Adattípusok

Adattípusok

- Tartomány, műveletek, reprezentáció
- Altípus: azonos műveletek és reprezentáció, szűkebb tartomány
- Osztályozás:
  - előre definiált vagy felhasználó által definiált
  - skalár-, összetett, referencia-, LOB- és objektumtípusok

Előre definiált skalártípusok

- Numerikus család
- Karakteres család
- Logikai család
- Dátum/intervallum család

Numerikus típus

- Alaptípusok:
  - PLS_INTEGER, BINARY_INTEGER
    - 32 bites előjeles egész
    - -2 147 483 648 – 2 147 483 647
  - BINARY_FLOAT
    - egyszeres pontosságú IEEE 754 lebegőpontos
  - BINARY_DOUBLE
    - dupla pontosságú IEEE 754 lebegőpontos
  - NUMBER
    - 1E-130 és 1E126 közötti fixpontos vagy lebegőpontos

NUMBER[(p[,s])]

NUMBER: legfeljebb 38 jegyű lebegőpontos

- Ha megadjuk a pontosságot és a skálát, fixpontos lesz: NUMBER(4), NUMBER(8,2)

Típus	*Kezelendő érték*	*Tárolt érték*
NUMBER	1111.2222	111.2222
NUMBER(3)	321	321
NUMBER(3)	3210	Túlcsordulás
NUMBER(4,3)	33222	Túlcsordulás
NUMBER(4,3)	3.23455	3.235
NUMBER(3,–3)	1234	1000
NUMBER(3,–1)	1234	1230

NUMBER

- Altípusai:
  - DEC, DECIMAL, NUMERIC (nemkorlátozott, fixpontos)
  - DOUBLE PRECISION, FLOAT (nem korlátozott, lebegőpontos)
  - INT, INTEGER, SMALLINT (korlátozott, fixpontos)
  - REAL (korlátozott, lebegőpontos)

PLS_INTEGER, BINARY_INTEGER

- Előnye: kisebb helyet foglal, mint a NUMBER, és hardveraritmetikát használ
  - Túlcsordulás esetén:

    ORA-01426: numeric overflow
- Altípusai:
  - NATURAL, NATURALN: természetes (+NOT NULL)
  - POSITIVE, POSITIVEN: pozitív (+ NOT NULL)
  - SIGNTYPE: előjel; -1,0,1
  - SIMPLE_INTEGER: azonos tartomány, de NOT NULL, és eltérő túlcsordulási szemantika (nincs kivétel)

BINARY_FLOAT, BINARY_DOUBLE

- Nem dobnak kivételt alul- ill. túlcsorduláskor, hanem konstansokkal kezelik a speciális helyzeteket
  - BINARY_DOUBLE_NAN
  - BINARY_DOUBLE_INFINITY
  - BINARY_DOUBLE_MAX_NORMAL
  - BINARY_DOUBLE_MIN_NORMAL
  - BINARY_DOUBLE_MAX_SUBNORMAL
  - BINARY_DOUBLE_MIN_SUBNORMAL

Karakteres típusok

- CHAR
- VARCHAR2
- RAW
- NCHAR
- NVARCHAR2
- LONG
- LONG RAW
- ROWID
- UROWID

Logikai típus

- BOOLEAN
- Háromértékű logika: TRUE, FALSE, NULL

Dátum/intervallum típusok

- Alaptípusok:
  - DATE
    - 7 bájton évszázad, év, hónap, nap, óra, perc, mp
  - TIMESTAMP[(p)]
    - időbélyeg, törtmásodpercek is (alapértelmezés: p=6)
  - INTERVAL
    - két időbélyeg közötti intervallum
    - YEAR[(p)] TO MONTH
      - alapértelmezés: p=2
    - DAY[(np)] TO SECOND[(mp)]
      - alapértelmezés: np=2, mp=6

LOB típusok

- Belső LOB-ok
  - BLOB
  - CLOB
  - NCLOB
- Külső LOB-ok
  - BFILE
- Kezelni őket a DBMS_LOB csomaggal lehet

Összetett típusok

- Rekordtípus
- Kollekciótípus
  - asszociatív tömb
  - beágyazott tábla
  - dinamikus tömb

Nevesített konstans és változó

- Változó

név típus `[[NOT NULL] {:= | DEFAULT}` *kifejezés*`];`

Pl:

```
szuletesi_ido DATE;
dolgozok_szama NUMBER NOT NULL DEFAULT 0;
```

- Nevesített konstans

név `CONSTANT` *típus* `[NOT NULL] {:= | DEFAULT}` *kifejezés;*

Pl:

```
max_napok_szama CONSTANT NUMBER := 366;
```

A deklarációban a típus az alábbiak egyike lehet 1.:

- *skalár_adattípus_név*
- *változónév*%TYPE
- *rekordtípus_név*
- *rekordnév*%TYPE
- *adatbázis_tábla_név.oszlopnév*%TYPE
- *adatbázis_tábla_név* %ROWTYPE
- *kurzornév*%ROWTYPE
- *kurzorváltozó_név*%TYPE
- *kurzorreferenciatípus_név*

A deklarációban a típus az alábbiak egyike lehet 2.:

- *kollekciótípus_név*
- *kollekciónév*%TYPE
- *objektumnév*%TYPE
- [REF] *objektumtípus_név*

Példák 1.

```
v_Telszam    ugyfel.tel_szam%TYPE
  DEFAULT '06-52-123456';
 v_Telszam2   v_Telszam%TYPE;

v_Ugyfel     ugyfel%ROWTYPE;
v_Ugyfel2    v_ugyfel%TYPE;
```

Példák 2.

```
DECLARE
v_Szam1         NUMBER NOT NULL := 10;
v_Szam2         NUMBER;
BEGIN
  v_Szam1 := v_Szam2;
END;
/
```

Példák 3.

```
c_Most          CONSTANT DATE := SYSDATE;

v_Egeszszam     PLS_INTEGER;
v_Logikai       BOOLEAN;

v_Pozitiv       POSITIVEN DEFAULT 1;
v_Idopecset     TIMESTAMP := CURRENT_TIMESTAMP;
```

Összetett típusok – Rekordtípus

- Rekordtípus deklarációja:

```
TYPE név IS RECORD (
      mezőnév típus [[NOT NULL] {:=|DEFAULT} kifejezés]
[,    mezőnév típus [[NOT NULL] {:=|DEFAULT} kifejezés]]…
);
```

- Rekord deklarációja:

```
rekordnév rekordtípusnév;
```

- Hivatkozás a rekord mezőjére:

```
rekordnév.mezőnév;
```

Összetett típusok – Rekordtípus

- Például:

```
TYPE t_alk_rec IS RECORD (
  nev     VARCHAR2(46),
  fizetes NUMBER(8,2),
  email   VARCHAR2(25) NOT NULL :=
  'a@b');
v_fonok t_alk_rec;
```

Hivatkozás:

```
v_fonok.nev
```

```
DECLARE
TYPE t_aru_tetel IS RECORD (
    kod         NUMBER NOT NULL := 0,
    nev         VARCHAR2(20),
    afa_kulcs   NUMBER := 0.25);

TYPE t_aru_bejegyzes IS RECORD (
    kod         NUMBER NOT NULL := 0,
    nev         VARCHAR2(20),
    afa_kulcs   NUMBER := 0.25);

  v_Tetel1      t_aru_tetel; -- Itt nem lehet inicializálás és NOT NULL!
  v_Tetel2      t_aru_tetel;

  v_Bejegyzes   t_aru_bejegyzes;
  v_Kod         v_Tetel1.kod%TYPE := 10;

BEGIN
  v_Tetel1.kod := 1;
  v_Tetel1.nev := 'alma';
  v_Tetel1.nev := INITCAP(v_Tetel1.nev);
   v_Tetel2 := v_Tetel1;
  -- v_Bejegyzes := v_Tetel1; -- Hibás értékadás
 END;
/
```

```
DECLARE
TYPE t_kolcsonzes_rec IS RECORD (
    bejegyzes   plsql.kolcsonzes%ROWTYPE,
    kolcsonzo   plsql.ugyfel%ROWTYPE,
    konyv       plsql.konyv%ROWTYPE
  );

v_Kolcsonzes  t_kolcsonzes_rec;

FUNCTION fv(p_Kolcsonzo v_Kolcsonzes.kolcsonzo.id%TYPE, p_Konyv
v_Kolcsonzes.konyv.id%TYPE) RETURN t_kolcsonzes_rec IS
    v_Vissza    t_kolcsonzes_rec;
  BEGIN
    v_Vissza.kolcsonzo.id  := p_Kolcsonzo;
    v_Vissza.konyv.id      := p_Konyv;
    RETURN v_Vissza;
  END;

BEGIN
v_Kolcsonzes.konyv.id := fv(10, 15).konyv.id;
END;
/
```

Felhasználói altípusok

- Egy *alaptípus* tartományának megszorításával jöhet létre
  - vannak beépített altípusok (STANDARD csomag)
    - SUBTYPE CHARACTER IS CHAR;
    - SUBTYPE INTEGER IS NUMBER(38,0);
- Felhasználói altípusok létrehozása: altípusdeklarációval
  - Blokk, alprogram vagy csomag deklarációs részében

```
SUBTYPE altípus_név IS alaptípus_név[(korlát)]
  [NOT NULL];
```

`korlát:` hossz, pontosság, skála információk

Példák altípus használatára

```
SUBTYPE t_szam   IS NUMBER;
SUBTYPE t_evszam IS NUMBER(4,0);

SUBTYPE t_nev  IS VARCHAR2(40);
SUBTYPE t_nev2 IS t_nev(50); -- felüldefiniáljuk az előző
    hosszmegszorítást
```

Példák altípus használatára

```
SUBTYPE t_tiz   IS NUMBER(10);
SUBTYPE t_egy   IS t_tiz(1);      -- NUMBER(1)
SUBTYPE t_ezres IS t_tiz(10, -3); -- NUMBER(10, -3)
v_Tiz   t_tiz := 12345678901; -- túl nagy az érték
    pontossága
v_Egy   t_egy := 12345;       -- túl nagy az érték pontossága
v_Ezres t_ezres := 12345;     -- 12000 tárolódik

SUBTYPE t_szo IS VARCHAR2;
v_szo   t_szo(10);
v_szo2  t_szo; -- Hiba, nincs megadva a hossz
```

```
CREATE TABLE tab1 (id NUMBER NOT NULL, nev
   VARCHAR2(40)); /


DECLARE
   SUBTYPE t_tabsor IS tab1%ROWTYPE;
   SUBTYPE t_nev IS tab1.nev%TYPE NOT NULL;
   SUBTYPE t_id IS tab1.id%TYPE;
   SUBTYPE t_nev2 IS t_nev%TYPE;
   v_Id    t_id;
   v_Nev1  t_nev;
   v_Nev2  t_nev := 'XY';
 BEGIN
   v_Id   := NULL;
   v_Nev2 := NULL;
END;
/
```

- Kivéve a BINARY_INTEGER/PLS_INTEGER esetén, ahol korlátozott altípust is létre lehet hozni

```
DECLARE
   SUBTYPE t_gyak_letszam IS
                  BINARY_INTEGER RANGE 5..20;
  ...
  v_gyl  t_gyak_letszam;
  ...
BEGIN
  v_gyl := 19; -- OK
  v_gyl := 3;
-- ORA-06502: PL/SQL: numerikus- vagy értékhiba
  ...
```

Típuskonverziók

- Implicit
  - a skalártípusok családjai között
- Explicit
  - beépített fügvényekkel: TO_DATE, TO_NUMBER, TO_CHAR

From:	To:									
	BLOB	CHAR	CLOB	DATE	LONG	NUMBER	PLS_INTEGER	RAW	UROWID	VARCHAR2
BLOB								Yes		
CHAR			Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CLOB		Yes								Yes
DATE		Yes			Yes					Yes
LONG		Yes						Yes		Yes
NUMBER		Yes			Yes		Yes			Yes
PLS_INTEGER		Yes			Yes	Yes				Yes
RAW	Yes	Yes			Yes					Yes
UROWID		Yes								Yes
VARCHAR2		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

Explicit konverziós függvények

ASCIISTR
BIN_TO_NUM
CAST
CHARTOROWID
COMPOSE
CONVERT
DECOMPOSE
HEXTORAW
NUMTODSINTERVAL
NUMTOYMINTERVAL
RAWTOHEX
RAWTONHEX
ROWIDTOCHAR
ROWIDTONCHAR
SCN_TO_TIMESTAMP
TIMESTAMP_TO_SCN
TO_BINARY_DOUBLE
TO_BINARY_FLOAT

TO_CHAR (character)
TO_CHAR (datetime)
TO_CHAR (number)
TO_CLOB
TO_DATE
TO_DSINTERVAL
TO_LOB
TO_MULTI_BYTE
TO_NCHAR (character)
TO_NCHAR (datetime)
TO_NCHAR (number)
TO_NCLOB
TO_NUMBER
TO_DSINTERVAL
TO_SINGLE_BYTE
TO_TIMESTAMP
TO_TIMESTAMP_TZ
TO_YMINTERVAL
TO_YMINTERVAL
TRANSLATE ... USING
UNISTR

Példa

```
DECLARE
   start_time CHAR(5);
   finish_time CHAR(5);
   elapsed_time NUMBER(5);
BEGIN
   SELECT TO_CHAR(SYSDATE,'SSSSS')
     INTO start_time
     FROM sys.DUAL;
   -- Feldolgozás
   SELECT TO_CHAR(SYSDATE,'SSSSS')
     INTO finish_time
     FROM sys.DUAL;
   elapsed_time := finish_time - start_time;
   DBMS_OUTPUT.PUT_LINE ('Elapsed time: '
                  || TO_CHAR(elapsed_time));
END;
/
```