i na neku kolonu znakovnog tipa? Evo jedne tabele u kojoj kolona Zip sadrži podatke tipa NUMBER, što se može ustanoviti iz njenog opisa:

```
describe ADDRESS

Name                              Null?     Type
--------------------------------  --------  ------------
LASTNAME                                    VARCHAR2(25)
FIRSTNAME                                   VARCHAR2(25)
STREET                                      VARCHAR2(50)
CITY                                        VARCHAR2(25)
STATE                                       CHAR(2)
ZIP                                         NUMBER
PHONE                                       VARCHAR2(12)
EXT                                         VARCHAR2(5)
```

Sada ćemo iz ove tabele selektovati samo poštanske brojeve za sve osobe po imenu Mary:

```
select SUBSTR(Zip,1,5)||'-'||
       SUBSTR(Zip,6) AS Zip
   from ADDRESS
  where FirstName = 'MARY';

ZIP
----------------------------------------
94941-4302
60126-2460
```

Vidimo, dakle, da **SUBSTR** funkcioniše potpuno ispravno, kao i kod doslovnih stringova, bez obzira što Zip predstavlja kolonu numeričkog tipa iz tabele ADDRESS. Da li se i ostale znakovne funkcije mogu na sličan način primeniti na numeričke podatke?

```
select Zip, RTRIM(Zip,20)
   from ADDRESS
  where FirstName = 'MARY';

ZIP RTRIM(ZIP,20)
----------  ----------------------------------------
949414302   9494143
601262460   60126246
```

Kolona na levoj strani predstavlja očigledan dokaz da je Zip kolona tipa NUMBER; naime, podaci su u njoj poravnati uz desnu ivicu, što je podrazumevani način poravnanja u kolonama numeričkog tipa. Nasuprot tome, u koloni koja je dobijena pomoću funkcije **RTRIM** podaci su

levo poravnati, što je karakteristično za kolone znakovnog tipa, pri čemu su sa desnog kraja poštanskih brojeva uklonjene eventualne nule i dvojke. Ovde sam, međutim, želeo da skrenem pažnju na nešto sasvim drugo. Ako se sećate, u Poglavlju 7 sam napomenuo da format funkcije **RTRIM** izgleda ovako:

```
RTRIM(string [, 'set'])
```

Parametar set, kojim se definiše podstring koji treba ukloniti sa desnog kraja postojećeg stringa, trebalo bi, dakle, da bude smešten između jednostrukih navodnika, ali u prethodnom primeru od navodnika nema ni traga:

```
RTRIM(Zip,20)
```

U čemu je, dakle, "kvaka"?

Automatska konverzija tipova podataka

Oracle je, kao što ste možda i sami pretpostavili, izvršio automatsku konverziju ovih objekata numeričkog tipa, odnosno kolone Zip i broja 20, u znakovne stringove, baš kao da ste na njih svojeručno primenili funkciju TO_CHAR. U stvari, uz svega nekoliko očiglednih izuzetaka, on će svaki tip podatka automatski transformisati u neki drugi tip podatka, ukoliko to zahteva konkretna funkcija koju ste na taj podatak primenili. Ako neku znakovnu funkciju primenite na podatak tipa NUMBER ili DATE, Oracle će taj podatak momentalno konvertovati u znakovni string i tako obezbediti uspešno izvršavanje te znakovne funkcije. Ako je, pak, reč o nekoj datumskoj funkciji, dok je kolona ili konstanta na koju je primenjujete znakovnog tipa sa formatom DD-MON-YY, Oracle će je automatski konvertovati u DATE tip podatka. Slično tome, primena neke aritmetičke funkcije na kolonu ili konstantu znakovnog tipa povlači za sobom automatsku konverziju ovih podataka u tip NUMBER da bi se moglo sprovesti željeno izračunavanje.

Da li ovo pravilo važi uvek i svuda? Ne. Da bi Oracle izvršio automatsku konverziju jednog tipa podatka u drugi, tip originalnog podatka mora "ličiti" na tip u koji bi ga, po logici stvari, trebalo konvertovati. Pri tom treba imati u vidu sledeća osnovna pravila:

- Svaki podatak tipa NUMBER ili DATE može biti konvertovan u znakovni string. Drugim rečima, bilo koju znakovnu funkciju možete primeniti na kolonu tipa NUMBER ili DATE. Kada nekoj znakovnoj funkciji prosleđujete konstantne brojeve, oni ne moraju biti uneti između jednostrukih navodnika, dok je kod konstantnih datumskih podataka to obavezno.
- Podatak tipa CHAR ili VARCHAR2 može biti konvertovan u odgovarajući podatak tipa NUMBER jedino pod uslovom da on u sebi sadrži isključivo cifre, decimalnu tačku i znak za minus, kojim se označavaju negativni brojevi.
- Podatak tipa CHAR ili VARCHAR2 može biti konvertovan u podatak tipa DATE pod uslovom da je ispisan podrazumevanim datumskim formatom (koji, obično, glasi: DD-MON-YY). Ovo pravilo važi za sve funkcije, osim za **GREATEST** i **LEAST**, koje sve prosleđene vrednosti tretiraju kao stringove, dok za funkciju BETWEEN pomenuto pravilo važi jedino pod uslovom da je prva kolona iza reči **BETWEEN** datumskog tipa. U suprotnom, neophodno je upotrebiti funkciju **TO-DATE**, uz prosleđivanje pravilno formatiranih parametara.

Mora se priznati da pomenuta pravila deluju prilično zbunjujuće, pa preporučujem da što češće primenjujete funkciju **TO_DATE** i druge funkcije za konverziju da biste uvek bili sigurni da će prosleđene vrednosti biti pravilno interpretirane. Da bih pomenuta pravila što bolje objasnio, u nastavku ću ih ilustrovati na primerima nekoliko slučajno odabranih znakovnih funkcija koje ću primeniti na podatke tipa NUMBER i DATE:

```
select INITCAP(LOWER(SysDate)) from DUAL;

INITCAP(LOWER(SYSDATE))
-----------------------
22-Mar-04
```

Primećujete da je funkcija **INITCAP** pretvorila početno slovo reči "mar" u veliko slovo, bez obzira što je ta reč "zakopana" duboko unutar stringa "22-mar-04". Ova karakteristika funkcije **INITCAP** nije ograničena samo na datumske podatke, mada je ovde prvi put ilustrovana. Da biste ovu karakteristiku što bolje razumeli, ilustrovaćemo je na još jednom primeru:

```
select INITCAP('this-is_a.test,of:punctuation;for+initcap')
    from DUAL;

INITCAP('THIS-IS_A.TEST,OF:PUNCTUATION;FO
-----------------------------------------
This-Is_A.Test,Of:Punctuation;For+Initcap
```

Kao što vidite, funkcija **INITCAP** je svaku reč iz ovog probnog stringa ispisala velikim slovom. Pri tom je početak svake reči odredila na osnovu činjenice da joj prethodi znak koji nije slovo. Datumske podatke možete, takođe, "seći" i "lepiti" (cut and paste) pomoću znakovnih funkcija, baš kao da je reč o običnim znakovnim stringovima:

```
select SUBSTR(SysDate,4,3) from DUAL;

SUB
---
MAR
```

NAPOMENA

Radi izvlačenja oznake za mesec iz nekog datumskog podatka (kao što je učinjeno u prethodnom primeru - prim. prev.), mogli biste upotrebiti i funkciju TO_CHAR ili EXTRACT.

U sledećem primeru podatku tipa DATE je sa leve strane pridodat određeni broj devetki da bi ukupna dužina podatka iznosila 20 znakova:

```
select LPAD(SysDate,20,'9') from DUAL;

LPAD(SYSDATE,20,'9')
--------------------
9999999999922-MAR-04
```

Funkciju **LPAD**, kao i sve ostale znakovne funkcije, možete bez ikakvih ograničenja primeniti na podatke tipa NUMBER, bez obzira da li je reč o konstantnim brojevima (kao što je to ilustrovano u sledećem primeru) ili o nekoj koloni numeričkog tipa:

```
select LPAD(9,20,0) from DUAL;

LPAD(9,20,0)
--------------------
00000000000000000009
```

Svi ovi primeri predstavljaju očigledan dokaz da znakovne funkcije tretiraju podatke tipa NUMBER i DATE na isti način kao i obične znakovne stringove. Krajnji rezultati ovih funkcija (to jest, ono što je prikazano na ekranu) sami po sebi takođe predstavljaju znakovne stringove.

U sledećem primeru jedan znakovni string (obratite pažnju na upotrebu jednostrukih navodnika) je od numeričke funkcije **FLOOR** tretiran kao podatak tipa NUMBER:

```
select FLOOR('-323.78') from DUAL;

FLOOR('-323.78')
----------------
-324
```

U narednom primeru dva znakovna stringa su automatski pretvorena u podatke tipa DATE za potrebe datumske funkcije **MONTHS_BETWEEN**. Ova automatska konverzija je omogućena činjenicom da su prosleđeni doslovni stringovi ispisani u podrazumevanom datumskom formatu DD-MON-YY:

```
select MONTHS_BETWEEN('16-MAY-04','01-NOV-04') from DUAL;

MONTHS_BETWEEN('16-MAY-02','01-NOV-02')
---------------------------------------
                             -5.516129
```

Jedno od ranije pomenutih pravila ukazuje da podaci tipa DATE ne mogu biti konvertovani u podatke tipa NUMBER. Pa, ipak, evo jednog praktičnog primera sabiranja i oduzimanja sa podacima datumskog tipa. Da li je to u suprotnosti sa pomenutim pravilom?

```
select SysDate, SysDate + 1, SysDate - 1 from DUAL;

SYSDATE   SYSDATE+1   SYSDATE-1
---------  ---------  ---------
22-MAR-04  23-MAR-04  21-MAR-04
```

Nije, jer ove operacije sabiranja i oduzimanja ne spadaju u oblast obične, već takozvane datumske aritmetike. Kao što je objašnjeno u Poglavlju 10, datumska aritmetika "poznaje" samo operacije sabiranja i oduzimanja, i to isključivo na podacima tipa DATE. Većina datumskih funkcija će znakovni string, koji je usklađen sa podrazumevnim datumskim formatom, automatski konvertovati u odgovarajući podatak tipa DATE. Jedini izuzetak se odnosi na pokušaj sabiranja doslovnog datumskog stringa sa nekim konstantnim brojem:

```
select '22-MAR-04' + 1 from DUAL;

ERROR: ORA-01722: invalid number
```

Čak i kada je reč o podacima pravog datumskog tipa, datumska aritmetika "poznaje" samo operacije sabiranja i oduzimanja. Svaki pokušaj primene bilo koje druge aritmetičke funkcije na podatke datumskog tipa dovešće do pojava greške, kao što je ilustrovano u sledećem primeru, gde sam jedan datumski podatak pokušao da podelim sa brojem 2:

```
select SysDate / 2 from DUAL;
          *
ERROR at line 1:
ORA-00932: inconsistent datatypes: expected NUMBER got DATE
```

Konačno, treba napomenuti da podatak tipa NUMBER nikada ne može biti konvertovan u podatak tipa DATE, zbog toga što "čist" broj nikada ne može imati podrazumevani datumski format, koji glasi DD-MON-YY:

```
select NEXT_DAY(032602,'FRIDAY') from DUAL;
                *
ERROR at line 1:
ORA-00932: inconsistent data types: expected DATE got NUMBER
```

Prema tome, da biste neki numerički podatak prosledili datumskoj funkciji, neophodno je da na njega prethodno primenite funkciju **TO_DATE**.

Jedno upozorenje u vezi automatske konverzije

Postoje jaki argumenti za i protiv upotrebe automatske konverzije tipova podataka u SQL-u. Sa jedne strane, ovaj metod značajno pojednostavljuje i smanjuje ukupan broj funkcija koje je neophodno upotrebiti u nekoj select izjavi. Sa druge strane, ukoliko se ispostavi da su Vaše pretpostavke koji će tipovi podataka biti smešteni u odgovarajućoj koloni pogrešne (na primer, ako pretpostavite da će određena kolona u sebi sadržati isključivo numeričke podatke, koje ćete moći bez problema da upotrebite u raznim proračunima), onda će Vaš upit u jednom trenutku prestati da funkcioniše, Oracle će na ekranu prikazati poruku o grešci, a Vi ćete morati da uložite određeno vreme i napor da otkrijete u čemu je problem. Nadalje, kada neko drugi bude čitao Vašu select izjavu, on će verovatno biti zbunjen primenom naizgled neodgovarajućih funkcija na znakovne ili numeričke podatke. Ako, međutim, upotrebite funkciju TO_NUMBER, time ćete eksplicitno staviti do znanja da očekujete isključivo podatke numeričkog tipa, bez obzira što konkretna kolona sadrži u sebi, recimo, podatke tipa VARCHAR2.

Stoga je možda najbolje pridržavati se "zlatnog pravila" da funkcije za automatsku konverziju treba koristiti tamo gde je rizik minimalan, kao, na primer, prilikom primene funkcija za manipulaciju stringovima na podatke numeričkog tipa, ali ne i prilikom izvođenja aritmetičkih operacija sa znakovnim stringovima. Za Vaše lično dobro i dobro onih koji će kasnije koristiti Vaše SQL izjave, potrudite se da pored svake select izjave u kojoj se vrši automatska konverzija tipova podataka stavite odgovarajući komentar.

Specijalizovane funkcije za konverziju

Kao što se može videti iz tabele 11.1, Oracle u sebi sadrži nekoliko funkcija koje su specijalno namenjene za konverziju podataka iz jednog tipa u drugi. Ukoliko planirate da SQL*Plus i Oracle koristite isključivo za kreiranje izveštaja, ove funkcije Vam verovatno nikada neće biti potrebne. Sa druge strane, ako SQL*Plus želite da upotrebite za ažuriranje (update) baze podataka, ako planirate da kreirate aplikacije u Oracleu, ili, pak, ako upotrebljavate opciju National Language Support, ispostaviće se da ćete od informacija koje ćete upoznati u ovom odeljku imati velike koristi. Detaljnije objašnjenje svake od funkcija iz ove kategorije možete pronaći i u Abecednom pregledu komandi i funkcija na kraju knjige.

NAPOMENA

Funkcija CAST se uglavnom koristi kod umetnutih tabela i promenljivih matrica, o čemu ćemo detaljnije govoriti u Poglavlju 34. Funkcija DECODE je detaljno obrađena u Poglavlju 16.

U opštem slučaju funkcije za konverziju prihvataju jednu jedinu vrednost kao ulazni parametar, dok kao izlazni rezultat takođe daju jednostruku i na odgovarajući način konvertovanu vrednost. Primera radi, pomoću funkcije **BIN_TO_NUM** možete neku binarnu vrednost pretvoriti u odgovarajući decimalni broj. Pri tom joj, kao ulaznu vrednost, možete proslediti listu binarnih cifara, međusobno razdvojenih zarezima, koja će biti tretirana kao jedan jedini ulazni string:

```
select BIN_TO_NUM(1,1,0,1) from DUAL;

BIN_TO_NUM(1,1,0,1)
-------------------
                 13
```

```
select BIN_TO_NUM(1,1,1,0) from DUAL;

 BIN_TO_NUM(1,1,1,0)
--------------------
                  14
```

Pri izvršavanju takozvanih flashback operacija (videti poglavlja 27 i 28) brojeve izmene sistema (system change number - SCN) možete konvertovati u odgovarajuće timestamp vrednosti pomoću funkcije **SCN_TO_TIMESTAMP**; analogno tome, funkcija **TIMESTAMP_TO_SCN** služi za određivanje SCN broja na osnovu timestamp podatka koji joj prosleđen.

Jedna od novina Oracle Database 10g softvera jeste da funkcije **TO_BINARY_DOUBLE** i **TO_BINARY_FLOAT** možete upotrebiti za konvertovanje različitih tipova podatka u brojeve sa pokretnom decimalnom tačkom, sa dvostrukom ili jednostrukom preciznošću, respektivno.

Funkcije za transformaciju

Mada se, u izvesnom smislu, sve funkcije koje vrše bilo kakvu izmenu u objektu na koji su primenjene mogu nazvati funkcije za transformaciju, postoje dve veoma neobične funkcije pomoću kojih možete na mnogo interesantnih načina kontrolisati izlazne rezultate na osnovu ulaznih parametara, umesto da ih jednostavno transformišete. To su funkcije **TRANSLATE** i **DECODE**.

TRANSLATE

TRANSLATE je jednostavna funkcija koja redom, znak po znak, vrši zamenu pojedinih znakova u zadatom stringu. Evo kako glasi njen format:

```
TRANSLATE(string,if,then)
```

Funkcija **TRANSLATE** redom analizira svaki pojedinačni znak u zadatom stringu da bi proverila da li se u njemu nalazi neki od znakova iz uslovnog stringa if. Kada u zadatom stringu pronađe neki znak iz stringa if, ona će interno zabeležiti lokaciju na kojoj se taj znak nalazi u stringu if, pa potražiti znak koji se nalazi na toj istoj lokaciji, ali u stringu then. Na taj način, funkcija **TRANSLATE** će sve one znakove iz zadatog stringa koji se poklapaju sa znakovima iz stringa if zameniti odgovarajućim znakovima iz stringa then. Mada se ova funkcija obično unosi u jednoj jedinoj programskoj liniji, na primer:

```
select TRANSLATE(7671234,234567890,'BCDEFGHIJ')
  from DUAL;

TRANSLA
-------
GFG1BCD
```

možda ćete je lakše razumeti ako je "razbijete" na dve programske linije, kao što je učinjeno u sledećem primeru (naravno, SQL nema ništa protiv primene ove tehnike):

```
select TRANSLATE(7671234,234567890,
                         'BCDEFGHIJ')
  from DUAL;

TRANSLA
-------
GFG1BCD
```

Kada funkcija **TRANSLATE** naiđe na cifru 7 u zadatom stringu, ona će istu tu cifru potražiti u stringu if i zatim će je zameniti znakom koji se nalazi na istoj toj poziciji u stringu then (u ovom konkretnom primeru to je veliko slovo G). Ukoliko se neki znak iz zadatog stringa ne sadrži u stringu if, funkcija **TRANSLATE** ga jednostavno neće ni prevesti (to se u ovom primeru odnosi na cifru 1).

Mada funkcija **TRANSLATE**, strogo tehnički gledano, predstavlja znakovnu funkciju, ona se, kao što vidite, zbog automatske konverzije podataka može primeniti na mešavinu znakovnih stringova i numeričkih podataka. Sledeći primer ilustruje veoma jednostavnu tehniku šifrovanja kod koje se svako slovo pomera za jednu poziciju unapred po abecedi. Pre mnogo godina špijuni su koristili slične metode zamene znakova da bi svoje poruke šifrovali pre nego što ih pošalju centrali. Primalac je, nakon prijema šifrovane poruke, trebalo samo da na nju primeni inverzni postupak. Sećate li se kompjutera sa nežnom bojom glasa, po imenu HAL, koji je bio jedan od glavnih "likova" u kultnom filmu 2001: Odiseja u svemiru? Ako pomoću funkcije **TRANSLATE** na HAL-ovo ime primenite tehniku pomeranja za jedno slovo unapred po abecedi, doći ćete do skrivene reklamne poruke:

```
select TRANSLATE('HAL','ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                       'BCDEFGHIJKLMNOPQRSTUVWXYZA') AS Who
from DUAL;

WHO
---
IBM
```

Radi detaljnijih informacija o mogućim načinima manipulacije stringovima pomoću regularnih izraza u programu Oracle Database 10g, još jednom pročitajte objašnjenje funkcije **REGEXP_REPLACE** u Poglavlju 8.

DECODE

Ako funkcija **TRANSLATE** vrši promenu konkretnog stringa znak po znak, **DECODE** se može posmatrati kao funkcija koja izmene u konkretnom stringu vrši vrednost po vrednost. Naime, svaku vrednost na koju naiđe u određenoj koloni funkcija **DECODE** zamenjuje odgovarajućom vrednošću, u skladu sa čitavim nizom if-then kriterijuma koje ste joj prosledili. **DECODE** je veoma moćna funkcija, koja se može korisno upotrebiti u širokom spektru različitih oblasti. Poglavlje 16 je u potpunosti posvećeno naprednoj upotrebi funkcija **DECODE** i **CASE**.

Funkcija **DECODE** ima sledeći format:

```
DECODE(value,if1,then1,if2,then2,if3,then3,…,else)
```

Broj *if-then* kombinacija je praktično neograničen; ovde su ilustrovane samo tri. Da biste što bolje shvatili na koji način DECODE funkcioniše, prisetite se sadržaja tabele NEWSPAPER koji ste upoznali u jednom od prethodnih poglavlja:

```
select * from NEWSPAPER;

FEATURE            S          PAGE
---------------    -          ----------
National News      A          1
Sports             D          1
Editorials         A          12
Business           E          1
Weather            C          2
Television         B          7
Births             F          7
Classified         F          8
Modern Life        B          1
Comics             C          4
```

```
Movies            B          4
Bridge            B          2
Obituaries        F          6
Doctor Is In      F          6
```

U narednom primeru šifrovaćemo brojeve stranica iz ove tabele. Umesto broja stranice 1, umetnućemo reči "Front Page". Za sve ostale stranice ispred rednog broja stranice biće umetnute reči "Turn to". Iz ovog primera se dobro vidi da parametar else može predstavljati neku funkciju, konstantnu vrednost ili naziv neke druge kolone.

```
select Feature, Section,
       DECODE(Page,'1','Front Page','Turn to '||Page)
    from NEWSPAPER;

FEATURE        S   DECODE(PAGE,'1','FRONTPAGE','TURNTO'||PAGE)
---------------    -              ----------------------------------------
--------
National News  A   Front Page
Sports         D   Front Page
Editorials     A   Turn to 12
Business       E   Front Page
Weather        C   Turn to 2
Television     B   Turn to 7
Births         F   Turn to 7
Classified     F   Turn to 8
Modern Life    B   Front Page
Comics         C   Turn to 4
Movies         B   Turn to 4
Bridge         B   Turn to 2
Obituaries     F   Turn to 6
Doctor Is In   F   Turn to 6
```

Postoje, doduše, izvesna ograničenja kada je reč o tipu podataka koje možete upotrebiti u listi if-then klauzula; o njima će biti više reči u Poglavlju 16.

Zaključak

Iako su sve funkcije u Oracleu prvenstveno namenjene za upotrebu na jednom jedinom tipu podataka (CHAR, VARCHAR2 ,NUMBER ili DATE), većina se u praksi može primeniti za više različitih tipova podataka. Zasluge za to pripadaju automatskoj konverziji tipa podataka. Uz nekoliko sasvim logičnih izuzetaka i nadu da će buduće verzije Oraclea u ovome biti kompatibilne sa Oracle Database 10g, ova automatska konverzija će biti moguća sve dok podaci koje treba konvertovati u dovoljnoj meri "liče" na tip podatka koji konkretna funkcija zahteva.

Znakovne funkcije će automatski izvršiti odgovarajuću konverziju podataka tipa NUMBER ili DATE. Numeričke funkcije mogu automatski konvertovati podatke tipa CHAR ili VARCHAR2, pod uslovom da oni u sebi sadrže samo cifre od 0 do 9, decimalnu tačku i, eventualno, znak minus na početku stringa. Numeričke funkcije ne mogu konvertovati podatke datumskog tipa. Sa druge strane, datumske funkcije mogu konvertovati znakovne stringove ukoliko su oni usklađeni sa podrazumevanim datumskim formatom DD-MON-YY, ali ne mogu vršiti konverziju "čisto" numeričkih podataka.

Dve specijalne funkcije **TRANSLATE** i **DECODE** rezultiraju u temeljnoj izmeni podataka na koje su primenjene. Funkcija **TRANSLATE** vrši zamenu pojedinačnih znakova konkretnog stringa u skladu sa definisanim uzorkom, dok se pomoću funkcije DECODE vrši zamena čitave vrednosti podatka, opet u skladu sa definisanim uzorkom (pattern).