Database Programming with SQL
kurs 2017 – database design and programming with sql
students slajdovi

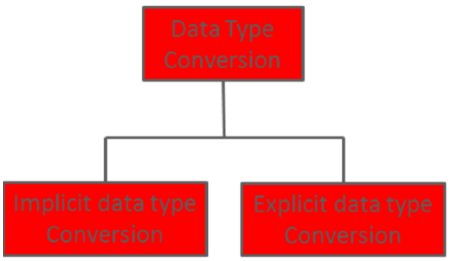
5-1 Conversion Functions

- U db formatiranje i promene izgleda se izvode pomoću funkcija konverzija
- Ove funkcije su sposobne za prikaz brojeva kao lokalne valute, prikaz datuma u različitim formatima, prikaz vremena u sekundama, i vođenje evidencije u kojem veku je datum

Data Types

- Kada je tabela kreirana za db, SQL programer mora definisati koja vrsta podataka će biti smeštena u svakom od polja tabele
- U SQL postoji nekoliko tipova podataka; ovi tipovi definišu domen vrednosti koji svaka kolona može sadržavati; npr, VARCHAR2, CHAR, NUMBER, DATE
- VARCHAR2: koristi se za znakove podataka promenjive dužine, uključujući brojeve, crte i specijalne znakove
- CHAR: koristi se za tekst i znakovne podatke fiksne dužine, uključujući brojeve, crte i specijalne znakove
- NUMBER: koristi se za brojeve promenjive dužine. Bez crta, teksta ili drugih nenumeričkih podataka. Valuta je smeštena kao brojevni tip podataka
- DATE: koristi se za datume i vremenske vrednosti. Interno, Oracle smešta datume kao brojeve i po difoltu, DATE informacija je prikazana kao DD-Mon-YYYY (23-Oct-2013)

Type Conversion

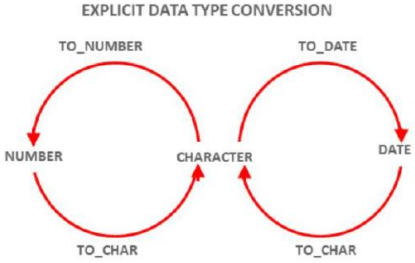


- Oracle Server može automatski konvertovati VARCHAR2 i CHAR podatak u NUMBER i DATE tip podataka
- Može konvertovati NUMBER i DATE podatak nazad u CHARACTER tip podatka
- Ovo se naziva implicitna konverzija podataka
- Iako je ovo interesantna mogućnost, uvek je bolje uraditi eksplicitnu konverziju tipa da bi se osigurala preciznost u SQL iskazima

Implicit data type conversions

FROM	TO
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

- Ovde će se raditi na funkcijama konverzija: konverzija tipa datuma u znak tip i obrnuto, konverzija broja u karakter tip i obrnuto



Date Conversion to Character Data

- Često je potrebno konvertovati datum iz difolt formata DD-Mon-YYYY u drugi specificirani format
- Funkcija koja to omogućava:
  TO_CHAR (date column name, 'format model you specify')
- 'format model' mora biti u apostrofima i osetljiv na veličinu slova; mora postojati zarez posle imena kolone, mogu se uključiti bilo kakvi validni formati podataka
- Koristiti sp za spell out a number
- Koristiti th da bi se broj prikazao kao redni broj (1st, 2nd, 3rd...)
- Koristiti fm element za odbacivanje padded praznih mesta ili odstranjivanje vodećih nula iz izlaza
- Tabela pokazuje modele različitih formata koji se mogu koristiti
- Kada se specificiraju elementi vremena, videti da sati (HH), minuti (MI), sekundi (SS) i AM ili PM se mogu takođe formatirati
- Primeri izlaza koji koriste modele različitih formata:

YYYY	Full year in numbers
YEAR	Year spelled out
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month
DDspth	FOURTEENTH
Ddspth	Fourteenth
ddspth	fourteenth
DDD or DD or D	Day of year, month or week
HH24:MI:SS AM	15:45:32 PM
DD "of" MONTH	12 of October

Examples:	Output
`SELECT TO_CHAR(hire_date, 'Month dd, YYYY') FROM employees;`	... June 07, 1994 ...
`SELECT TO_CHAR(hire_date, 'fmMonth dd, YYYY') FROM employees;`	... June 7, 1994 ...
`SELECT TO_CHAR(hire_date, 'fmMonth ddth, YYYY') FROM employees;`	June 7th, 1994 January 3rd, 1990 ...

- Primer 2 koristi "fm" za sprečavanje vodećih nula u datumu
- Primer 3 dodaje "th" itd na prikaz dana kao rednog broja
- Primer izlaza koristi modele različitih formata:

Examples:	Output
`SELECT TO_CHAR(hire_date, 'fmDay ddth Mon, YYYY') FROM employees;`	Tuesday 7th Jun, 1994
`SELECT TO_CHAR(hire_date, 'fmDay ddthsp Mon, YYYY') FROM employees;`	Tuesday, seventh Jun, 1994
`SELECT TO_CHAR(hire_date, 'fmDay, ddthsp "of" Month, Year') FROM employees;`	Tuesday, seventh of June, Nineteen Ninety-Four

- Primer 2 dodaje "SP" na format dana za prikaz rednog broja dana

- Primer 3 prikazuje godinu u rečima i tekst “of“ između dana i meseca; primetiti da navodnici su neophodni oko tekst literala
- Primeri izlaza koji koriste različite modele formata za vreme:

Examples:	Output
`SELECT TO_CHAR(SYSDATE, 'hh:mm') FROM dual;`	02:07
`SELECT TO_CHAR(SYSDATE, 'hh:mm pm') FROM dual;`	02:07 am
`SELECT TO_CHAR(SYSDATE, 'hh:mm:ss pm') FROM dual;`	02:07:23 am

Number Conversion to Character Data (VARCHAR2)

- Brojevi smešteni u db nemaju format; to znači danemaju valutni znak/simbol, zareze, decimale ili druge formate
- Za dodavanje formata, prvo treba konvertovati broj u format znaka
- SQL funkcija koja koristi konverziju broja u željeni formnat znaka je:

```
TO_CHAR(number, 'format model')
```

- Tabela ilustruje neke od elemenata formata koji su nam ponuđeni sa TO_CHAR funkcijom

ELEMENT	DESCRIPTION	EXAMPLE	RESULT
9	Numeric position (# of 9's determine width)	999999	1234
0	Display leading zeros	099999	001234
$	Floating dollar sign	$999999	$1234
L	Floating local currency symbol	L999999	FF1234
.	Decimal point in position specified	999999.99	1234.00
,	Comma in position specified	999,999	1,234
MI	Minus signs to right (negative values)	999999MI	1234-
PR	Parenthesize negative numbers	999999PR	<1234>
EEEE	Scientific notation (must have four EEEE)	99.999EEEE	1,23E+03
V	Multiply by 10 n times (n= number of 9's after V)	9999V99	9999V99
B	Display zero values as blank, not 0	B9999.99	1234.00

```
SELECT TO_CHAR(salary,
  '$99,999') AS "Salary"
FROM employees;
```

Salary
$24,000
$17,000

- Ako je broj 9-ki u modelu formata manji od broja cifara u broju, ##### je prikazan, npr SELECT TO_CHAR(2400, ’$9,999’) će vratiti ####### pošto model formata ima četiri “9“-ke a dati broj ima pet cifara
- Koji su modeli formata korišćeni za kreiranje sledećih izlaza ?
  $3000.00; 4,500; 9,000.00; 0004422

SQL:	Output
`SELECT TO_CHAR(3000, '$99999.99') FROM dual;`	$3000.00
`SELECT TO_CHAR(4500, '99,999') FROM dual;`	4,500
`SELECT TO_CHAR(9000, '99,999.99') FROM dual;`	9,000.00
`SELECT TO_CHAR(4422, '0009999') FROM dual;`	0004422

Character Conversion to Number

- Često je poželjno konvertovati string karaktera u broj. Funkcija koja izvodi konverziju:

```
TO_NUMBER(character string, 'format model')
```

- Model formata je opcionalan, ali treba da se uključi ako string karaktera koji se konvertuje sadrži bilo koje karaktere drugačije od brojeva
- Ne može se pouzdano izvesti kalkulacija sa podacima karakterima

```
SELECT TO_NUMBER('5,320', '9,999')
AS "Number"
FROM dual;
```

Number
5320

- Bonus kolona uključuje podatke koji sadrže 4 karaktera, model formata specificira 3 karaktera, tako da se vraća greška

```
SELECT last_name, TO_NUMBER(bonus, '999')
FROM employees
WHERE department_id = 80;
```


```
SELECT last_name, TO_NUMBER(bonus, '9999')
 AS "Bonus"
FROM employees
WHERE department_id = 80;
```

LAST_NAME	Bonus
Zlotkey	1500
Abel	1700
Taylor	1250

- APEX će vratiti Oracle Error – invalid number – ako model formata se ne poklapa sa brojem vraćenim od strane db

Character Conversion to Date

- Za konverziju stringa karaktera u format datuma, koristi se:

```
TO_DATE('character string', 'format model')
```

- Ova konverzija uzima ne-datum vrednost string karakter kao “November 3, 2001“ i konvertuje ga u vrednost datum
- Model format kaže serveru kako string karakter izgleda:

```
TO_DATE('November 3, 2001', 'Month dd, yyyy')
```

i vraća 03-Nov-2001

- Pri konverziji karakter u datum, fx (format exact) modifikator specificira tačno podudaranje karakter argumenta i datum model format
- U sledećem primeru, primetiti da “May10“ nema prostor između “May“ i “10“
- Format model fx se poklapa sa karakter argumentom pošto nema prostor između “Mon“ i “DD“

```
SELECT TO_DATE('May10,1989', 'fxMonDD,YYYY') AS "Convert"
FROM DUAL;
```

CONVERT
10-May-1989

fx Modifier Rules

- Modifikator fx ima pravila:
  - punktuacija i tekst sa navodnicima u karakter argumentu mora da se precizno poklopi sa odgovarajućim delovima modela formata (osim sa case)
  - argument karakter ne sme imati dodatna prazna mesta (bez fx, Oracle Server ignoriše dodatna prazna mesta)

- broječani podaci u argumentu karaktera mora imati isti broj cifara kao odgovarajući element u modelu formata (bez fx, broj u karakter argumentu može izbeći vodeće nule)

Examples:	Output
`SELECT TO_DATE('Sep 07, 1965', 'fxMon dd, YYYY') AS "Date" FROM dual;`	07-Sep-1965
`SELECT TO_DATE('July312004', 'fxMonthDDYYYY') AS "Date" FROM DUAL;`	31-Jul-2004
`SELECT TO_DATE('June 19, 1990','fxMonth dd, YYYY') AS "Date" FROM DUAL;`	19-Jun-1990

RR Date Format and YY Date Forma

- Svi podaci bi trebalo da su smešteni korišćenjem četvorocifarske godine (YYYY)
- Ipak postoje neke stare db koje i dalje koriste dvocifarski (YY) format
- Zato nije jasno da li 02-Jan-98 znači Januar 2, 1998 ili 2098
- Ako je podatak konvertovan iz znak podatka u datum podatak i sadrži samo dvocifarsku godinu, Oracle ima način interpretacije ovih datuma u tačan vek
- Npr, '27-Oct-95'

```
SELECT TO_DATE('27-Oct-95','DD-Mon-YY')
 AS "Date"
FROM dual;
```

Date
27-Oct-2095

- Dvocifarska godina se interpretira kao 2095 a to možda i nije način na koji se želelo pročitati godina
- Ako YY se koristi u modelu formata, godina se pretpostavlja da bude u trenutnom veku
- Ako dvocifarska godina nije u trenutnom veku, koristi se RR

```
SELECT TO_DATE('27-Oct-95','DD-Mon-RR')
 AS "Date"
FROM dual;
```

Date
27-Oct-1995

- Dvocofarska godina se sada interpretira kao 1995

A Few Simple Rules

- Ako je specificiran format datuma sa RR formatom, povratna vrednost ima dve mogućnosti, u zavisnosti od trenutne godine
- Ako je trenutna godina između 00-49: datum od 0-49 i datum je u trenutnom veku; datum od 50-99: datum je u prethodnom veku
- Ako trenutna godina je između 50-99: datum od 0-49: datum će biti u sledećem veku; ako je datum od 50-99: datum će biti u trenutnom veku

		If the specified two-digit year is:	
		0-49	50-99
If two digits of the current year are:	0-49	The return date is in the current century	The return date is in the century before the current one
	50-99	The return date is in the century after the current one	The return date is in the current century

- Tabela ispod daje neke primere kako YY i RR se interpretiraju, u zavisnosti od trenutne godine:

Current Year	Specified Date	RR Format	YY Format
1995	27-Oct-95	1995	1995
1995	27-Oct-17	2017	1917
2015	27-Oct-17	2017	2017
2015	27-Oct-95	1995	2095

- Kada dajem upit nad zaposleni db korišćenjem sledećih iskaza, ovaj vraća svaki red iz tabele
- Znam da postoje samo nekoliko zaposlenih koji su unajmljeni pre 1990

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YY')
FROM employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-YY');
```

- Pošto model formata u WHERE rečenici koristi YY, a tekuća godina je 2015, upit daje redove sa hire_date manje od 2090
- Može se popraviti zamenom YY u modelu formata sa RR, ili korišćenjem četvorocifarskog prikaza godine YYYY

5-2 NULL Functions

- Osim funkcija koje kontrolišu kako su podaci formatirani ili konvertovani u drugi tip, SQL korisati set opštih funkcija dizajniranih posebno za rad sa null vrednostima
- Null može biti ništa ali ono utiče kako se izrazi evaluiraju, kako se izračunavaju srednje vrednosti, i gde se vrednosti pojavljuju u sortiranoj listi

How Functions are Evalueted

- Do sada su korišćene singe-row funkcije u jednostavnim iskazima; ipak je moguće nestovati funkcije do bilo koje dubine; važno je znati kako se nestovane funkcije evaluiraju
- “Nesting“ se odnosi na to da je jedna stvar unutar druge; sledeći primer je nestovana (ugnježdena) funkcija; proces evaluacije započinje od najdubljeg do najšireg nivoa

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'), 'fmDay,
  Month ddth, YYYY') AS "Next Evaluation"
FROM employees
WHERE employee_id = 100;
```

- Rezultat je: Friday, December 18th, 1987

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'), 'fmDay,
  Month ddth, YYYY') AS "Next Evaluation"
FROM employees
WHERE employee_id = 100;
```

- Korak 1: Na hire date se dodaje 6 meseci
- Korak 2: Prvi petak posle dana vraćenog u koraku 1 će biti identifikovan
- Korak 3: Difoult format datuma će biti formatiran za čitanje i prikaz kao datum vraćen kroz korak 2 u sličnom formatu: Friday, December 18th, 1987, i pojaviće se na izlazu pod kolonom sa nazivom “Next Evaluation“
- Nestovane funkcije se evaluiraju počevši od najdublje prema spolja

Functions Pertaining to Null Values

- Null je vrednost koja je nedostupna, nedodeljena, nepoznata ili neprimenjiva
- Kao rezultat, mi je ne mžemo testirati da li je ista kao neka druga vrednost pošto ne znamo koja je to vrednost; nije ista kao bilo koja druga vrednost
- Ako je pitanje: Da li je tačno da je X = Y ? Da bi se odgovorilo na pitanje potrebno je znati vrednosti X i Y.
- Oracle ima četiri funkcije koje koriste null vrednosti: NVL, NVL2, NULLIF, COALESCE

NVL Function

- NVL funkcija konvertuje null vrednost u poznatu vrednost fiksnog tipa podataka, bilo da je datum, karakter ili broj
- Tip podataka kolone sa null vrednostima i nova vrednost moraju biti iste
- NVL funkcija je:

```
NVL (expression 1 value that may contain a null, expression 2
value to substitute for null)
```

- NVL (vrednost ili kolona koja može imati null, vrednost za zamenu sa null)
- Sledeći upit koristi NVL funkciju sa karakter tipom podataka:

```
SELECT country_name, NVL(internet_extension, 'None')
  AS "Internet extn"
FROM wf_countries
WHERE location = 'Southern Africa'
ORDER BY internet_extension DESC;
```

- Null values are replaced with the text 'None'.

COUNTRY_NAME	Internet extn
Juan de Nova Island	None
Europa Island	None
Republic of Zimbabwe	.zw
Republic of Zambia	.zm
Republic of South Africa	.za

- Tipovi podataka sa kolonom sa null vrednostima i nove vrednosti moraju biti iste kao na sledećim primerima:

Examples:	Output	

`SELECT last_name, NVL(commission_pct, 0)`
`FROM employees`
`WHERE department_id IN(80,90);`
	Zlotkey
	Abel
	Taylor
	King
		.2
		.3
		.2
		0

`SELECT NVL(date_of_independence, No date')`
`FROM wf_countries;`
 	
`Note: date_of_independence is a Varchar2 data type`
	1-Jul-1867
	15-Sep-1821
	5-Jul-1975
	No date	

- Null vrednosti u tabelama su zamenjene korišćenjem NVL funkcije (crveno)
- NVL funkcija se može koristiti za konverziju vrednosti u kolonama koje sadrže null u broj pre kalkulacija
- Kada se izvede neka aritmetička kalkulacija sa null, rezultat je null
- NVL funkcija može konvertovati null vrednosti u broj pre izvođenja aritmetičke kalkulacije da bi izbeglo null kao rezultat
- U primeru, commission_pct kolona u employees tabeli sadrži null vrednosti

- NVL funkcija se koristi za izmenu null u 0 pre aritmetičkih kalkulacija

```
SELECT last_name, NVL(commission_pct, 0)*250
  AS "Commission"
FROM employees
WHERE department_id IN(80,90);
```

LAST_NAME	Commission
Zlotkey	50
Abel	75
Taylor	50
King	0
Kochhar	0
De Haan	0

NVL2 Function

- Funkcija NVL2 evaluira izraz sa tri vrednosti; ako prva vrdnost nije null, onda NVL2 funkcija vraća drugi izraz; ako prva vrednost jeste null, onda se vraća treći izraz
- Vrednosti u izrazu 1 mogu imati bilo koji tip podataka; izrazi 2 i 3 mogu biti bilo koji tip podataka sem LONG
- Tip podataka vraćene vrednosti je uvek isti kao tip podatka izraza 2 sem ako je izraz2 podatak karakter, u kojem slučaju vraćeni tip je VARCHAR2
- LONG je promenjiva-dužine karakter tip podataka sve do 2GB u veličini
- NVL2 funkcija je:

```
NVL2 (expression 1 value that may contain a null, expression 2
value to return if expression 1 is not null, expression 3 value to
replace if expression 1 is null)
```

- Lakši način za pamćenje NVL2 je misliti: “Ako izraz 1 ima vrednost, zameniti izraz 2; ako izraz 1 je null, zameniti izraz 3“
- NVL2 funkcija na slici koristi brojčani tip podataka za izraze 1, 2 i 3:

```
SELECT last_name, salary,
  NVL2(commission_pct, salary + (salary * commission_pct), salary)
  AS income
FROM employees
WHERE department_id IN(80,90);
```

LAST_NAME	SALARY	INCOME
Zlotkey	10500	12600
Abel	11000	14300
Taylor	8600	10320
King	24000	24000
Kochhar	17000	17000
De Haan	17000	17000

- NVL2 proverava ako izraz 1 (commission_pct) ima vrednost. Ako ima vrednost, izraz 2 se vraća (salary + (salary * commision_pct)). Ako je izraz 1 null, izraz 3 se vraća (salary)

NULLIF Function

- NULLIF funkcija upoređuje dva izraza; ako su ista funkcija vraća null; ako nisu ista funkcija vraća prvi izraz:

```
NULLIF(expression 1, expression 2)
```

- U ovom primeru, NULLIF upoređuje dužine imena i prezimena zaposlenih; ako su dužine oba imena iste, NULLIF vraća NULL (kao u redu 2 Curtis Davies), inače vraća izraz 1 LENGTH od first_name

```
SELECT first_name, LENGTH(first_name) AS "Length FN", last_name,
  LENGTH(last_name) AS "Length LN", NULLIF(LENGTH(first_name),
  LENGTH(last_name)) AS "Compare Them"
FROM employees;
```

FIRST_NAME	Length FN	LAST_NAME	Length LN	Compare Them
Ellen	5	Abel	4	5
Curtis	6	Davies	6	-
Lex	3	De Haan	7	3

NULLIF funkcije se često koriste posle izrade data migration projekata za testiranje ako je podatak u ciljnom sistemu isti kao originalni izvor sistem

- Tako da se NULLIF koristi za traženje exceptions, ne podudaranja – normalno null kao rezultat iz NULLIF je dobro, sve dok se želi da podatak iz izvora i podatak iz cilja budu isti

COALESCE Function

- COALESCE funkcija je ekstenzija NVL funkcije, osim što COALESCE može uzeti višestruke vrednosti
- Reč "coalesce" doslovce znači doći zajedno; ako je prvi izraz null, funkcija nastavlja sve do linije dok se ne nađe not null izraz
- Naravno, ako je prvi izraz sa vrednoti, funkcija vraća prvi izraz i funkcija se zaustavlja
- COALESCE fubkcija je:

```
COALESCE (expression 1, expression 2, ...expression n)
```

- Videti SELECT iskaz iz tabele zaposlenih, ako zaposleni ima vrednost (a da nije null) za commission_pct, ovo se vraća, inače ako salary ima vrdnost, vraća se salary
- Ako zaposleni commision_pct i salary su NULL, vraća se broj 10

```
SELECT last_name,
  COALESCE(commission_pct, salary, 10)
  AS "Comm"
FROM employees
ORDER BY commission_pct;
```

LAST_NAME	Comm
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
Higgins	12000
Gietz	8300

5-3 Conditional Expressions

- Tipična odluka izgleda kao: Ako biznis traži da se vodi evidencija o proteklom vremenu, onda vreme treba biti entitet ili u suprotnom vreme treba biti atribut

How Functions are Evaluated

- Proces donošenja odluka u programiranju nije mnogo različit od procesa koji koristimo u svakodnevnom životu; u SQL koristi se uključivanje odluka metodama procesiranja uslova

Conditional Expressions

- Dva uslovna izraza su CASE i DECODE
- NULLIF je logički ekvivalent CASE izraza u tome što CASE upoređuje dva izraza
- NULLIF upoređuje dva izraza, i ako dva izraza su jednaka, vraća se null; ako nisu onda se vraća prvi izraz
- Postoje dva seta komandi ili sintaksi koje se mogu koristiti za pisanje SQL iskaza: ANSI/ISO SQL 99 compilant standard statements i Oracle proprietary statements

- Dva seta sintaksi su slični ali postoje i razlike
- Ovde se koristi oba seta ali se preporučuje korišćenje ANSI/ISO SQL 99 sintaksa
- CASE i DECODE su primeri jedne od tih razlika
- CASE je ANSI/ISO 99 SQL 99 compliant statement; DECODE je Oracle Proprietary statement; oba vraćaju istu informaciju korišćenjem različite sintakse

CASE expression

- CASE izraz radi isto što i IF-THEN-ELSE iskaz
- Tipovi podataka u CASE, WHEN, ELSE izrazima moraju biti isti
- Sintaksa za CASE izraz:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
          [WHEN comparison_expr2 THEN return_expr2
           WHEN comparison_exprn THEN return_exprn
           ELSE else_expr]
END
```

- Upit proverava department_id: ako je 90 vraća 'Management', ako je 80 vraća 'Sales', ako je 60 vraća 'It' else vraća 'Other dept.'

```
SELECT last_name,
CASE department_id
   WHEN 90 THEN 'Management'
   WHEN 80 THEN 'Sales'
   WHEN 60 THEN 'It'
   ELSE 'Other dept.'
END AS "Department"
FROM employees;
```

LAST_NAME	Department
King	Management
Kochhar	Management
De Haan	Management
Whalen	Other dept.
Higgins	Other dept.
Gietz	Other dept.
Zlotkey	Sales
Abel	Sales
Taylor	Sales
Grant	Other dept.
Mourgos	Other dept.
Rajs	Other dept.
Davies	Other dept.
Matos	Other dept.
Vargas	Other dept.
Hunold	It
Ernst	It
Lorentz	It
Hartstein	Other dept.
Fay	Other dept.

DECODE Expression

- Ova funkcija evaluira izraz na sličan način kao IF-THEN-ELSE logika
- DECODE upoređuje izraz sa svakom od pretraženih vrednosti
- Sintaksa DECODE je:

```
DECODE(column|expression, search1, result1
          [, search2, result2,...,]
          [, default])
```

- Ako defoult vrednost je izbačena, null vrednost je vraćena gde pretražena vrednost nije identična sa bilo kojim vrednosti
- Ovaj upit vraća tačno isti rezultat kao prethodni CASE primer ali koristi drugačiju sintaksu

```
SELECT last_name,
  DECODE(department_id,
    90, 'Management',
    80, 'Sales',
    60, 'It',
    'Other dept.')
AS "Department"
FROM employees;
```

LAST_NAME	Department
King	Management
Kochhar	Management
De Haan	Management
Whalen	Other dept.
Higgins	Other dept.
Gietz	Other dept.
Zlotkey	Sales
Abel	Sales
Taylor	Sales
Grant	Other dept.
Mourgos	Other dept.
Rajs	Other dept.
Davies	Other dept.
Matos	Other dept.
Vargas	Other dept.
Hunold	It
Ernst	It
Lorentz	It
Hartstein	Other dept.
Fay	Other dept.